EMOTIONS EXPERIENCED BY FIRST-YEAR ENGINEERING STUDENTS DURING PROGRAMMING TASKS

by

Syedah Zahra Atiq

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy

School of Engineering Education

West Lafayette, Indiana

August 2019

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

I dedicate this dissertation to my family

Atiq, Samra, Safee, Hajra, Mohsina, Madiha, Fazal, and Inayat

For their unconditional love, and their unwavering support

ACKNOWLEDGMENTS

I would like to express my deepest appreciation and thanks to everyone who helped me complete my dissertation and my doctoral studies.

My family for being my pillars of support.

My parents (Samra and Atiq) for believing in me and supporting all my decisions and endeavors. I am proud to call myself your daughter.

My brother, sister-in-law, and sisters (Safee, Madiha, Hajra, and Mohsina) for always giving me sincere advice, and being patient with me during this journey.

My nephews (Fazal and Inayat) for being a constant source of joy for me.

My dissertation committee for their support, suggestions, and feedback.

Dr. Michael C. Loui for his untiring support and mentorship. I appreciate the time and effort he spent in providing thoughtful and thorough feedback.

Dr. Jennifer J. DeBoer for providing me the opportunity to work on diverse and impactful research projects.

Dr. Brent K. Jesiek for his valuable insights about ENGR 132 and qualitative research methods.

Dr. Edward J. Berger for lending me his equipment for collecting data for this study.

Dr. Idalis Villanueva for her input on collecting and analyzing the electrodermal data.

Samia Najeeb for her friendship and support. I cherish her wisdom especially when the going gets tough.

Saira Anwar for being my family away from home and for always being there when I need her. I also want to thank Saira for taking out time to assist me with parts of data analysis.

Dr. Ali Raza Jafri for being a friend, a mentor, and for introducing me to engineering education at Purdue University. I would not be writing this, if I had not called him for advice on a winter morning in January 2013.

All my friends and peers in West Lafayette who contributed to my journey in big and small ways. A special shout out to Genisson Coutinho, Juan David Ortega-Alvarez, Karen De Urquidi, Matilde Sanchez-Pena, Emilie Siverling, Hoda Ehsan, and Hossein Ebrahiminejad. Thank you for all the times you have been there for me.

Loretta McKinnis and Carol Brock for believing in me, and for all the efforts they made to help me achieve my academic goals.

All members of the DeBoer lab for the feedback they provided over the years to help me refine my research ideas.

Dr. Heidi Diefes-Dux for sharing her experiences and insights about how ENGR 132 is designed and taught.

Dr. Ruth Streveler and Dr. Audeen Fentiman for their mentorship and support over the years.

Graduate school at Purdue University for the Bilsland Dissertation Fellowship that supported the last year of my doctoral studies.

Lastly, this study would not have been possible without the time and effort of all participants of this study. Thank you!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

Author: Atiq, Syedah Zahra. Ph.D.
Institution: Purdue University
Degree Received: August 2019
Title: Emotions Experienced by First-Year Engineering Students During Programming Tasks
Committee Chair: Michael C. Loui and Jennifer J. DeBoer

Computer programming is a difficult undertaking for novices, requiring a lot of patience and persistence. Hence, in a programming class, students experience an array of emotions that may promote or thwart their performance and learning. For instance, frustration may reduce students' motivation to learn programming. In extreme cases, continued frustration may convince students to abandon plans for engineering or computing careers. Even though emotions are crucial for learning, very little is known about how students experience emotions in an introductory programming class.

In this dissertation, I report my investigation of emotions experienced by first-year engineering students during programming tasks, the reasons for experiencing those emotions, and the self-regulation strategies they adopted to cope with those emotions.

I recruited 17 novice first-year engineering students taking an introductory programming class for the first time. Each participant took part in two sessions, which collected multi-modal data: programming task and retrospective think-aloud interview. During the programming task, participants worked on four programming problems for thirty minutes. In this session, I collected five types of data: screen capture, facial expressions, eye-gaze data, electrodermal activity, and survey instruments that assessed emotions and neuroticism. After the programming task, I conducted a short post-task interview to ask follow-up questions. The participants returned three to seven days after the programming task for a retrospective think-aloud interview. During this session, participants viewed a video of their actions during the programming task. After every two minutes of viewing, I paused the video and asked about the emotions they experienced during that segment.

The overarching findings from this study suggest the students experienced frustration most frequently while working on programming problems. Students also experienced multiple emotions because of the same event. For instance, one student reported feeling annoyed because she had

made a mistake, but she also experienced joy when she was able to fix the mistake. Findings of this study also suggest that most students tended to persevere despite encountering errors. When they overcame the errors, they experienced joy and pride.

A better understanding of student emotions may help educators design curriculum and pedagogy to help mitigate the effects of negative emotions, and to promote positive emotions. This improved curriculum and pedagogy may eventually help students maximize their learning and performance in programming courses. Subsequently, student motivation and interest in programming may also be increased by using this improved and enhanced curriculum and pedagogy.

CHAPTER 1: INTRODUCTION

Background of the Problem

Computer programming is considered a necessary skill for engineering students (Sun & Sun, 2011). Consequently, programming courses are introduced to undergraduates early in their engineering education. However, learning programming is a difficult undertaking (Robins, Rountree, & Rountree, 2003). It requires patience and persistence on the part of the student (Rogerson & Scott, 2010). It is also challenging because novice students may not have accurate mental models required to learn programming (Guzdial, 2015; Robins et al., 2003). Hence, students in a programming course may experience a wide array of emotions that may positively or negatively impact their learning and academic performance (Pekrun & Perry, 2014; Zeidner, 2014). Students' responses to these emotions may affect their performance in engineering courses and their persistence in engineering programs. For instance, engineering students who struggle in a programming course may leave engineering (Meyer & Marx, 2014; Secules, Elby, & Gupta, 2016; Secules, Gupta, Elby, & Turpen, 2018). Literature also suggests that students may be academically proficient but may still encounter difficulties in a programming course (Meyer & Marx, 2014). Students from minority ethnic and racial backgrounds may also struggle in a programming course because of various reasons. For instance, one reason is that minority students do not have access to resources that are provided to majority, mainly white and Asian students (May & Chubin, 2003). All these challenges associated with learning programming may lead students to drop out of engineering or switch to another engineering major (Meyer & Marx, 2014; Suresh, 2006).

Besides programming courses, researchers have sought to identify other causes of attrition from engineering programs. For this purpose, researchers have studied background factors (May & Chubin, 2003; Wladis, Hachey, & Conway, 2014) and non-cognitive factors like students' self-efficacy beliefs (Booth, Doyle, & Musson, 2013) and their effect on student retention in engineering. However, little is known about how emotions affect student performance, learning, and retention in engineering.

A few researchers are beginning to investigate student emotions in the context of engineering education. Several researchers have adopted a multi-modal approach to understand student emotions in engineering education (Cheng, Husman, Fishman, & Barnes, 2015; Husman, Cheng, Puruhito, & Fishman, 2015; Villanueva, Raikes, Ruben, Schaefer, & Günther, 2014; Villanueva, Campbell, Raikes, Jones, & Putney, 2018; Villanueva, Valladares, & Goodridge, 2016). These authors have used a combination of survey instruments and physiological biomarkers (saliva and electrodermal activity) to understand student emotions in engineering and STEM courses. However, this area of research is very new and has room for further investigation. The goal of this research is to extend the engineering education scholarship by understanding student emotions during programming tasks. Since introductory programming courses may lead students to leave engineering (Secules, Elby, & Gupta, 2016), it is imperative to understand the multiple factors, including emotions, that may lead students to leave engineering, or to feel that they are "not cut out" for engineering.

Purpose of the Study and Research Questions

This study adopts a multi-modal approach to understand engineering students' emotions, as they work on a programming task. Specifically, this study uses various self-report data (interviews and surveys), observations (from video) and physiological biomarkers (electrodermal activity) to understand student emotions, how emotions change as a result of working on a programming task, and what self-regulation strategies students use to deal with the emotions. For this purpose, this study uses established theoretical frameworks (control-value theory of achievement emotions and self-regulated learning) to guide different stages of research. In this dissertation study, I answer the following research questions:

1. (a) What emotions do first-year engineering students experience while they work on a computer programming task?

   (b) What reasons do students describe for experiencing the different emotions?
2. How do student emotions change as a result of working on programming problems?
3. What self-regulation strategies do students use to cope with these emotions?

Significance of the Study

This study has two significant features. This study employs a novel research design and combination of methods. Specifically, I am using an inter-disciplinary and multi-modal methods approach to understand a complex phenomenon like emotions (Afzal & Robinson, 2015). I am using the control-value theory of achievement emotions to guide the different stages of this study. This theory has been extensively used to understand student emotions in other STEM fields (Schukajlow, Rakoczy, & Pekrun, 2017), but this theory has not been used in the context of computer programming.

For almost two decades, researchers in different fields have investigated students' emotions in the academic context (Linnenbrink, 2006; Pekrun & Linnenbrink-Garcia, 2014b). As a result, researchers in various fields have called for more interdisciplinary investigation to better understand student emotions (Calvo & D'Mello, 2010; Pekrun & Perry, 2014). Although this research study is being done by a single researcher, this research is at the confluence of many different fields, that is, engineering, engineering education, computer science, computer science education, psychology, education, neuroscience, and affective computing. In other words, this research is grounded in evidence, theories, and methods from these fields and is interdisciplinary in nature.

A better understanding of student emotions may help educators design curriculum and pedagogy to help mitigate the effects of negative emotions, and to promote positive emotions. This improved curriculum and pedagogy may eventually help students maximize their learning and performance in programming courses. Subsequently, student motivation and interest in programming may also increase by using this improved and enhanced curriculum and pedagogy. A better understanding of student emotions may also help develop affective-sensitive learning environments for programming. These learning environments will not only detect student emotions to provide necessary interventions but will also provide a fine-grained data on out-of-class student learning and behaviors (Bosch & D'Mello, 2015).

Scope and Delimitations

This research study is situated in the context of the First-year engineering (FYE) program at Purdue University. The sample comprises 17 domestic FYE students taking an introductory programming class for the first time. This study excludes expert programmers because novices and experts comprehend programming differently (Robins et al., 2003). Moreover, this study also excludes international students because there are differences in the way emotions are experienced across nations and cultures. These differences may be due to many factors, for example, language and the ways in which emotions are expressed across cultures (Elfenbein & Ambady, 2002; Jack, Garrod, Yu, Caldara, & Schyns, 2012).

The overarching purpose of this dissertation is to use multi-modal data to understand the range of emotions that FYE students experience and the self-regulation strategies they adopt during programming tasks. The study is primarily qualitative (data and analyses), with minor quantitative elements. To present the findings, I present rich descriptions along with interpretations that are grounded in established theoretical frameworks and literature. Moreover, preliminary findings from this dissertation study have been published in two conferences (Atiq, 2018b, 2018a).

This study does not aim to establish a causal relationship between student emotions and constructs like performance and self-efficacy. The study is conducted in a lab setting. However, there is some evidence of transferability to the classroom context.

Theoretical Framework

What are Emotions and Achievement Emotions?

Emotions are multi-componential psychological processes that include different affective, cognitive, motivational, expressive, and physiological components (Scherer, 2009). To understand each component, consider the example of a student who is anxious before performing a programming task (see Figure 1). The affective component is the subjective emotional feeling that the student experiences (e.g., feeling nervous), the cognitive component deals with the emotion specific thoughts (e.g., worry), the motivational component deals with the students' behavioral responses (e.g., avoidance behavior), the expressive component is how the emotions are expressed (e.g., facial expressions), and physiological component deals with physiological changes in the body caused by emotions (e.g., changes in skin conductance caused by perspiration, or increased heart rate).

Most of the literature about emotions uses the terms *moods* and *emotions* interchangeably (Linnenbrink, 2006). However, there is a difference between the two. While moods are longer lasting and may not have a referent, emotions consist of short and possibly intense episodes in response to a stimulus (Pekrun & Linnenbrink-Garcia, 2014b). A person's situation and history play a significant role in the emotions they may feel. Two individuals may experience different emotions while facing the same situation, because of various personal histories and cultural differences (Pekrun & Perry, 2014).

Emotions in education are crucial for academic achievement, psychological well-being, motivation, beliefs in self-concept, and self-efficacy of both teachers and students. Hence, it is important to understand what emotions are and how they affect us in so many ways (Zeidner, 2014). However, researchers are just beginning to comprehend the role of emotions in an academic setting (Pekrun & Linnenbrink-Garcia, 2014a).

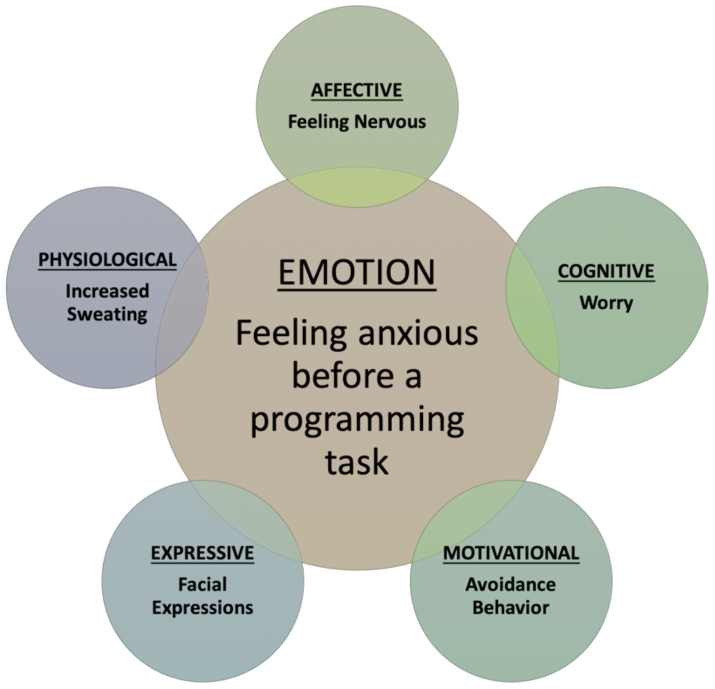


Figure 1. Multi-Componential Nature of Emotions

For the purpose of this research, I have used two theoretical frameworks to guide the research design, data collection, data analysis, and interpretation of data. These theoretical frameworks are control-value theory of achievement emotions (Pekrun & Perry, 2014) and self-regulated learning (Zimmerman, 2002, 2005; Zimmerman & Campillo, 2003). The following sections summarize each of these frameworks.

Control-Value Theory of Achievement Emotions

Control-value theory (CVT) posits that achievement emotions impact learning and academic performance through cognitive and motivational mechanisms; hence it is called the cognitive-motivational model of emotion (Pekrun & Perry, 2014). This theory provides an integrated framework for studying emotions in academic settings. Hence, this theory will primarily be used to design the study, to interpret data, and to discuss the findings of this study.

According to the control-value theory of emotions, achievement emotions link directly to achievement activities or achievement outcomes (Pekrun & Perry, 2014). This theory views emotions as a set of interrelated psychological processes that has multiple components, i.e., affective, cognitive, physiological, and motivational (Pekrun, Goetz, Titz, & Perry, 2002; Pekrun & Perry, 2014). When assessing student emotions, it is crucial to understand each of these components.

Pekrun describes achievement emotions in a three-dimensional taxonomy. The three dimensions of the taxonomy are object focus, activation, and valence. The *object focus* can be either activities or the outcomes. When studying emotions, the focus should be placed not only on the activity the students engage in, but also on the outcome of the activity. The *valence* of emotions can either be positive/pleasant or negative/unpleasant. *Activation* refers to the emotional arousal that a person experiences physiologically in response to a particular emotion. Both positive and negative emotions can be activating (happiness, hope, pride, anxiety, anger, shame), or deactivating (relief, hopelessness, boredom, satisfaction) (Pekrun & Perry, 2014). Figure 2 shows this three-dimensional taxonomy in the 'EMOTION' section.

Structure of the Theory

The central part of the theory revolves around the premise that students experience positive emotions when they feel in control of activities, or when they value activities. Similarly, students experience negative emotions when they feel out of control of activities, or when they do not value the activities. Hence, *control appraisals* are the controllability of actions and their outcomes, and *value appraisals* are the subjective importance of activities and their outcomes. Different control and value appraisals lead to different emotions.

The CVT has three main components: antecedents, emotions, and outcomes. *Antecedents* influence emotions either directly or indirectly. *Emotions* in turn influence achievement. *Achievement* affect subsequent emotions and antecedents. Antecedents can either be proximal (they directly affect emotions) or distal (they influence emotions by first influencing the proximal antecedents). Hence, achievement emotions, their antecedents, and outcomes are linked together by reciprocal causation, resulting in a feedback loop (see Figure 2). This cyclic nature of different components in this theory also implies that emotions can be regulated by targeting any of the elements of the feedback loop (Pekrun & Perry, 2014).

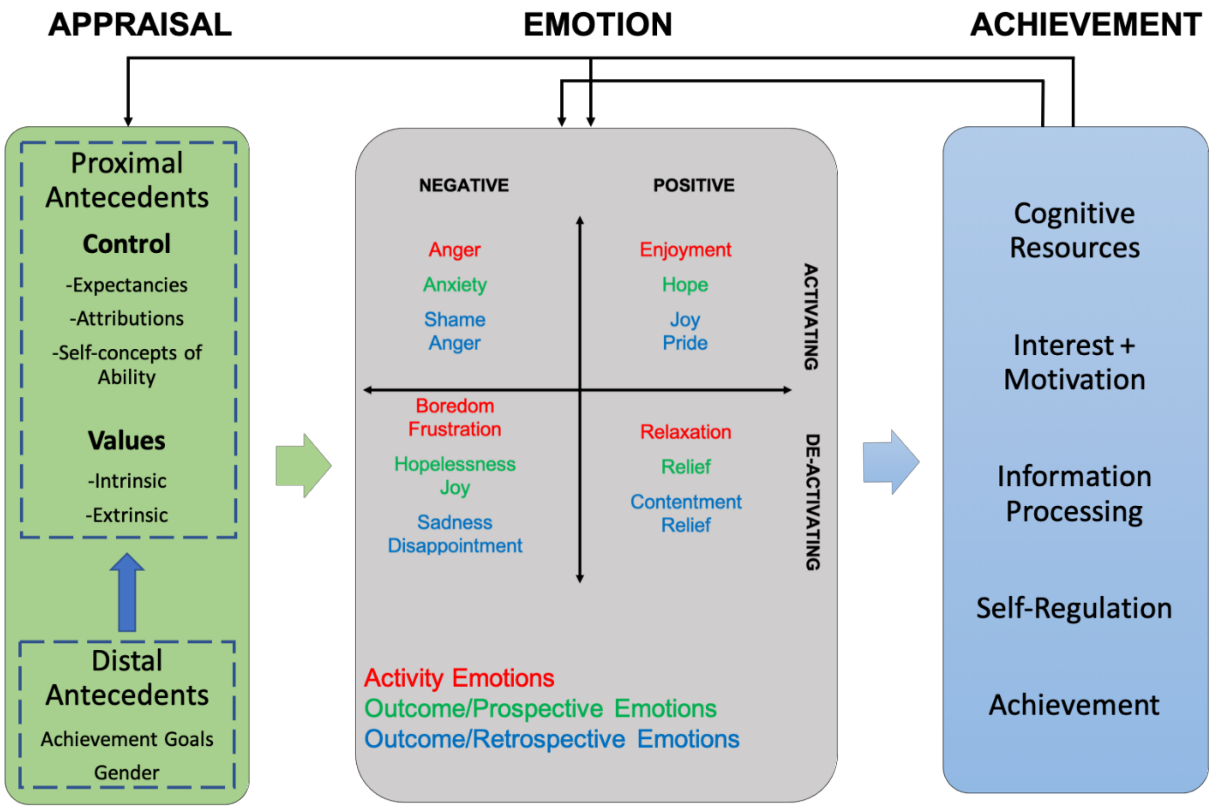


Figure 2. Structure of the Control-Value Theory of Achievement Emotions
Adapted from (Pekrun & Perry, 2014)

Proximal Antecedents

According to the CVT, different control and value appraisals evoke different prospective outcome emotions, retrospective outcome emotions, and activity emotions (Pekrun & Perry, 2014).

Students experience *prospective emotions* like joy when there is high perceived control, while they experience hopelessness when there is lack of perceived control. Similarly, students experience hope when control is uncertain, and they anticipate success. On the other hand, students experience anxiety when control is uncertain, and when they expect failure.

Retrospective emotions like disappointment and relief are assumed to depend on the perceived match between expectations and outcomes. The anticipated success that does not occur leads to disappointment, while anticipated failure that does not occur leads to relief. Furthermore, attributing success to oneself leads to pride, while attributing failure to oneself leads to shame. Attributing success to others leads to gratitude, while attributing failure to others leads to anger.

A*ctivity emotions* are proposed to depend on competence appraisals and intrinsic values. For instance, competence and positive intrinsic qualities of action and activity lead to enjoyment of the activity, while lack of activity incentive values leads to boredom. Negative activity incentive values lead to anger and frustration.

Distal Antecedents

According to CVT, distal antecedents like achievement goals and gender influence emotions indirectly by first influencing the control and value appraisals (Pekrun & Perry, 2014). There are two types of *achievement goals*: mastery goals and performance goals.

The CVT argues that *mastery goals* focus on the students' mastery of the activity and the value an individual places in that activity. For instance, mastery goals normally foster positive activity emotions (enjoyment) and reduce negative activity emotions (boredom). On the other hand, *performance* goals focus on the perceived controllability and positive value of successful outcomes. For instance, performance goals frequently foster positive outcome emotions like hope and pride.

Gender linked stereotypes normally influence the socialization of appraisals, translating to gender-based differences in achievement emotions. For instance, for mathematics women experience more anxiety and less enjoyment than their male counterparts (Frenzel, Pekrun, & Goetz, 2007).

The Achievement Emotions Questionnaire (AEQ)

CVT is the theoretical framework on which the Achievement Emotions Questionnaire – AEQ was designed (Frenzel et al., 2007; Pekrun, Götz, & Perry, 2005). AEQ is a multidimensional self-report instrument that can be used to assess college students' achievement emotions. AEQ is composed of three sub-questionnaires used to evaluate emotions while students attend class (80 items), while studying (75 items), or while taking a test (77 items). The three sub-questionnaires can be used together or individually. Each sub-questionnaire is further divided into emotions students may experience before, during, or after the class, after studying, or after taking a test. The questionnaire can be adapted to different courses and for concurrent assessment activities (Pekrun et al., 2005).

Self-Regulated Learning

Self-regulation learning is a process in which students adapt and orient their thoughts, feelings, and behaviors to attain their goals (Zimmerman, 2002, 2005; Zimmerman & Campillo, 2003). Goals are critical to self-regulation, as they are a student's representation of what they want to occur and what they want to avoid in the future. According to Zimmerman's framework of self-regulated learning, every action (performance) is informed by selection of appropriate strategies (forethought) and is followed by an evaluation to gather feedback for the next action (self-reflection). Figure 3 shows the processes and sub-processes of self-regulated learning.

Forethought

In the forethought stage, students prepare for the upcoming activity (Pintrich & Schunk, 2001). During this stage, students take time to determine how to best perform a task. Forethought stage consists of two types of processes: task analysis (goal setting and strategic planning), and self-motivation (Zimmerman, 2002).

Task Analysis

During task analysis, students set specific goals, and make strategic plans to help them achieve these goals (Zimmerman, 2005). *Goal setting* is a type of task analysis during which students decide upon the intended outcomes of their performance (Zimmerman, 2005). *Strategic* *planning* is a type of task analysis in which students think about different strategies to improve their performance (Zimmerman, 2002). Students need task-appropriate strategies to perform a task in an optimal way. During strategic planning, students select strategies and methods that they will implement in the performance stage (Zimmerman, 2005).

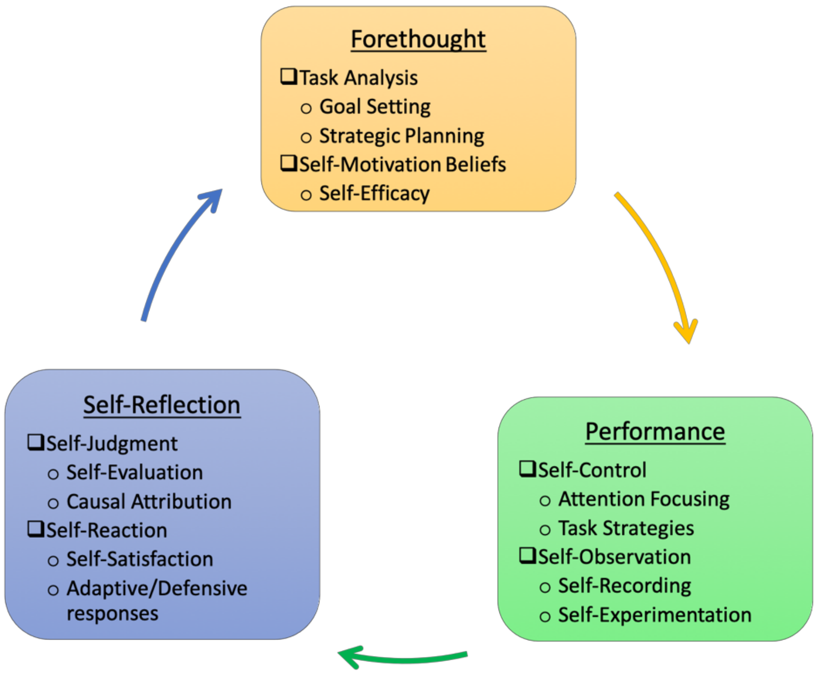


Figure 3. Processes and Sub-Processes of Self-Regulated Learning
Adapted from (Zimmerman, 2002)

Self-Motivation Beliefs

Task analysis is driven by students' motivation beliefs (Zimmerman, 2005), especially their self-efficacy beliefs. A person's self-efficacy is their belief in their own capability to succeed at certain tasks (Bandura, 1977; Zimmerman, 2005). There is a difference between self-efficacy and confidence. While self-efficacy is belief about one's ability to perform a task, confidence refers to one's belief about the likelihood of succeeding. Self-efficacy is task dependent: that is, a person may have high self-efficacy beliefs about one task but not about another (Bandura, 1991). For instance, a student in an introductory programming class may have high self-efficacy beliefs about conditional statements, but not about loops.

Performance

In the performance stage, students work on the activity (Zimmerman, 2002). During this stage, students engage in two types of processes that may affect their motivation and learning: self-control and self-observation (Pintrich & Schunk, 2001).

Self-Control

During self-control, students implement the strategies they selected during the forethought stage. Applying these strategies help students focus on the task and optimize their performance (Zimmerman, 2005). Students adopt two self-control strategies, that is, attention focusing and task strategies. *Attention focusing* is a form of self-control, which students implement to improve their concentration on the task they are working on (Zimmerman, 2005). Students also use attention focusing strategies to divert all external distractions that they may encounter while working on a task. Students use various *task strategies*, which assist them in their learning and performance (Zimmerman, 2005).

Self-Observation

During self-observation, students track their performance, the surrounding conditions, and the outcomes of the performance (Zimmerman, 2005). Students apply two types of self-observation techniques: self-recording, and self-experimentation. During self-recording, students keep records of their performance. They can use these records later as feedback for improvement (Zimmerman, 2005). During self-experimentation, students vary different aspects of their work to see what works for them (Zimmerman, 2005). Students typically engage in self-experimentation when their work on an activity did not yield desired results. For instance, novice programming students may engage in self-experimentation to resolve the syntax errors they encounter in their code.

Self-Reflection

Self-reflection is a person’s ability to think about their thought process and experiences (Bandura, 1989). By doing self-reflection, people can generate knowledge about themselves and the world around them. In self-regulated learning theory, self-reflection consists of two processes: self-judgment and self-reaction (Zimmerman, 2002).

Self-Judgment

During self-judgment, students evaluate their performance and attribute a cause to the outcomes (Zimmerman, 2005). Self-judgment includes two processes: self-evaluation and causal attribution. Self-evaluation entails comparison of one's performance with a goal (Zimmerman, 2005). After self-evaluation, students attribute the outcome to a cause (Zimmerman, 2005). For instance, if a student performs poorly on a test, they could blame their poor performance on the failure of the strategy they adopted, or on their low ability, hence, discouraging themselves from improving in the future (Weiner, 1972).

Self-Reaction

Self-reaction involves two processes: self-satisfaction and adaptive/defensive responses (Zimmerman, 2005). Self-satisfaction entails students' perception of satisfaction or dissatisfaction with their performance. Self-satisfaction also entails the affect students experience after their performance on the task (Zimmerman, 2005). Adaptive/defensive inferences are conclusions students make on how they need to change their self-regulatory behaviors for the subsequent actions they might take. Adaptive inferences normally lead students to adopt better strategies and forms of self-regulation. On the other hand, defensive inferences undermine successful adaptation, even though they prevent students from future dissatisfactions (Zimmerman, 2005).

Literature Review

Emotions in the Context of Engineering Education

Emotions play a vital role in the academic achievement of students. However, there is little previous research on emotions in engineering education, and most of the existing work focuses on STEM subjects, specifically mathematics, with anxiety being the most studied emotion (Jamil, Saad, Jaafar, & Abdullah, 2011; Leppävirta, 2011; Schukajlow et al., 2017). Some work has, nevertheless, started to emerge in the engineering education research community. One qualitative research study uses narrative analysis of student reflections to understand the role of emotions in learning while experiencing an integrated curriculum within an interdisciplinary, project-based design studio (Kellam, Costantino, Walther, & Sochacka, 2011). The findings suggest that students at the start of the semester were anxious and frustrated, but as the semester progressed, they began to feel more positive emotions as they learned more self-regulation strategies. This study relied solely on students' narratives, which provides only one perspective. For emotions, which are multi-componential and evolve rapidly, it is important to capture multiple perspectives. To understand student emotions, researchers must employ multi-modal data collection and analysis strategies (Calvo & D'Mello, 2010; Pekrun & Linnenbrink-Garcia, 2014a).

In the realm of engineering education, Villanueva and colleagues have proposed a protocol that employs multi-modal data such as biomarkers (e.g., electrodermal activity and saliva samples), and self-report methods to understand students' emotional states in a lab setting (Villanueva et al., 2016). Such protocols are appropriate in helping researchers identify emotions and their regulation in instructional contexts. A similar study aimed to understand student emotions in an engineering ethics course by using student salivary samples and self-report data (Husman et al., 2015). The salivary biomarker contains a hormone called cortisol that indicates the levels of stress experienced by the person. Students who reported enjoyment experienced lower stress levels, and vice versa. These studies are promising and provide novel ways of studying student emotions. However, there is a dearth of such literature in the context of computer programming courses for engineering students. On the other hand, fields like computer science education and affective computing have attempted to understand student emotions as they work on programming tasks. It would be useful to know how researchers in these domains are trying to understand student emotions in the context of computer programming.

Emotions in the Context of Computer Programming

Many universities offering STEM undergraduate programs require students to learn computer programming during their freshman year. In recent years, researchers have been studying the emotions of students while they learn how to program (Bosch & D'Mello, 2015; Bosch, D'Mello, & Mills, 2013; Kinnunen & Simon, 2010a, 2010b, 2011, 2012; Lishinski, Yadav, & Enbody, 2017; Lishinski, Yadav, Good, & Enbody, 2016). In a series of studies, Kinnunen and Simon developed a framework that describes how computing majors experience emotions during each stage of tackling programming assignments in an introductory course (Kinnunen & Simon, 2010c, 2012a). For instance, in the “Getting Started” stage, students who do not know how to proceed with solving a problem may feel confused or puzzled. On the other hand, in the Succeeding stage, students who successfully solve a problem may experience a feeling of relief and accomplishment. This work by Kinnunen and Simon studied affective states by interviewing the students four times over the course of the semester. However, they did not study the evolution of students’ affective states during the semester. Building on the work of Kinnunen and Simon, Lishinski et al., (2017) quantitatively investigated the connection of emotions with learning outcomes and self-efficacy beliefs in an introductory computer programming course (Lishinski et al., 2017). According to this study, frustration was the most frequent emotion. Moreover, students’ emotions positively correlate with their performance.

The original aim of Kinnunen and Simon’s work was to identify the different stages that students go through to solve programming problems (Kinnunen & Simon, 2010b), for which they use a grounded theory methodology. The authors identified student emotions as a by-product of their research. Hence, their research on emotions is not grounded in existing theories of emotions, even though many theories exist. Since Lishinski et al., (2017) used Kinnunen and Simon’s work as a foundation for their work, but their research is also not grounded in any existing theory of emotion. Hence, these studies neglect to understand other emotions that are important for student learning and performance during programming tasks.

Mercedes and colleagues, on the other hand, investigated how affective states of novice programmers evolved over time and how emotions impacted students’ performance (Mercedes, Rodrigo, Baker, et al., 2009; Mercedes, Rodrigo, Shaun, & Baker, 2009; Mercedes, Rodrigo, Sugay, Baker, & Tabanao, 2009). Specifically, they collected data during five lab sessions over a nine-week period and they analyzed the affective states of freshmen computer science students who were taking an introductory programming course. The researchers collected two types of data: student observations by trained observers and student activity logs. The affective states studied were boredom, confusion, delight, surprise, frustration, flow, and neutral. Moreover, the student activity logs collected for each student constituted the computer number, timestamp, error message, file name, and source code. In the first stage of the study, Mercedes, Rodrigo, Sugay, et al., (2009) observed how student affective states changed over a period, concluding that confusion, flow, and neutrality varied significantly over time and variations in frustration were marginally significant. The authors also found that students seemed to adapt easily to introductory topics like outputs and conditions but found constructs about object-oriented programming particularly confusing.

During the next stage of their research, Mercedes, Rodrigo, Baker, et al., (2009) employed student activity logs in addition to the observation data, to study the observable affective states that predict students' degree of achievement. The results indicated that students who asked for help from the instructor about integrated development environment related tasks did not perform well on the midterm exam. In the third stage of the study, Mercedes, Rodrigo, Shaun, et al., (2009) attempted to detect student frustration in a programming course using coarse-grained data such as student compilation logs. The results indicated that students' level of frustration could be detected based on quantitative attributes of the student compilation logs (total number of errors, the number of compilations, pairs of consecutive compilations with the same error, and the average time between compilations). These research studies by Mercedes and colleagues aim to develop intelligent tutors to help students learn programming. Hence, they use external observations and student activity logs to identify the student emotions and when students experience certain emotions. These studies do not directly interact with the students, hence, missing out on students' descriptions of their own experiences. It would be beneficial to use personal accounts from students to design intelligent programming tutors, capable of identifying student emotions.

Emotions in programming courses have also been studied for non-computing majors. Bosch and D'Mello (2015) have studied the affective experience of psychology undergraduate students taking an introductory programming course. Students in this study were tested individually in a two-hour session, with each session consisting of three stages: scaffolded learning, fade out, and retrospective affect judgment. The first two stages provided students the opportunity to work on basic programming problems with and without hints, respectively. After the completion of these two stages, student emotions were measured using a retrospective judgment protocol, which enabled students to reflect on their experience while watching the videos of their face and their on-screen activity. The learning environment kept track of student interaction events like key presses and system actions like providing feedback to students. The results of this study showed that engagement, confusion, frustration, boredom, and curiosity were the most frequently occurring affective states. Some other affective states like anxiety, happiness, anger, surprise, disgust, sadness, and fear were also present but were infrequent.

Methods for Assessing Emotions

For assessing emotions, different data collection methods have advantages and disadvantages, and no single method can answer all research questions. Most research on emotions in educational settings uses quantitative, deductive, and experimental approaches (Pekrun & Linnenbrink-Garcia, 2014a). For instance, previous research provides various approaches to assess emotions using self-report, physiological, and behavioral strategies (Mauss & Robinson, 2009). Although these strategies have many advantages, they fail to capture the richness of emotions. For this reason, studies of emotions in educational settings can benefit from a multi-modal approach integrating deductive and inductive methods, quantitative and qualitative data, and laboratory and field studies (Afzal & Robinson, 2015; Pekrun & Linnenbrink-Garcia, 2014a). The following sections summarize the literature on different methods that have been employed to study student emotions in the programming context and points out some areas where the field can benefit from future research.

Quantitative Methods for Assessing Emotions

A conventional method for measuring emotions is by using validated self-report instruments, mainly because self-report instruments are easy to implement (Pekrun & Linnenbrink-Garcia, 2014b). For student emotions in programming courses, researchers have used validated instruments to measure anxiety. Most of these instruments have been used to measure student anxiety (Leso & Peck, 1992), relationship between computer experience, computer anxiety, and self-efficacy of undergraduate computer science students (Connolly, Murphy, & Moore, 2009; Doyle, Stamouli, & Huggard, 2005), interaction between factors (e.g., computer anxiety, programming anxiety) influencing student achievement in programming (Owolabi, Olanipekun, & Iwerima, 2014), and students' programming mindset and their self-perception as programmers (Scott & Ghinea, 2014).

Although using validated self-report surveys to collect data is a standard research practice, these surveys may not provide an accurate measure of emotions because self-report measures may be subject to inaccurate memory recall (Paulhus & Vazire, 2009). Observations, on the other hand, provide a better understanding of the context and phenomenon being studied by the study (Kawulich, 2005). Mercedes and colleagues (2009a, 2009b, 2009c) used an observation protocol called Baker Rodrigo Ocumpaugh Monitoring Protocol (BROMP) to collect student emotional reactions while they performed programming tasks in a lab (Baker, Corbett, Roll, & Koedinger, 2008; D'Mello, Picard, & Graesser, 2007; Ocumpaugh, Baker, & Rodrigo, 2015). Observations may provide a good understanding of how students behave during the lab session but capturing student' emotions by observation alone is not a simple task. Capturing emotions using a protocol like BROMP requires highly trained observers, who must use their judgment to record student emotions. Moreover, BROMP quantifies student emotions, which loses rich insights that students may provide if they were to reflect on their experiences.

Researchers have also used compilation and activity logs to determine if a student is stuck somewhere or may be experiencing negative emotions (Lee, Rodrigo, Baker, Sugay, & Coronel, 2011; Mercedes, Rodrigo, Baker, et al., 2009). Student activity logs may provide an estimate of when students get stuck while programming and hence measures student frustration, but they may not be appropriate to measure other emotions experienced in a programming course. Qualitative research, in such cases, may play a useful role in helping researchers understand emotions as this type of investigation enables data collection unconstrained by the researchers' perspective (Afzal & Robinson, 2015; Pekrun & Linnenbrink-Garcia, 2014).

Qualitative Methods for Assessing Emotions

Qualitative research can provide rich, insightful explanations on the emotion's students feel during a programming session. However, there is a shortage of qualitative studies that focus on understanding student emotions while they learn how to program. This may be because many researchers researching this area are educators with very little training in education research (Cooper et al., 2016). In a qualitative study, Kinnunen et al. used a semi-structured interview to collect data from nine CS students taking an introductory undergraduate programming course (Kinnunen & Simon, 2010a, 2010b, 2011, 2012). They conducted four interviews per student over the course of the semester to understand the emotional toll that programming takes on students. Rogerson et al., (2010) on the other hand performed a phenomenological study to examine how students' experiences of learning programming were affected by fear. Finally, Hawi (2010) used narrative interviews of business computing students to identify causal attributes of success and failure in a programming course.

Multi-Modal Data for Assessing Emotions

A few studies exist that employ multi-modal data to understand the relationship between the affective and cognitive achievement of students enrolled in programming courses (Denton & McKinney, 2004) and to investigate the effects of pair-programming on students in an introductory programming course (Brougham, Freeman, & Jaeger, 2003). In this study, Bosch et al. (2015) studied the emotions psychology students experience while taking an introductory programming course. Students worked on programming problems while the system recorded their facial expressions and keystroke data. After completing the task, students labeled their own emotions in a retrospective affect judgment stage at random intervals. Although these studies are collecting and analyzing quantitative and qualitative data, most of these data are quantitatively analyzed, hence, missing out on the rich descriptions of qualitative insights. Researchers studying emotions have advocated the integration of both quantitative and qualitative data to understand the complex nature of emotions (Pekrun & Perry, 2014).

Critique and Justification

From the review of literature, it is evident that there are no multi-modal studies that investigate student emotions in the context of learning programming in an introductory course. Hence, following the call of researchers working on studying emotions (Afzal & Robinson, 2015; Pekrun & Linnenbrink-Garcia, 2014a), I have designed a multi-modal study to investigate emotions experienced by first-year engineering students during programming tasks.

Most previous studies identify certain frequently occuring emotions like frustration (Kinnunen & Simon, 2012; Lishinski et al., 2017), and patterns of emotions like confusion followed by frustration (Bosch & D'Mello, 2015). Most of these studies focus on the impact of negative emotions (e.g., frustration) on learning outcomes or self-efficacy. However, these studies do not explain the situations that trigger certain emotions in students. The current research on emotions also does not explain positive emotions or a mix of multiple emotions in the context of programming. While negative emotions may hamper student performance, it is not always the case.

Moreover, previous literature identifies emotions experienced by students, but do not describe in detail the events that trigger these emotions, and how students cope with and respond to those emotions. Hence, my study not only provides rich descriptions on how students experience both positive and negative emotions, and the coping mechanisms, but also provides triangulation using secondary sources of data. Finally, I am using established theories to inform all parts of my research study: unlike previous studies the findings of this study are theoretically grounded.

Dissertation Organization

This dissertation is organized into 5 chapters. Chapter 1 introduces the research study by providing the background, purpose, research questions, significance, and scope. Chapter 1 also provides an overview of the theoretical frameworks used to guide different stages of the study. This chapter also presents a review of literature about research on emotions in the context of engineering and computing education. Chapter 2 explains the multi-modal methodology, participant selection and recruitment, and data collection and analysis techniques used to answer the research questions of this study. Chapter 3 provides a detailed account of the findings of the study, while Chapter 4 discusses findings, transferability to other contexts, and explains some limitations of the research. Chapter 5 concludes the dissertation by describing the implications of this study for educators and researchers, limitations, and provides suggestions for future research.

CHAPTER 2: METHODOLOGY AND METHODS

Philosophical Underpinnings and Positionality

The purpose of this section is to promote transparency by candidly disclosing my epistemological commitments and my relationship with the participants of this study, as potential sources of bias. The epistemological stance I am pursuing for my research is pragmatism (Denzin & Lincoln, 2011). Pragmatism leads us to believe that no methodological approach is better than the other in the generation of knowledge. We must evaluate the findings of our research in terms on how suitable our methodological tools are for the purpose of answering our research questions (Denzin & Lincoln, 2011). Hence, I am using both quantitative and qualitative methods as they provide different perspectives on the phenomenon under study (Frega, 2011).

Using both quantitative and qualitative methods is particularly important for studying a complicated phenomenon like emotions, with multiple components (Afzal & Robinson, 2015; Pekrun & Linnenbrink-Garcia, 2014). To study emotions in educational context, Pekrun and colleagues call for using multi-methods approach by adapting both quantitative and qualitative methods in flexible ways (Pekrun & Linnenbrink-Garcia, 2014). In this context, I am using different forms of data: self-reports, observations, physiological biomarkers, and interviews, to serve two purposes. First, various types of evidence may help understand the different components of emotion (affective, motivational, cognitive, expressive, and physiological). Second, multiple sources of data will serve to promote validity, specifically procedural validity for triangulation purposes (Walther et al., 2017; Walther, Sochacka, & Kellam, 2013). It is important to clarify that this is not a mixed-method study because one stage of the study is not informing the other stage. Rather, this is a multi-modal study (Afzal & Robinson, 2015; Creswell & Clark, 2011; Pekrun & Linnenbrink-Garcia, 2014a), with a primary focus on qualitative findings.

I have not been involved in teaching or designing the programming course that I have studied, and my only role is that of a researcher. However, I have nearly two decades of experience, studying and teaching computer science, which requires extensive programming. I have encountered scores of peers and students who have faced difficulty in learning programming, and have performed inadequately, or have avoided programming during their educational or professional careers. My motivation to design and conduct this study stems from my experience, where I have regularly questioned peoples' negative emotions towards programming. Hence, I feel I have an interest that could influence the study through my own biases. To minimize any personal influence of my own experience in the research study, I have written memos during data collection. The memo writing helped me reflect on my engagement with the data and helped me analyze these data in a more neutral way (Tufford & Newman, 2012). Furthermore, I did extensive journaling during the data analysis stage. This journaling subsequently helped me write about my process in an in-depth and transparent manner.

Context

Purdue University is a large university in the midwestern United States. It is well-known for engineering and mostly has full-time, residential, and traditional aged undergraduate students. The First-Year Engineering (FYE) program at Purdue University serves the incoming first year undergraduate engineering students, approximately 2500 students every year. In 2018, the FYE program had an enrollment of 27% women and 9% international students. The FYE curriculum introduces students to the fundamentals of engineering, and to help them select an engineering major. The FYE students take fundamental STEM and communications courses during the first year of their undergraduate studies. Among these courses, the sequence of two courses ENGR 131 and 132 (Transforming Ideas to Innovation I and II) introduces students to engineering design, data visualizations, computer programming, communications, and teamwork. ENGR 132 focuses specifically on introducing programming skills using MATLAB to FYE students, and hence, is the focus of this study.

ENGR 132 is offered year-round at Purdue and the student population in each semester is different. During the Summer and Fall semesters, transfer students and repeaters typically take this course, while in the Spring semester, new undergraduate students take this course. Students taking ENGR 132 may take other programming courses in parallel, depending on which engineering major they want to pursue. Typically, students intending to major in computer engineering or mechanical engineering take CS 159, Programming Application for Engineers, which is offered by the Department of Computer Science. ENGR 132 is a two-credit course (two 110-minute sessions per week), which is taught using active, blended, and project-based learning methodologies. Students attend lectures and work on problems in class using pair programming. Outside the class, they watch online modules, and work on projects with their teams. Each section of ENGR 132 has up to 120 students. The syllabus for ENGR 132 for Spring 2018 is attached in Appendix A.

Selection Criteria

I used purposive criterion sampling to recruit participants for this study (Patton, 2014). Purposive sampling ensures theoretical validity by capturing the diversity and multiple perspectives of students (Lincoln & Guba, 1985; Walther et al., 2017, 2013). The selection criteria for recruiting participants were defined by first considering students' expertise in programming. It is a common understanding that learning to program is a hard task for novices (Boulay, 1986; Guzdial, 2015; Robins et al., 2003; Soloway, Bonar, & Ehrlich, 1983; Wiedenbeck, Labelle, & Kain, 2004), because of which novices may experience a range of emotions while learning programming in an introductory programming class (Sun & Sun, 2011).

For this study, I considered novices, who took ENGR 132 for the first time in Spring 2018. In the context of this study, novices are students who have not had any programming experience before taking ENGR 132, and they were not be taking another programming course in parallel. My selection criteria excluded two categories of students. I excluded students repeating the course, because they have had ample exposure to programming. I also excluded international students because there are differences in how emotions are experienced across nations and cultures. These differences are due to many factors, for example, language and the ways in which emotions are expressed across cultures (Elfenbein & Ambady, 2002; Jack et al., 2012).

Recruitment

For selecting sample size in qualitative research, there are no concrete rules. However, a common practice is to select enough participants, such that data saturation is reached (Marshall, Cardon, Poddar, & Fontenot, 2013; Morse, 2000). Following this common practice, I decided to recruit around 20 participants for this study.

After approval from the human subject's protection office, and conducting a pilot study, I started recruiting participants (details of the pilot study are mentioned in later sections). I contacted the FYE administration to help me with recruitment. On March 1, 2018, the FYE administration sent the study's recruitment email (Appendix B) to five sections of ENGR 132 (almost 528 students). This email contained a URL to the background survey (Appendix C). I accepted responses for five days. Of these students, twenty-seven students filled the background survey. Of the twenty-seven responses, eleven students met the selection criteria. I sent an invitation to participate in the study to these eleven students.

A sample size of eleven participants did not meet the sample size criteria that I had originally proposed. Hence, I requested the FYE administration to send my recruitment email to two more sections of ENGR 132 on March 5, 2018. The FYE administration sent the recruitment email to two additional sections of ENGR 132 (almost 240 students). I accepted responses for five more days. Of these students, eight students responded, out of which four students met the selection criteria. I sent an invitation to participate in the study to these four students.

After sending two recruitment emails to seven sections of ENGR 132, I was able to recruit only fourteen students. Additionally, most of the study participants were female, and there was a shortage of male participants in my study. I then adopted a snowball sampling strategy (Patton, 2014) and requested a few participants to spread the word for the research study among their male peers. Following the IRB guidelines, I did not contact any students directly for recruitment purposes. During this round of recruitment, five students filled the background survey, out of which three met the selection criteria. I sent an invitation to participate in the study to these three students.

Participants

I recruited 18 participants who were enrolled in ENGR 132 during Spring 2018. According to the information from the background survey, all students who were sent an invitation to participate in the study met the selection criteria. However, one student mentioned during the follow-up interview that she had taken CS 159 (Programming Application for Engineers) during Fall 2017. Hence, she was removed from subsequent data analysis. All participants were given a $40 Amazon gift card as compensation for participating in the study.

Overall there were 10 female and 7 male participants whose data was used in the study. The ages of the participants ranged from 18 to 19 years. The predominant ethnicity was white (11 participants), followed by Asian (3 participants), Latinx (2 participants), African American (1 participants), and American Indian or Alaskan Native (1 participant). Participants' demographic information is provided in Table 1. All the students who were sent an invitation to participate in the study showed up for the study, and I did not experience any no-shows. All participants were assigned pseudonyms for confidentiality purposes.

The data collection took place from Monday March 19, 2018 to Monday April 4, 2018, immediately after the Spring break. Students had learned complex loops (nested loops) in class before the spring break. It is noteworthy that the second major exam for ENGR 132 took place during the data collection time period (March 26, 2018). Hence, some students may have been motivated to participate in this study to practice for the exam.

Table 1: Self-Reported Demographic Information of the Study Participants

Pseudonym	Age	Intended Major	Gender	Race/Ethnicity
Andrew	19	Chemical Engineering	Male	Black or African American
Anna	19	Biomedical Engineering	Female	White
Becky	19	Engineering	Female	White
Christina	18	Chemical Engineering or Electrical Engineering	Female	White
Ella	19	Chemical Engineering	Female	Asian
Emily	18	Industrial Engineering	Female	White
Erica	19	Biomedical Engineering	Female	White
George	19	Mechanical Engineering	Male	American Indian or Alaskan native
Harris	19	Biomedical Engineering	Male	Asian
Jack	18	Chemical Engineering	Male	Hispanic or Latinx
John	18	Civil Engineering	Male	White
Lilly	18	Multidisciplinary Theatre Engineering	Female	White
Mark	19	Mechanical Engineering	Male	White
Martha	19	Biomedical Engineering	Female	White
Rachel	19	Materials Engineering	Female	White
Randy	18	Chemical Engineering	Male	Hispanic or Latinx
Sarah	18	Material Engineering	Female	Asian

Research Design

Figure 4 illustrates the detailed research design of the study, with each step explained in subsequent sections. I first describe the experimental setup and software used for my pilot studies and data collection. I then discuss the types of data I collected, the steps involved in data collection, and the data analysis strategies. Finally, I discuss the decisions I took to ensure the quality of my research study.

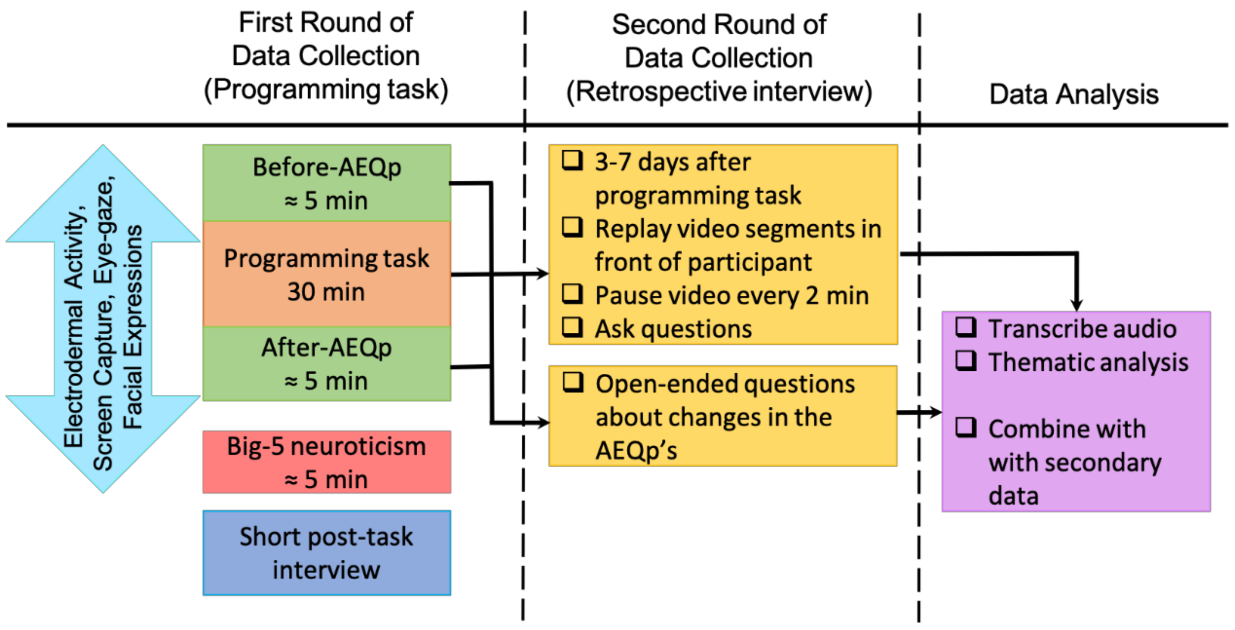

First Round of Data Collection (Programming task)
Second Round of Data Collection (Retrospective interview)
Data Analysis
Electrodermal Activity, Screen Capture, Eye-gaze, Facial Expressions
Before-AEQp ≈ 5 min
Programming task 30 min
After-AEQp ≈ 5 min
Big-5 neuroticism ≈ 5 min
Short post-task interview
❑ 3-7 days after programming task
❑ Replay video segments in front of participant
❑ Pause video every 2 min
❑ Ask questions
❑ Open-ended questions about changes in the AEQp's
❑ Transcribe audio
❑ Thematic analysis
❑ Combine with with secondary data

Figure 4. Research Design of the Study

Experimental Setup

For data collection I used iMotions, a software platform that lets researchers integrate and synchronize different data sources (e.g., biomarkers, surveys, screen capture) in one place. There are two sides of the iMotions experimental setup (see Figure 5). The first side is for the *participant being tested* which includes the workstation consisting of a computer (monitor, keyboard, and mouse) with a frontal camera and eye tracker installed under the monitor. The second side is a *live view for the researcher* which enables the researcher to view the participant's activity in real-time and make necessary changes, if needed (e.g., prompted the participant if their face was out of the range of the camera). Students were also provided with scratch paper and pencil, in case they needed it for scratch work. However, none of the participants used the scratch paper, and hence no scratch papers were collected after the programming task ended.

The experimental setup is shown in Figure 6. The workstation was equipped with a webcam to capture facial expressions. The camera placement and participants' positioning are also crucial for accurate capture of facial expressions. In this case, the participant actively used the keyboard and mouse, and periodically looked down. Hence, the best placement of the camera was underneath the monitor (iMotions, 2016). The participant was also seated comfortably in front of the camera so that their face was in the center of the frame.

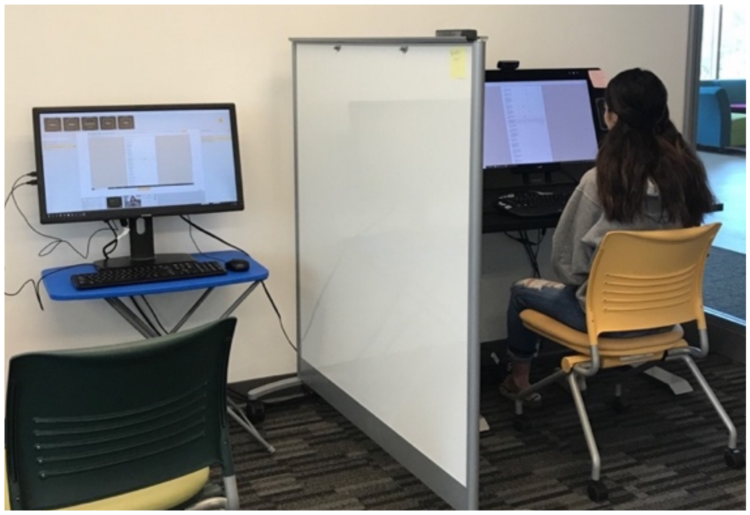


Figure 5. The Workstation and The Researcher Station

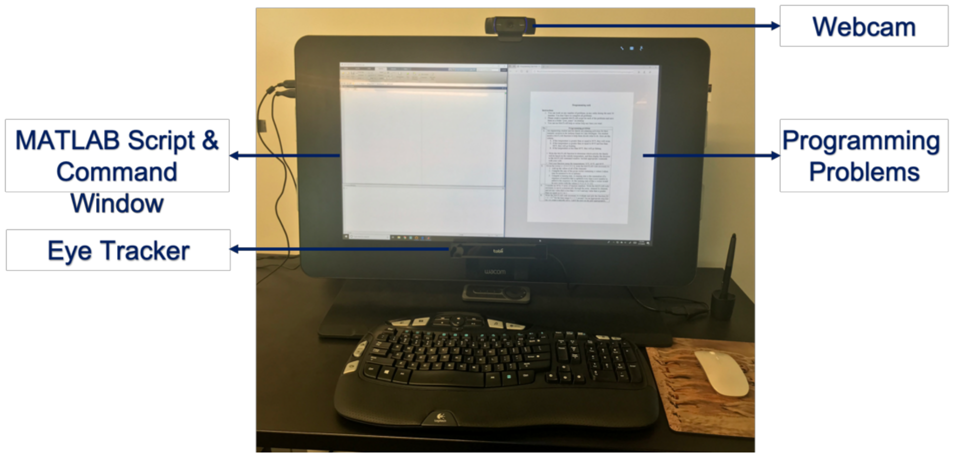


Figure 6. Setup of the Experiment

Students' eye-gaze scan path was captured using an eye tracker. The eye tracker was installed on the base of the monitor so that students' eyes were in the range of the eye tracker. For eye tracking to work accurately, the eye tracker calibrated the eye-gaze before the programming task. For eye-gaze calibration, students followed a moving point on the screen. iMotions prompted the participant to repeat the process of calibration if it was not done correctly the first time.

Types of Data

I collected different types of data both concurrently and retrospectively. For research on emotions, simultaneously collecting different kinds of data increases the validity and accuracy of the study (Afzal & Robinson, 2015). The evidence I collected is both quantitative and qualitative, while the data analysis is primarily qualitative, with some supplemental statistical analysis. I collected data in two rounds: 1) programming task and 2) retrospective think-aloud interview. This section explains the types of data collected during both stages of the study.

Before- and After-AEQp

The Before-AEQp and After-AEQp were adapted from the sub-section of the Achievement Emotions Questionnaire that assesses test-related emotions (Pekrun, Goetz, Frenzel, Barchfeld, & Perry, 2011; Pekrun et al., 2005), where the p in AEQp refers to programming. The AEQp categorizes emotions into four categories, based on the valence and activation of the emotion (Pekrun et al., 2011; Pekrun et al., 2005; Pekrun & Perry, 2014). These four categories are positive activating (enjoyment, hope, and pride), positive deactivating (relief), negative activating (anger, anxiety, and shame), and negative deactivating (hopelessness). The Before-AEQp assessed enjoyment, hope, pride, anger, anxiety, shame, and hopelessness. The After-AEQp assessed enjoyment, pride, relief, anger, and shame. Hope, anxiety and hopelessness are prospective emotions, because of which they are assessed only in the Before-AEQ (Pekrun et al., 2011). Similarly, relief is a retrospective emotion and is assessed only in the After-AEQ (Pekrun et al., 2011). All the other emotions are assessed in both Before- and After-AEQp surveys.

I rephrased all items about tests on the questionnaire because a programming task is not a test per se. Moreover, I rephrased items which were not relevant to programming tasks. For instance, "When I get the results of the test, my heart beats with pride" was rephrased to "When my program ran correctly, my heart beats with pride." Since this questionnaire was not intended to collect massive amounts of data, I did not validate the adapted version using statistical methods. However, I discussed the phrasing of items on the modified questionnaire with my pilot study participants and with peers having prior programming experience, to check its face validity (Patton, 2014). The adapted version of Before-AEQp and After- AEQp is available in Appendix D. I calculated the average of students' responses on each sub-scale of the questionnaire. For example, the Task Enjoyment subscale has five items, and the score for Task Enjoyment is the average of the responses to those five items. I used participant responses on the Before- and After-AEQp as prompts for the retrospective think-aloud interview.

I asked the participants about the emotions they had anticipated before the programming task and their perceived emotions after the programming task ended. Participants were asked questions based on the situations they had experienced (Endler & Okada, 1975). For instance, a participant may experience anxiety before starting a task. Hence, the Before-AEQp asked students about anxiety related questions. However, the After-AEQp did not ask them anything about anxiety as anxiety typically dissipates once the associated situation ends.

Electrodermal Activity

The observation of electrodermal activity (EDA) requires non-invasive equipment (shimmer device[1]) which has two electrodes that apply a constant low voltage to measure the electrical current between the electrodes. Researchers can place the electrodes on the fingers, palm, or feet of a participant. Researchers typically put the electrodes on the palms because this configuration captures distinguished EDA (Boucsein, 2012). In the current study, the participants had to use their hands to type, and since EDA data is sensitive to movement, I placed the electrodes on the participant's foot (see Figure 7), which is another site where EDA can be collected accurately (Boucsein, 2012).

The placement of the electrodes is not the only prerequisite for accurate collection of EDA. The participant should also be seated in a comfortable position with minimal movement (especially the limbs where electrodes are attached), should be able to breathe normally without irregularities, and should avoid talking (Boucsein, 2012; iMotions, 2016). There are some other confounding factors for EDA data collection, which need to be accounted for while collecting EDA. For instance, EDA may be affected by medical history of heart conditions, caffeine and drug consumption, traumatic brain injury, to name a few (Boucsein, 2012; Villanueva et al., 2018). However, I did not consider these pre-requisites for two reasons: to keep the study design simple and to prevent the sample from reducing further. That is, using these parameters as exclusion criteria would have likely reduced the sample size of the study further.

[1] http://www.shimmersensing.com/

Post-task Interview

After the completion of the programming task, I conducted a short fifteen-minute interview with each participant, asking questions about their overall experience, the challenges they faced, and how they acted to overcome those challenges. This interview protocol was semi-structured, was audio recorded, and transcribed verbatim for subsequent data analysis. Appendix E has the protocol for the post-task interview.

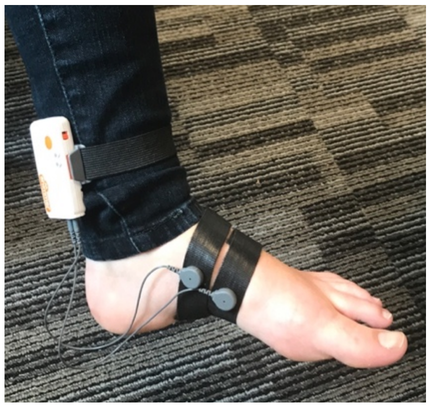


Figure 7. Shimmer Device Attached to Participant's Foot

Retrospective Think-Aloud Interview

During the second stage of the study, which took place 3 to 7 days after the programming task, the participants came back for a retrospective think-aloud interview. I had analyzed participants' responses on the Before- and After-AEQp before they arrived for the retrospective interview.

The primary reason for conducting this interview was to collect data that answered the research questions of the study. This interview helped students reflect on the actions they took during the programming task and verbalize their thought process, the reasons for experiencing certain emotions, and help reveal the self-regulation strategies students were using during the programming task. During this interview student responses on the Before- and After-task AEQp were used to identify changes in student emotions between the beginning and the end of the programming task. Appendix F shows the protocol for the retrospective think-aloud interview.

A retrospective think-aloud interview is a verbalization technique in which the participants verbalize their thoughts while watching a recording of their performance on a task that they had previously performed (Elling, Lentz, & de Jong, 2011; Haak, Jong, & Schellens, 2003). For the current study, it was suitable to do a retrospective think-aloud interview instead of a concurrent think-aloud interview for two reasons. The programming tasks required full concentration of the participant, hence, concurrently verbalizing their thoughts may have distracted them. Moreover, this research aims to assess emotions; a distraction from simultaneously working on programming problems and verbalizing thoughts may have influenced their emotions.

Big-5 Neuroticism Subscale

I used the Neuroticism sub-scale of the Big-Five personality inventory to assess students inherent personality trait as it relates to neuroticism. The big-five personality inventory organizes personality based on five dimensions: Extraversion, Agreeableness, Conscientiousness, Neuroticism, and Openness to Experience. Research suggests that these five personality types are applicable across observers and cultures and are stable over time (McCrae & John, 1992). This sub-scale was administered at the end of the programming task so that students' responses did not influence their response on the Before- and After-AEQp and the programming task. Appendix G shows the eight items of the neuroticism sub-scale.

Secondary Data

One drawback of a retrospective think-aloud interview is that the participants may not retain a strong memory of their actions during the programming task. To overcome this drawback and to aid participants' memory, I collected different types of secondary data, to help students recall their experience on the programming task (Paulhus & Vazire, 2009). During the programming task, I continuously recorded three types of data that I replayed during the retrospective interview. First, I used iMotions to record everything on the participant's screen during the task. Second, I used an eye tracker to record the location on the screen at which the participant was gazing. Third, I used a webcam to record the participant's facial expressions.

Programming Task

The programming task consisted of 4 programming problems in MATLAB. I carefully selected these problems by considering students' skill level and types of problems students solve in ENGR 132. I acquired a pool of questions (practice questions and exams) from the FYE administration. From this pool of questions, I selected a few problems. I discussed these problems with an instructor who had taught and designed the ENGR 132 course for many years. Based on her experience, she provided an estimate of the difficulty level of these problems, especially for novice students taking ENGR 132 for the first time. I considered the difficulty level of the problems so that most students could solve these problems during a 30-minute period. Moreover, these problems were selected because they cover a range of programming constructs that students had already studied in their ENGR 132 class. One difference between the programming task and ENGR 132 is that for ENGR 132, students are provided with code templates, but in the programming task, I did not provide them with the template. Since, I do not have experience teaching ENGR 132, I was not aware that students are provided with code templates. Moreover, code templates were not mentioned during my discussion with the instructor. Nevertheless, since data were collected during the last quarter of the semester, students had had ample practice because of which they did not need code templates for the programming task.

I discussed these problems with participants of the second pilot study and asked them about the difficulty of the problems. All pilot study students reported that the problems were easy for them. Table 2 shows the four programming problems and the associated learning objectives from ENGR 132. Students had the option to work on the problems in any order and could also switch back and forth between problems. For problem 1, students were asked to comment their code and were also given test cases that they could use to test their code. The screen capture data suggests that almost all students wrote comments for all problems they had attempted. Unless specified, students could use a pre-defined MATLAB function to solve a problem or write code. For example, students could use the built-in cumsum function to compute the running sum or they could write code from scratch. Students saved their code scripts (m files) for every problem in a folder on the Desktop. In ENGR 132, students normally work on assignments in groups and seek help from their peers and from the teaching assistants. However, for this programming task, they used the available resources (MATLAB or online help).

Table 2: Problems on the Programming Task

No	Programming Problem	Learning Objectives
1.	An engineering student and his family are planning activities for their summer vacation to the Indiana Dunes & Lake Michigan. The student needs a MATLAB function to help them decide what to do. Here are the criteria: 1. If the temperature is greater than or equal to 90°F, they will swim. 2. If the temperature is greater than or equal to 80°F and less than 90°F, they will go boating. 3. If the temperature is less than 80°F, they will go fishing. Write the MATLAB function to determine which activity the family will do based on the outside temperature, and then display the decision to the MATLAB command window. Include appropriate comments with your code. Test your function using the temperatures 70°F, 85°F, and 90°F.	LO 1.1. Construct an appropriate function definition line LO 1.2. Code a selection structure LO 1.3. Construct relational and logical statements from English statements LO 1.4. Execute a user-defined function
2.	Given the vector *x* = [1 8 3 9 0 1], write the MATLAB code necessary to: 1. Add up the values of all of the elements 2. Compute the sine of the given vector containing x-values (values may be assumed to be in radians) 3. Compute a running sum. A running sum is the summation of a sequence of numbers that is updated every time a new number is added to the sequence. So, the running sum of this x vector would be new vector with the values [1 9 12 21 21 22].	LO 2.1. Use built-in functions to perform algebraic and trigonometric calculations LO 2.2. Code a definite looping structure that employs vector indexing (if cumsum function not used) LO 2.3. Execute a script from the MATLAB Command Window
3.	Consider an *M*-by-*N* array of random numbers. Write the MATLAB code necessary to move systematically through the array, element by element, and set any value that is less than 0.2 to 0 and any value that is greater than (or equal to) 0.2 to 1.	LO 3.1. Code nested structures LO 3.2. Replace elements of arrays LO 3.3. Execute a script from the MATLAB Command Window
4.	Write the MATLAB code necessary to evaluate and plot the function $f(t) = 4 - t^2e^{-3t}$ for the time range $0 \leq t \leq 3$ second. Use an appropriate step size for *t* to create a smooth curve. Label the axes on the plot appropriately.	LO 4.1. Use built-in functions to perform algebraic and trigonometric calculations LO 4.2. Create an x-y plot from a single data set LO 4.3. Format plots for technical presentation LO 4.4. Execute a script from the MATLAB Command Window

I adapted grading rubrics from ENGR 132 to grade students' code scripts. Student grades were used to assess students' performance on the programming task. Students' grades on each of the problem are shown in Table 3. The programming task shown to the students and the adapted grading rubrics are shown in Appendix H. Table 3 also lists approximate time that the students spent working on each problem and the order in which they worked on the problems. This order of numbers also shows if the students went back to work on a previous problem. These data were extracted by viewing the video of the screen capture for each student. Time spent on each problem includes the time it took the student to read the problem, type code, look at help resources (if any), fix and debug code, run code, and save the files for that problem. Table 3 shows the average time spent and average score on each problem (The average includes time and scores on the problems that the students did not attempt or submit).

Table 3: Time Spent on Each Problem, Grade, and Order of Working on the Problem

Pseudonym	Problem **1**		Problem **2**		Problem **3**		Problem **4**		**Order in which the problems were attempted**
	Time (min)	**Score (16)**	**Time (min)**	**Score (12)**	**Time (min)**	**Score (12)**	**Time (min)**	**Score (16)**	
Andrew	12	12	17	8	2	0	0*	0*	1→2→3
Anna	16	16	10	12	0*	0*	4	0	1→2→4
Becky	11	16	9	7	8	0	2	5	1→2→3→4
Christina	11	15	8	8	4	0	7	3	1→2→3→4
Ella	8	16	7	12	6	7	9	16	1→2→4→3
Emily	11	16	7	10	5	11	7	8	1→2→3→4→3→2→4
Erica	10	16	14	10	6	9	0*	0*	1→2→1→2→3
George	16	0*	0*	0*	0*	0*	14	0+	1→4
Harris	9	16	4	12	6	0	11	12	1→2→3→4→3→2→4
Jack	8	16	13	11	8	0	1	12	1→2→3→2→4→2→3→2→3
John	14	16	5	12	2	0	9	16	1→2→4→3
Lilly	8	16	12	10	3	0	7	16	1→2→4→3→2→3
Mark	13	10	0*	0*	16	0+	1	0+	1→3→4
Martha	15	16	8	8	7	0	0*	0*	1→2→1→2→3
Rachel	14	16	6	8	3	0	7	16	1→2→3→4→1→3
Randy	12	12	7	10	3	0	8	8	1→2→3→4
Sarah	27	16	3	8	0*	0*	0*	0*	1→2
Average	**12.6**	**14.1**	**7.6**	**8.5**	**4.6**	**1.6**	**5.1**	**6.5**	

Not Attempted (*), File not Saved (+)

Data Collection

First Round of Data Collection: Programming Task

The following sections describe the structure of the programming task and the data collected during the programming task.

I welcomed the participant at the reception of WANG hall, introduced myself, and brought them to the room where the experiment was set up. I provided a detailed overview of the study to the participant and then handed them the consent form (see Appendix I). Once they had read and signed the consent form, I connected a noninvasive device on one foot (shimmer device) (see Figure 7).

Before the task started, iMotions performed calibration between the participant's eye-gaze and the eye-tracker. The participant's baseline EDA was then collected. The participant was then prompted to fill out the Before-AEQp, in which they self-reported the emotions they thought they would experience during the programming task. Once they completed the AEQp, the programming task started, and the participant was given 30 minutes to work on the problems. After completing the task, the participant was prompted to fill out the After-AEQp, in which they self-reported the emotions they thought they experienced while working on the problems. After completing the After-AEQp, participants filled the Neuroticism sub-scale of the Big-Five personality test. Finally, I conducted a short post-task interview during which I asked questions about students' experience with the programming task. Figure 4 shows a block diagram explaining this process. Appendix J contains the protocol for the programming task and the short post-task interview.

Second Round of Data Collection: Retrospective Think-aloud Interview

I conducted the retrospective think-aloud interview in two stages. During the first stage, I replayed the video of participant's programming task for two minutes intervals, during which participants could verbalize their thoughts. After two minutes, I paused the video and asked these questions: a) What emotion do you think you were experiencing here? b) What do you think are some reasons for experiencing these emotions? and c) What strategies did you adopt to deal with the emotion (How did you deal with this emotion)?

I provided participants with a sheet of paper with a list of emotions derived from Pekrun's taxonomy of academic emotions (Pekrun & Perry, 2014). The list of emotions was provided to help students assign an appropriate label to the emotion they thought they experienced during a certain time during the programming task (Bosch & D'Mello, 2015). The emotions on the list are driven by theory and has a range of both positive and negative emotions that students frequently experience in an academic setting (Pekrun & Perry, 2014). Students were also given the option to articulate an emotion not mentioned in the list and to provide a definition for that emotion. There are many instances where students reported feeling neutral, which means that they are not feeling any type of positive or negative emotions (Bosch & D'Mello, 2015). The list of emotions is shown in Table 4.

Table 4: List of Emotions

Enjoyment	Anger	Sadness
Hope	Anxiety	Disappointment
Joy	Shame	Contentment
Pride	Boredom	Relaxation
Relief	Frustration	Neutral
Gratitude	Hopelessness	Others (Please define this emotion)

During the second stage of the retrospective think-aloud interview, I showed participants their responses to the Before- and After- AEQp, placed side by side. I asked them why they experienced a change in each emotion between the beginning and the end of the programming task. Figure 8 shows the image of the retrospective think-aloud interview with one student.

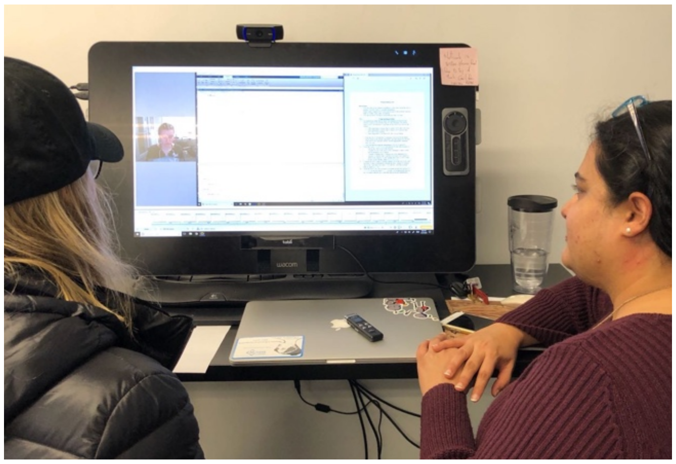


Figure 8. Retrospective Think-Aloud Interview with One Participant

Data Analysis

I used different qualitative and quantitative analyses methods to answer the research questions. I used thematic analysis to analyze the primary source of data, that is, the retrospective think-aloud interview. I used quantitative analyses procedures for other sources of data (e.g., AEQp responses and EDA). I describe each of these methods in the following sections.

Thematic Analysis

I analyzed the retrospective think-aloud interviews using a mix of deductive and inductive thematic analysis (Aronson, 1995; Braun & Clarke, 2006; Patton, 2014). The deductive thematic analysis was driven by the theoretical framework and literature that guided the study (Braun & Clarke, 2006), whereas in inductive thematic analysis, themes emerged from the data (Patton, 2014). The steps for thematic analysis are shown in Table 5 and are explained in the subsequent sections.

Table 5. Different Stages of Thematic Analysis
Adapted from (Braun & Clarke, 2006)

Steps	Description
Getting familiar with the data	• Transcribed the audio files • Read and re-read transcripts to make sense of these data
Development and reliability of the codebook	• Generated initial codebook using theory, literature, and data from pilot studies • Established reliability with the help of another researcher
Assigning or generating initial codes	• Assigned codes using the codebook • Generated a new code if it did not exist in the codebook, and added the code to the codebook
Categorization	• Used theoretical framework and literature to categorize codes
Generating themes	• Collated codes into potential themes, based on similar characteristics
Descriptive report and writing memos	• Selected compelling excerpts • Summarized and interpreted the excerpts using theory and literature • Wrote memos during data analysis

Getting Familiar with the Data

I transcribed the audio for 5 retrospective think-aloud interviews. I sent the rest of the audio recordings to a transcription service. These transcripts were then transferred to a software for qualitative data analysis (NVivo). Although I collected these data, I read all transcripts to familiarize myself with the data. It is essential for the researcher to immerse themselves in the data to check the accuracy of the transcripts and to familiarize themselves with the depth and breadth of these data (Braun & Clarke, 2006).

Development and Reliability of the Codebook

I developed the initial codebook by using the two theoretical frameworks that guided the different stages of the study (Pekrun & Linnenbrink-Garcia, 2014; Zimmerman, 2002). After the initial development, I revised the codebook using the data collected from the pilot studies conducted in July 2017 and January 2018. Hence, the codebook kept evolving as the analysis progressed. To improve the reliability of the coding process, I requested a fellow researcher to code four transcripts with me (referred subsequently as the secondary researcher). The secondary researcher is a Ph.D. student in engineering education with previous educational and teaching experience in computer science. Her previous experience was crucial in helping her understand these data. We used the following procedure to improve the reliability of the coding process.

As the first step, the secondary researcher and I devised the following strategies for coding the data. First, we decided to code as close to the data as possible, by selecting appropriate labels for the codes. Second, we decided to choose as much of the excerpt as possible. This step was essential to make sure that we went back to the raw data only as needed and could understand the excerpt by looking at the context (Braun & Clarke, 2006). Third, we also jointly decided to add all new codes to the codebook. Since I used a hybrid approach of thematic analysis, it was appropriate to add new codes as they emerged from these data.

After coding one transcript independently, both researchers came together to consolidate our respective codes. For code consolidation, we used the following process. First, we exported our codebooks from NVivo and printed them. Since I would be doing all subsequent coding, my codebook was considered as the master codebook (MC). Second, we compared similar codes in both codebooks. If both of us had used the same code for an excerpt, we marked that code in the

MC. If the codes were similar but with different labels, we discussed the codes, picked the most appropriate label for the code, and checked it off in the MC. Third, when the code appeared only in the secondary researcher's codebook, we first determined if the code should be included in the MC. If yes, we decided an appropriate label for the code and added it to MC. Otherwise, the code was discarded. Fourth, we discussed the codes that we had placed in different categories. These codes were moved to an appropriate category in the MC based on how the theory explained the construct or by mutual discussion. Finally, the secondary researcher and I coded three other transcripts independently and repeated all the steps of consolidation.

Assigning or Generating Initial Codes

Once the codebook was refined using codes from the first four transcripts, I assigned codes to the remaining thirteen transcripts independently. I followed the procedure devised in the previous section for assigning codes. That is, if the code was available in the codebook, I assigned it to the relevant excerpt; otherwise, I generated a new code, and then assigned it to an excerpt. However, it must be noted that after the development of the initial codebook, data saturation was reached and very few new codes were generated.

Categorization

For Research Question 1, I categorized my findings based on previous literature about the step's students take to solve programming problems (Kinnunen & Simon, 2010c, 2012a), and the control-value theory of achievement emotions (Pekrun & Perry, 2014). Kinnunen and Simon developed a process model that describes how computing majors experience the process of doing programming assignments in an introductory programming course (Kinnunen & Simon, 2010a, 2010c, 2011, 2012a). This process model has six stages: 1) Getting Started, 2) Encountering Difficulties, 3) Dealing with Difficulties, 4) Succeeding, 5) Submitting, and 6) Stopping.

However, students in the current study were not required to submit their MATLAB scripts, so there is no Submitting stage in the adapted process model for this study. Moreover, data from the current research captures student behaviors and emotions while they typed code. Hence, the adapted process model has an additional stage called "Typing Code". The six stages of the adapted model are 1) Getting Started, 2) Typing Code, 3) Encountering Difficulties, 4) Dealing with Difficulties, 5) Succeeding, and 6) Stopping.

Figure 9 shows the adapted process model. This adapted process model and the themes within each category will be discussed in depth in the next chapter.

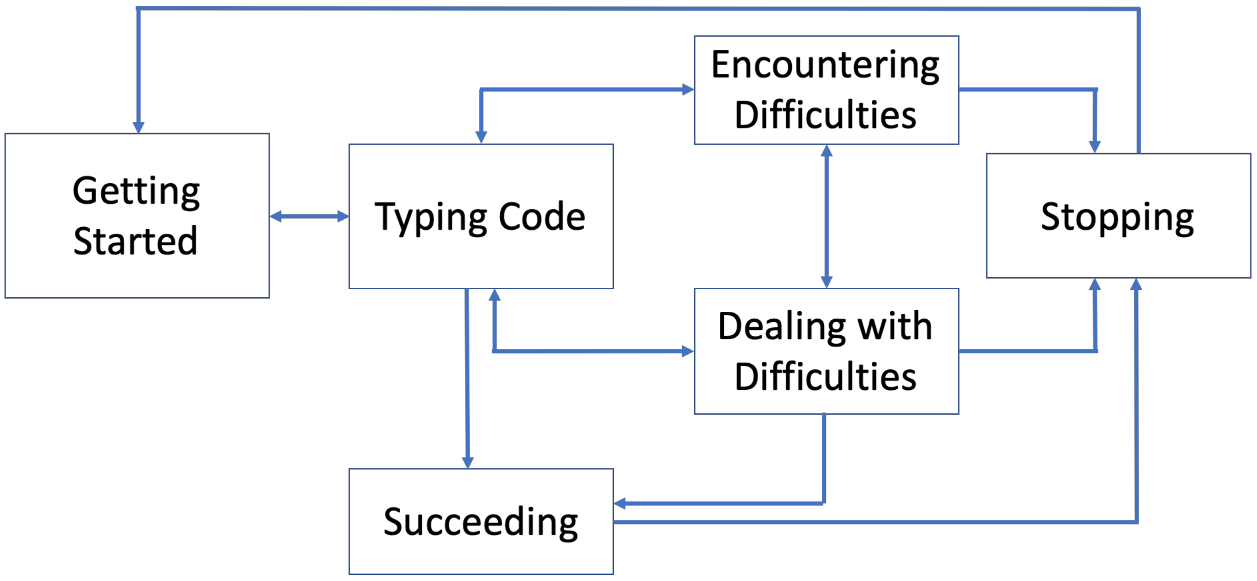


Figure 9. Steps Students take to Solve Programming Problems

For Research Question 2, I used the idea of change in emotions to devise categories. That is, an emotion could either increase or decrease between the beginning and end of the programming task, or there was no change in emotion reported. For Research Question 3, I used the self-regulated learning framework to categorize the findings. This process of categorization for all research questions was not done in NVivo. Instead, I extracted all the codes from NVivo, printed them, placed each code on post-it notes, and then placed each code in the relevant category. As an example, Figure 10 shows the categorization process for Research Question 1.

Generating Themes

Once I had categorized codes for each research question, I further categorized codes within each category into sub-categories (or themes), based on similar characteristics. This process was inductive in nature, where the themes emerged from the codes (Braun & Clarke, 2006). There were some codes which were applied to the same excerpts. These codes were merged together. Some codes were also placed in a wrong category; hence, they were moved to the appropriate category. For instance, the code “Did not understand the instructions for the problem” was initially placed in the “Getting Started” category. However, while generating themes and reading the excerpts again, I realized this code belonged to the “Encountering Difficulties” stage, hence the code was moved to this category. The final codebook is in Appendix K.

Descriptive Report and Writing Memos

This step included writing the findings in a concise and organized manner. For this purpose, I selected excerpts that captured the essence of each theme and category. In addition to selecting relevant excerpts, I wrote memos for each part of the data analysis. These memos subsequently helped me in writing a compelling argument to answer each research question. Sample memos during data collection and analysis are shown in Appendix J.

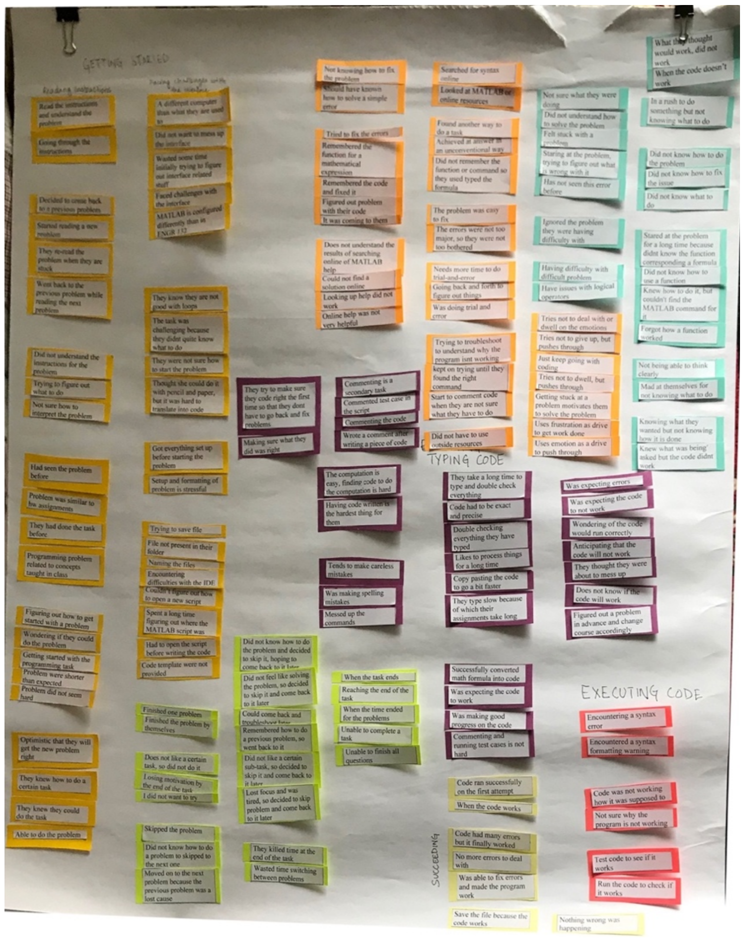


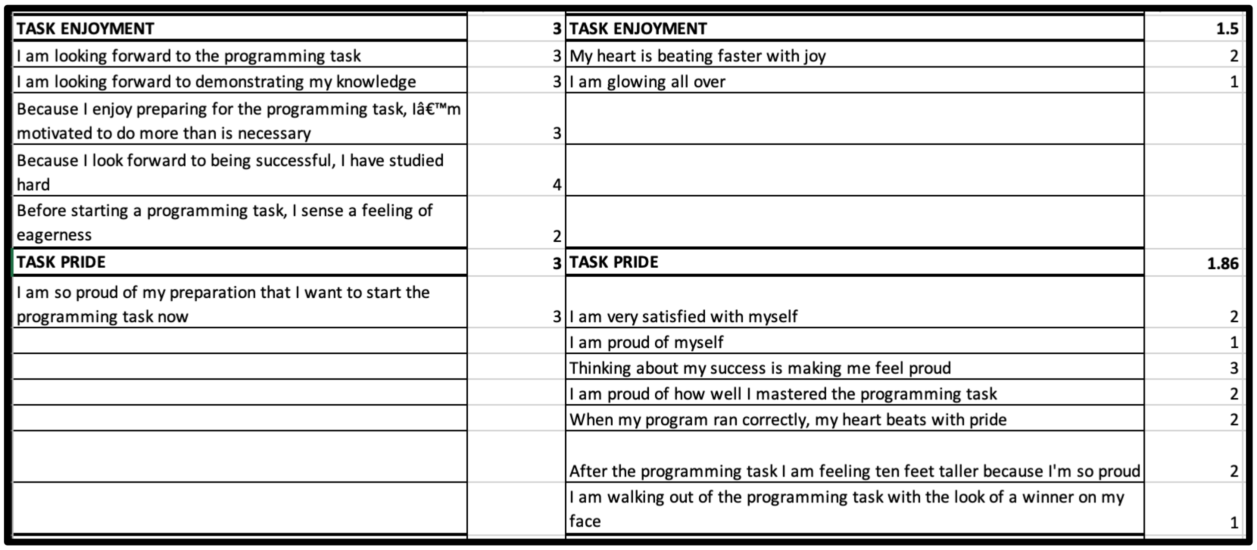
Figure 10. Categorization Process for Research Question 1 Using the Process Model

Quantitative Analyses

The quantitative data collected from this study included Before- and After-AEQp, the neuroticism data, and EDA data. The Before- and After-AEQp data were used as prompts to generate interview questions during the retrospective think-aloud interview (see Figure 11). The EDA signal consists of two parts: tonic and phasic responses.

The phasic part of the EDA signal refers to a short, fast emotional or cognitive response (Boucsein, 2012), hence, phasic EDA was used for the analysis. To account for interindividual differences in the EDA, range correction was calculated, which normalizes the phasic EDA between 0 – 1. The range correction was done using the following formula (Boucsein, 2012; Villanueva et al., 2018).

$$Range - Corrected\ EDA = \frac{|EDAi - EDAmax|}{|EDAmax - EDAmin|}$$



TASK ENJOYMENT	**3**	**TASK ENJOYMENT**	**1.5**
I am looking forward to the programming task	3	My heart is beating faster with joy	2
I am looking forward to demonstrating my knowledge	3	I am glowing all over	1
Because I enjoy preparing for the programming task, Iâ€™m motivated to do more than is necessary	3		
Because I look forward to being successful, I have studied hard	4		
Before starting a programming task, I sense a feeling of eagerness	2		
TASK PRIDE	**3**	**TASK PRIDE**	**1.86**
I am so proud of my preparation that I want to start the programming task now	3	I am very satisfied with myself	2
		I am proud of myself	1
		Thinking about my success is making me feel proud	3
		I am proud of how well I mastered the programming task	2
		When my program ran correctly, my heart beats with pride	2
		After the programming task I am feeling ten feet taller because I'm so proud	2
		I am walking out of the programming task with the look of a winner on my face	1

Figure 11. A Snapshot of Collated Before- and After-AEQp Responses from One Participant

Pilot Studies

First Pilot Study

During Summer 2017, I conducted a pilot study with 5 participants enrolled in the summer offering of ENGR 132. For recruiting students, the FYE team helped me contact the instructor who asked me to design one slide with details about the pilot study. The instructor showed the slide to students in her course, and interested students signed up for the study using an online form hosted on Qualtrics. I wanted to recruit students after they had been exposed to most of the course material (up until complex or nested loops). According to the ENGR 132 syllabus, students studied complex loops on July 11 in their class. Hence, the recruitment schedule worked out as planned. I started recruitment during the first week of July. The data collection took place from July 13 to August 4.

My study benefited from this pilot study in three main ways. I refined my data collection protocols, as I was previously not familiar with using iMotions. I streamlined the process by interacting with these students. For instance, for my initial research design, I administered AEQp, three times: once before the programming task, once during, and once after the programming task. I also identified potential problems that may arise during the programming task and found ways to prevent them. For instance, one student was accidentally hitting keys on the keyboard, because of which he was moving on to next stimulus (After-AEQp) before finishing their work on the previous stimulus (first MATLAB session). Finally, I figured out the time estimates for completing the survey. For instance, I had originally allocated 7 minutes for each of the before- and after-AEQp. However, all the students completed each of the questionnaires in less than 5 minutes.
In addition to the abovementioned benefits, I also faced challenges collecting EDA data. I did not have any prior experience with EDA data and hence this pilot study helped me understand ideal EDA collection methods. I first collected EDA with electrodes on the fingers — see Figure 12 (a) — but this proved to be uncomfortable for the participant. So, I moved the EDA measuring device to the wrist — see Figure 12 (b). This caused problems because EDA data is sensitive to movement (Boucsein, 2012). Another suitable body location for collecting EDA data is the foot of the participant (see Figure 7). This site gave the best results, not just in terms of more pronounced EDA, but also in terms of minimal movement artifacts. For the final data collection, I used this setup for collecting EDA.

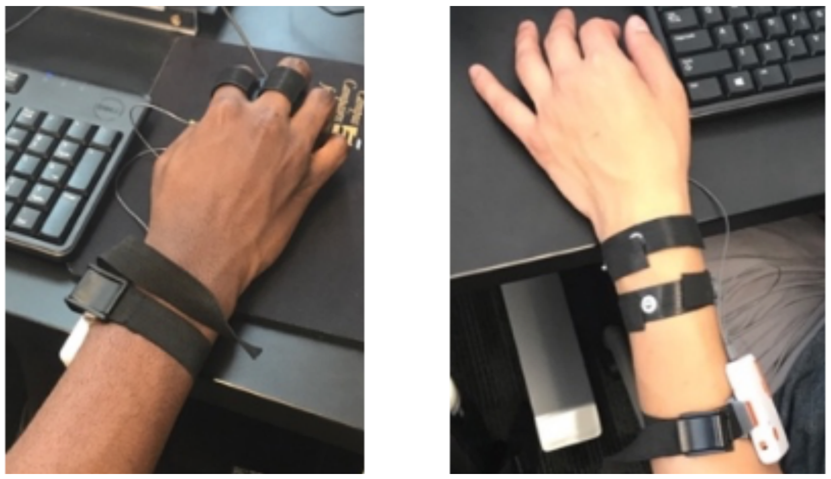


Figure 12. EDA Collected From (a) Fingers and (b) Wrist

IRB Application and Approval

After the successful defense of my dissertation proposal on September 20, 2017, I compiled the IRB application for the study by incorporating improvements suggested by my dissertation committee. The IRB application was submitted to the human research protection program at Purdue University during the mid of October 2017. The IRB application was approved during the first week of January 2018, after incorporating revisions suggested by the IRB. See Appendix I for the IRB approval letter.

Second Pilot Study

After the approval of the IRB application, I conducted a second pilot study using the revised research design. For the purpose of recruitment, the FYE team sent an invitation email to students who had taken ENGR 132 during Fall 2017. In February 2018, I recruited five students for the second pilot study. These students had recently completed ENGR 132 successfully. This pilot study helped me fine-tune the retrospective think-aloud interview. Specifically, during this pilot study, I decided to replay the video for two minutes before pausing and asking students about their emotions. This decision was made by trying out different video replay times with students (one minute, two minutes, and three minutes), and asking students which timeslot best helped them recall the past events. Most of the students thought a two-minute time slot was neither too long, nor too short, to help them recall the events.

Quality of Research

I used the quality framework proposed by Walther et al., (2013) to inform the different methodological decisions I made to ensure the quality of my research study. This framework provides guidelines for "making the data" during the collection phase, and "handling the data" during the analysis phase. Purposive sampling ensures *theoretical validation* by capturing the diversity and multiple perspectives (Walther et al., 2013). I used purposive criterion sampling to recruit participants for this study (Patton, 2014). *Procedural validation* refers to all the procedures that are incorporated in the research design to mitigate the threats to validity (Walther et al., 2013). For this purpose, I used multiple sources of data for triangulation. Furthermore, for handling data, I did not conduct the analysis in isolation. I kept on discussing findings with my professors and peers. I wrote memos during each step of data collection and analysis. *Communicative validation* works to establish interpretation with the internal and external customers of the research (Walther et al., 2013). While analyzing these data, I checked the reliability of the code book with another researcher for communicative validation. *Process reliability* is the means by which the research process is made independent from random influences (Walther et al., 2013). To collect data in a dependable way, I conducted pilot studies to check the appropriateness of the task difficulty, and for the phrasing of the interview questions. During the pilot studies, I also refined the protocol for the interviews. Moreover, I embedded secondary data collection (eye-gaze, facial expressions, and screen capture) to aid student recall during the retrospective think-aloud interview. I also provided students with a list of emotions, so that they could easily identify and label the emotion they thought they had experienced.

CHAPTER 3: FINDINGS

Research Question 1

RQ1a: What emotions do first-year engineering students experience while they work on a computer programming task?

RQ1b: What reasons do students describe for experiencing the different emotions?

The primary data used to answer this research question are the retrospective think-aloud interviews from 17 participants of the study. While students worked on the programming task, they engaged in different activities, which induced a range of positive and negative emotions in students.

In the "Getting Started" stage, students prepared themselves for the task (e.g., by reading the problems and selecting one of them). In the "Typing Code" stage, students typed code for the problem they selected in the "Getting Started" stage. If the code executed successfully, they moved to the "Succeeding" and "Stopping" stages. However, if the code did not execute successfully, students moved to the "Encountering Difficulties" stage. To overcome the difficulties, students adopted different strategies in the "Dealing with Difficulties" stage. In this stage, students encountered two cases. If students successfully solved the errors, they continued working on the problem. However, if students were unable to fix the errors, they decided to move to the next problem. The double-sided arrows between two stages in Figure 13 indicate that students could move back and forth between those stages. For instance, while encountering errors, students could go back and type something to fix the error.

During each of these stages, students experienced a range of positive and negative emotions, which influenced their self-regulation and decision making during the programming task. Table 6 shows the frequency of occurrence of each emotion in these data, and the number of students who reported experiencing that emotion. The predominant emotions were frustration (24%), anxiety (10%), confusion (10%), neutral (9%), and relief (9%). All 17 students reported feeling frustrated, while 15 students reported feeling anxious at various times during the programming task. Additionally, students also reported some emotions that did not appear on the list in Table 4.

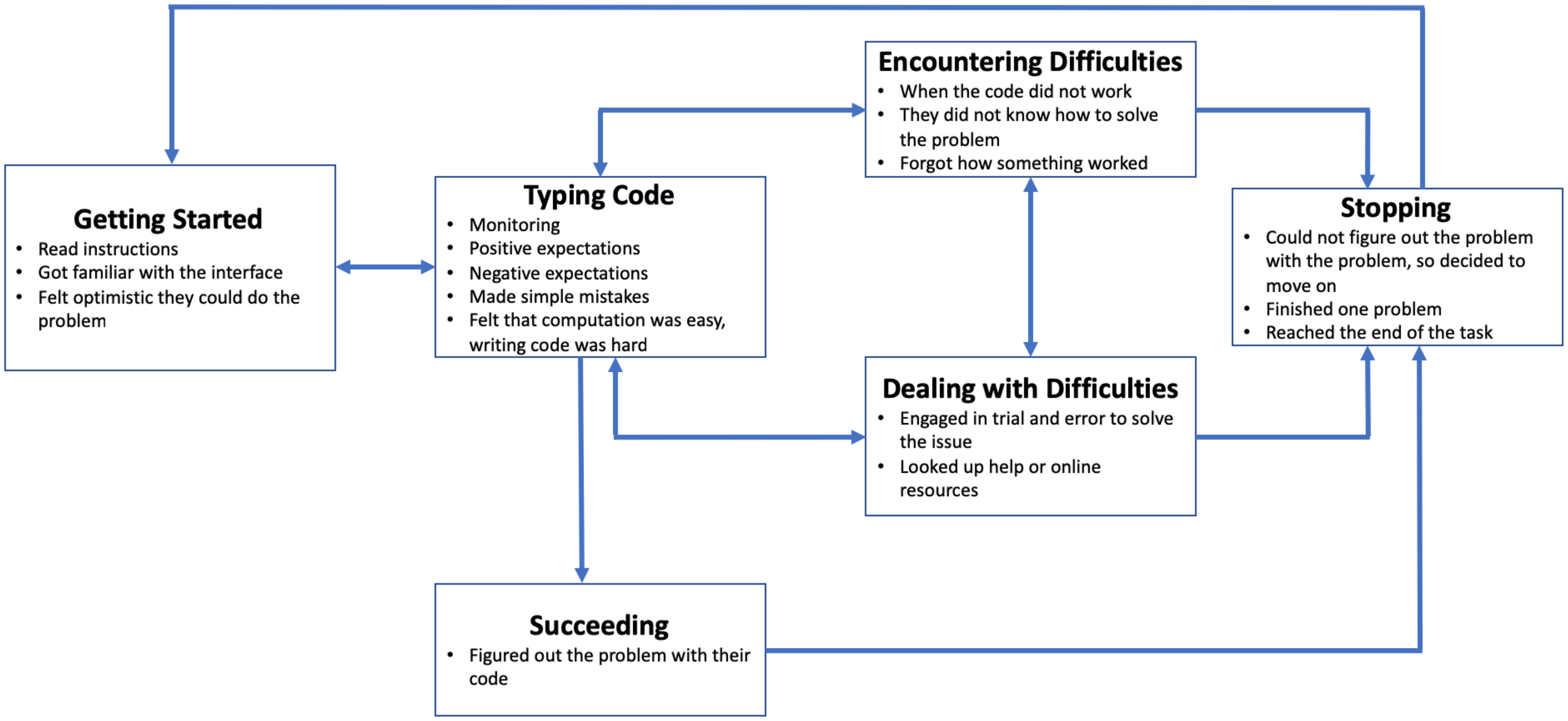

Getting Started
Read instructions
Got familiar with the interface
Felt optimistic they could do the problem
Typing Code
Monitoring
Positive expectations
Negative expectations
Made simple mistakes
Felt that computation was easy, writing code was hard
Encountering Difficulties
When the code did not work
They did not know how to solve the problem
Forgot how something worked
Dealing with Difficulties
Engaged in trial and error to solve the issue
Looked up help or online resources
Stopping
Could not figure out the problem with the problem, so decided to move on
Finished one problem
Reached the end of the task
Succeeding
Figured out the problem with their code

Figure 13. Process Model That Students Follow While Working on the Programming Task

Students generally experienced frustration when they encountered difficulties. However, after overcoming the difficulties, students experienced relief. On the other hand, students experienced anxiety when they anticipated an undesirable event to occur. If the event did not occur, students experienced relief. There were many instances where students reported feeling neutral: they did not feel any type of positive or negative emotions. Students felt neutral when they engaged in an activity which flowed smoothly and was primarily uneventful. However, there are exceptions to these general patterns of experiencing emotions, which are discussed in the subsequent sections.

Table 6: Frequency of Occurrence of an Emotion in the Retrospective Think-Aloud Data

Emotion	Number of occurrences of this emotion	Percentage	Number of students who reported this emotion
Anger	11	2	5
Annoyance	4	1	9
Anxiety	52	10	15
Boredom	7	1	3
Confusion	49	10	12
Contentment	14	3	7
Disappointment	27	5	12
Enjoyment	15	3	8
Frustration	123	24	17
Hope	15	3	7
Hopelessness	12	2	3
Joy	35	7	12
Nervousness	12	2	4
Neutral	46	9	11
Overwhelmed	12	2	4
Pride	27	5	11
Relief	45	9	13
Sadness	5	1	3
Shame	4	1	2
Total	515		

The following sections discuss in detail the emotions students experienced during each stage of the process model and the reasons they provided for experiencing those emotions. Table 7 summarizes the findings for this research question.

Getting Started

As soon as students finished the Before-AEQp survey, they were prompted to start the programming task. Students entered the "Getting Started" stage when they started working on a problem they had selected, or when they stopped working on one problem and started on a different problem. In this stage, students read instructions, became familiar with the interface, and saved files and folders. The following sections describe these behaviors, the emotions that the students experienced, and the reasons for those emotions.

Reading the Instructions

In this stage, while reading the problems on the programming task, students experienced positive deactivating emotions like relaxation and relief, as evidenced from excerpts by Andrew and Rachel.

> *I'm trying to read problem number one and at that point I noticed, I have done this before. So I'm more neutral and relax at that point, I'm in relief [...] I was relaxed because I was expecting a much harder question.*
>
> *Andrew, Problem 1*
>
> *Alright my first thought, I was just kind of going over the task. It was a lot shorter than I thought it was going to be, which I felt pretty relieved about. I thought it was going to be a lot more like what our homework assignments are.*
>
> *Rachel, Problem 1*

According to the control-value theory of achievement emotions, deactivating emotion like relief is induced when an expected event did not occur (Pekrun et al., 2002), or if failure has been averted (Pekrun, 2006). Andrew and Rachel were anticipating failure because they were expecting to encounter challenges that would hinder their success on the programming task. However, after reading the instructions, they realized that the problems were familiar, or were shorter than their expectations, hence averting failure.

Table 7: Emotions Students Experience and Reasons for Experiencing Those Emotions

Stage	What is happening within each stage	Emotions	Reason for each emotion
Getting Started	Reading the instructions	Relaxation	Felt familiar with the problem
		Relief	Expected the problem to be harder
	Getting familiar with the interface	Anxiety	Unsure about saving files in the right place
		Overwhelmed	Did not have code templates for the task, which they get in the class
			Unfamiliar with working on a Windows computer
	Optimistic they could do the problem	Relief	Knew they could solve the problem
		Joy	Anticipated to enjoy writing code
Typing Code	Monitoring	Anxiety	Anticipated an error, so double checked everything
		Frustration	Attempted to write correct syntax to ensure that the computer understands it
	Positive Expectations	Pride Relief	Converted math formula to code correctly Confident that the code would work
	Negative Expectations	Anxiety	Did not expect the code to work
	Simple Mistakes	Frustration	Made spelling mistakes
	Computation is easy, writing code is hard	Frustration	Knew how to solve the problem, but felt that writing code for it was hard
Encountering Difficulties	When the code does not work	Frustration	Wondered why their code was not working Thought it would work but did not
		Anxiety	Conscious about the time limit
	They did not know how to do a problem	Hopelessness	Thought they could not solve the problem
		Frustration	Did not initially know how to do the problem
		Shame	Did not know how to do the problems even though they were easy
	Forgot how something worked	Frustration Anger	Could not recall the MATLAB function for a command
Dealing with Difficulties	Engaged in trial-and-error to solve the issue	Frustration	Wondered why the code was not working

Table 7 Continued

Stage	What is happening within each stage	Emotions	Reason for each emotion
	Looked up help or online resources	Pride Relief Joy	Tried what help suggested and the code worked successfully
		Shame	Failed to overcome the difficulty even after obtaining help
Succeeding	Figured out the problem with their code	Relief	Remembered the function they needed
		Hope Joy	Fixed the code work after many errors
		Pride	Figured out something they didn't know how to do
Stopping	Could not understand the problem, so decided to move on	Frustration	Felt they would rather work on something they know
		Hopelessness	Thought they could not do the problem
	Finished one problem	Pride	Felt proud of their capabilities
		Relief	Attempted something that was not too challenging for them
		Disappointment	Took too long to finish one problem
	Reaching the end of the task	Shame	Could not solve the problem, which was seemingly easy
		Frustration	Failed to solve the problem
		Relief	Did not have to try anymore as the task was over

Getting Familiar with the Interface

Some students started working on the programming task by familiarizing themselves with the interface or by saving files in the right location. Getting familiar with the interface may introduce incidental challenges, not related to programming task itself, mainly because it takes up additional time from the task. During this stage, students experienced negative emotions like anxiety, because in their classes, students were accustomed to starting with complete code templates, which were not provided to them during the programming task. Excerpts from Christina and Rachel explain the incidental challenges they faced.

> *At this time, I'm also probably looking to see if I need to save it in a specific name, but it just said to put it in the folder on the desktop. […] Probably a little bit of may be anxiety just making sure I'm saving it in the right place.*
>
> *Christina, Problem 1*
>
> *Looking at it, kind of a new interface for me so I wanted to familiarize myself with that. I normally work on a Mac so just, I don't even know what this interface is so just working on it […] So, I figured out how to save there and it was a lot more blank than what I am used to and the interface that I have has a lot more words to it, this was kind of empty [referring to the blank MATLAB script], kind of overwhelming, especially with the scripts we were supposed to make. They [referring to ENGR 132] have a whole header for us already so when you started the assignment, you already had words down, kind of helps you a little bit and so, I kind of felt overwhelmed looking at all this blank space to code with […] I guess mostly feeling overwhelmed. But then kind of relieved that the problems weren't too bad.*
>
> *Rachel, Problem 1*

Christina felt anxious about saving files in the right place. According to the control-value theory, anxiety is induced when there is uncertainty about the outcome of an event (Pekrun, 2006; Pekrun & Perry, 2014). Christina's video shows that she struggled to save files in the right place at the beginning of the programming task, and spent a few minutes trying to figure out how to save the MATLAB script in the right place, before moving on to typing code for problem 1. Christina's behaviors as shown in the video confirms the uncertainty she experienced while saving the file in the right location, hence she felt anxious.

Rachel on the other hand encountered two unfamiliar situations (Windows interface and lack of code templates), which caused her to feel overwhelmed. For Rachel, code templates were an important part of the programming process. She was the only participant who mentioned the absence of the code templates. Rachel's video of the screen capture shows that after completing problem 1, she worked on other problems, but she returned to problem 1 and added comments so that her MATLAB scripts looked like a code template similar to the ones students get in ENGR 132 (see Table 3). Rachel's feeling of being overwhelmed may be conceptualized as anxiety, as Rachel was dealing with two unfamiliar events, causing uncertainty about the outcome of the programming task. Rachel's uncertainty could impede her performance on the programming task. Rachel's experience corroborates with the control-value theory, which suggests that anxiety is induced when students are uncertain about the outcome of a certain event (Pekrun, 2006).

Optimistic They Could Do the Problem

Once the students had selected a problem, they evaluated if they would be successful at writing code for that problem. If students felt confident about their ability to write code for the chosen problem, they experienced positive emotions like relief and joy. Excerpts from Sarah and Ella explain why they experienced relief and joy respectively.

> *I guess relief 'cause once I read the first problem, it was something I knew I could do vs. a hard problem which I didn't know how to do. So, it was definitely relief.*
>
> *Sarah, Problem 1*
>
> *So the first one was pretty straightforward. I realized that I could just write, um, if statements form, which should be pretty easy and it was something that I had just done, so I was pretty happy about [...] I would say joy because it's um, yeah. And I feel like I could probably enjoy writing the code too.*
>
> *Ella, Problem 1*

Sarah and Ella's experiences conform with the control-value theory, which suggests that positive emotions like joy and relief are a function of outcome expectance, when there is high probability of success and non-occurrence of failure respectively (Pekrun, 2006). In addition to experiencing positive emotions, data from both Sarah and Ella exhibit confidence about their abilities to successfully solve some problems on the programming task. Confidence in their abilities could be explained by the control-value theory, which explains that students exhibit action-control expectancies when they are confident about their ability to successfully perform an action (Pekrun & Perry, 2014).

Although Sarah felt relieved and confident after reading the first problem, she spent almost the entire duration of the programming task (27 minutes) writing correct code for problem 1 (see Table 3). Hence, her feeling of relief may have been short lived, because her video shows her struggling on problem 1. Problem 1 involved writing a basic control structure (if/elseif/else/end) using logical operators. These programming constructs were introduced to students early in the semester and they had had ample practice with these constructs while working on the homework problem sets. Despite that, Sarah repeatedly switched between writing code, searching online for help, and fixing errors. Sarah was finally able to successfully complete problem 1, but she had almost no time left to work on the other three problems. Ella on the other hand, was not just optimistic she could do problem 1, she successfully completed the problem in almost eight minutes, hence, giving her ample time to work on the other three problems.

Typing Code

During this stage, while typing code, students exhibited some behaviors and thoughts, which they verbalized during the retrospective interview. The following sub-sections explain students' behaviors and expectations while typing code, and the emotions they experienced during this stage, or anticipated to experience in the subsequent stages.

Monitoring

Some students adopted a cautious stance while typing code. These students expected to encounter errors in their code, and hence decided to double check everything as they typed. These students adopted preemptive behavior, where they knew they had the propensity to make mistakes. To avoid experiencing negative emotions, they made sure they typed correct code the first time, instead of making errors and then fixing them. Both Randy and Rachel explained how they double checked everything while writing code.

I do a lot of little things and big things that ultimately screw me over when it comes to coding so I just kind of anticipate that something will pop up. I feel like I'll get to the end of it and something will be wrong, and I can't find out what it is, it just gives me a lot of anxiety. I mean the only thing I can do about it is power through and double check everything which is why I go back and I take a really long time to type it because I want to make sure it's right the first time and I don't have to go back and figure out what's wrong.

Randy, Problem 2

I just, with coding its very temperamental, like it's very picky and has to be exact and so if you don't know exactly how it's supposed to be done and the computer can't understand what you are trying to do, it's frustrating because you can't get too much feedback from it. You just kind of have to keep working through it, it might be something simple like an extra period when there isn't supposed to be or the order of something is wrong or might be a spelling error you skipped over so that frustration comes out a lot when I am coding.

Rachel, Problem 1

Although both Randy and Rachel were cautious and double checked everything as they wrote code, both had different reasons for adopting these behaviors. While Randy attributed his cautious coding style to his own propensity to make mistakes, Rachel attributed her caution to the fact that the syntax of the code had to be precise before it is executed. Additionally, both students did not experience frustration and anxiety while typing their code, but they were expecting to experience negative emotions in case their code did not work. Hence, to avoid feeling those emotions, they took necessary precautions.

Video of the screen capture shows that both students exhibited similar patterns while working on the programming problems. Table 3 shows that both students worked on the problems in the same order (1→2→3→4). However, Rachel went back to problems 1 and 3 after she had worked on the four problems. Both students spent similar amounts on time on each problem. Although Rachel struggled with some of the problems, she did not encounter any syntax error during the programming task, hence confirming that she adopted a cautious stance while typing code. Randy initially did not encounter syntax errors, confirming his cautious typing style, but while working on problem 4, Randy repeatedly encountered syntax errors, suggesting that he gave up on the cautious style because he was running out of time and wanted to complete his work on the programming task.

Positive Expectations

In this stage, students had typed the code but had not executed it. While students typed code, they had certain expectations about the outcome of what they had typed. Some students had positive expectations, hence, they felt confident their work would yield positive outcomes. In this case, students felt proud of their abilities and in some cases relief, as shown in excerpts by John and Randy.

> *I think pride and relief because I correctly translated the math language to the MATLAB language. Okay. Um, so that felt good to have a working equation. And right now I'm plotting which is something I'm fairly comfortable with.*
>
> *John, Problem 4*
>
> *Um right now, just kind of, I feel a little bit of pride because I was getting ready to test my code and I was pretty confident that it would work.*
>
> *Randy, Problem 1*

John and Randy both explained that they felt proud of the code that they had typed. However, there are subtle differences between their experiences. John felt proud because he had successfully converted a mathematical equation into MATLAB code, and he felt confident that his code would execute successfully. On the other hand, Randy felt proud because he thought whatever he had written would execute correctly. John's video shows that he converted the formula into code for problem 4 but he had not executed it, hence he reported feeling proud in the retrospective think-aloud interview. His feeling of pride is confirmed by his performance on problem 4 (see Table 3); It took John about nine minutes to correctly complete problem 4 and he is one of the four students who received a perfect score on this problem. Randy's excerpt refers to his progress on problem 1, where he is confident that his code would work. Video data confirms Randy's sense of pride because his code for problem 1 executed without any glitches or errors. However, Randy did not get a perfect score on the problem (see Table 3) because he did not implement the solution as a user-defined function, as stated in the instructions of the problem and he spent more than half the time of the programming task on problem 1 (16 minutes).

According to the control-value theory, pride is experienced when students achieve something they had perceived as hard or unattainable. In other words, pride is a control dependent emotion, which is experienced when students perceive their success to be caused by their own actions (Pekrun, 2006; Pekrun et al., 2002). For these students, writing code was a hard task, but once they had written the code, they felt proud of their accomplishment. These students had not executed the code and hence, they had not seen the output of the code. However, they had positive expectations and felt confident that their code would execute correctly. The video data further confirms students feeling of confidence and pride about the problem and if their positive expectations materialized into positive outcomes.

Negative Expectations

In this stage, students had typed the code but had not executed it. Some students had negative expectations from the code they had written, and hence they thought their code would be erroneous. In such a case, students felt anxious and frustrated, as shown in excerpts by Jack and Randy.

> *At this point I was nervous and anxious, so I had to test my code to see if it worked [...] I kinda was expecting my code was not going to work but I didn't know how begin to fix the issue. So, I decided to go back to the browser.*
>
> *Jack, Problem 2*
>
> *I anticipated messing up somewhere. Just, I do a lot of little things and big things that ultimately screw me over when it comes to coding so I just kind of anticipate that something will pop up…I feel like I'll get to the end of it and something will be wrong and I can't find out what it is, it just gives me a lot of anxiety.*
>
> *Randy, Problem 2*

According to the control-value theory, anxiety is induced if students focus on experiencing failure (Pekrun, 2006). Even though students had not yet executed the code, Jack and Randy felt anxious because they were anticipating their code to be erroneous. Jack explained that he decided to open the browser to search for help, to fix errors in his code. Randy on the other hand anticipated negative outcomes and that induced anxiety.

Students could have both positive and negative expectations about different problems on the programming task. For instance, Randy also had positive expectations (see previous section on positive expectations). Since he had both positive and negative expectations at different times during the programming task, it is likely that he felt confident about his ability to successfully solve one problem, and he doubted his ability to successfully solve another problem. In the previous section, Randy had positive expectations about problem 1, and for the most part, his positive expectations yielded positive outcomes. Hence, Randy felt proud even before he had executed his code for that problem. However, in the current excerpt about problem 2, he anticipated committing errors and hence he thought he would experience anxiety. Randy's video of problem 2 shows that he correctly wrote built-in functions for sum and sine, but when he encountered the running sum problem, he hardcoded the solution and moved on to the next problem.

Making Simple Mistakes

While typing code, some students made simple incidental mistakes (e.g., spelling mistakes). Making these incidental mistakes hindered students' progress because it took time to fix these mistakes, hence causing frustration, as shown in the excerpt by Becky.

> *A little bit of frustration as I kept on making spelling errors and I was like, come on, Becky you can spell, like normal words. This is not complicated and also slightly frustrated at MATLAB 'cause it does not have spell check. 'cause sometimes some things are meant to be wrong, like variables, so I was like it can't just tell me like this would make my life easier.*
>
> *Becky, Problem 1*

According to the control-value theory, frustration is induced when the activity is not sufficiently controllable (Pekrun, 2006). Since Becky experienced incidental challenges like spelling mistakes, she did not feel in control of the programming task. She also added that MATLAB does not have spell check functionality, hence, it could not point out the spelling mistakes, and that frustrated her. It is interesting to note that Becky is not experiencing frustration due to programming per se, but because of the spelling mistakes she made while typing code.

Computation Is Easy, Writing Code Is Hard

Sometimes students perceived they could find a solution of a descriptive problem in their minds, but faced difficulty converting their solution into syntax. In such a case, they experienced frustration because they thought they knew what to do but were unable to implement their thoughts correctly. Rachel explained her feeling of frustration when she could not convert her thoughts into MATLAB code.

> *Oh, but this last problem on problem 2, I knew what it was saying, I would know how to do it on a piece of paper, it's just that doing it on a computer, putting in these commands so that the computer understands you, I couldn't do it. Still at the end, thinking about it, I couldn't do it. It's frustrating knowing how to do something without coding but then not knowing how to do it with coding because it seems so easy but it's just figuring it out, it's not actually that the computations are difficult, it's finding the commands to do that or the ways to do it on MATLAB. I don't ever remember doing like that in class which again is why I felt frustrated.*
>
> *Rachel, Problem 2*

Here Rachel used the phrase "piece of paper" metaphorically to refer to the computation she did in her mind. However, she was unable to write code corresponding to the solution in her mind, and consequently, she felt frustrated. To be able to convert a solution into code, students construct mental models, called "notional machines" in the context of computer science (Guzdial, 2015). However, novice students lack the detailed mental models needed to convert problem statement into code (Robins et al., 2003). This difficulty of converting problem statement into code leads students to feel frustrated. According to the control-value theory, failure induces frustration (Pekrun, 2006). Since Rachel failed to translate the solution in her mind to executable code, she felt frustrated. Rachel's video shows that she spent almost 2 minutes (out of the total 6 minutes spent on problem 2 – see Table 3) trying to write code to calculate the running sum. Even though Rachel thought she could do the problem in her mind, she moved on to problem 3 as soon as she realized she could not write code for running sum. Rachel's example shows that students did not have to struggle on one problem for too long to feel frustrated.

Encountering Difficulties

In the "Encountering Difficulties" stage, students' difficulties ranged from encountering syntax errors to realizing they did not know how to solve the problem or parts of the problem. These difficulties hindered students' progress, hence inducing negative emotions. In this stage, the predominant emotion was frustration because students experienced failure while working on the problems. These findings align with the control-value theory, which suggests that frustration is induced when failure is experienced (Pekrun, 2006). However, there are subtle differences in students' experiences, which are discussed in the following sections.

When the Code Does Not Work

Most students experienced negative emotions when their code did not execute successfully. Students predominantly experienced frustration, but some also experienced anger and anxiety. Becky and Harris explained their feeling of frustration when their code did not execute successfully.

> *So here I was a little angry and a little frustrated 'cause of the same reasons as why is this not working, I do not know what's wrong with the program at this point, I don't know what I am doing wrong. I was starting to blame myself for getting it wrong, blaming MATLAB for getting it wrong [...] Also getting slightly anxious with the time limit, 'cause I knew there was a time limit and I had wasted a lot of time on this.*
>
> *Becky, Problem 2*
>
> *I think a bit more frustration because I thought the while loop that I had, um, that I had made up would solve the problem in terms of how to get those values that I was looking for or the f of t values [referring to f(t)] and then, um, when it didn't work, a lot of frustration there.*
>
> *Harris, Problem 4*

Both students felt frustrated because their code did not execute successfully. These findings conform with the control-value theory, which suggests that failure induces frustration in students (Pekrun, 2006). However, there are some differences in their experiences.

Becky experienced two different emotions at the same time, due to different but related events. She encountered syntax errors in her code, which required time to fix. However, she did not have enough time left, hence, she felt anxious. These findings conform with the control-value theory, which suggests that anxiety is induced if students' focus on failure (Pekrun, 2006). Becky's feeling of anxiety is not related to programming per se, but because she was running out of time on the programming task. Becky's video confirms her feeling of anxiety. She spent 9 minutes on problem 2 (see Table 3), during which she realized she would not be able to overcome the errors, hence, she moved on to the next problem.

On the other hand, Harris had high perceived control on the task, that is, he was expecting his code to work. When his code did not work, he felt frustrated. According to the control-value theory, perceived control determines the likelihood of obtaining a certain outcome (Pekrun & Perry, 2014), and when that outcome is not achieved, students experience negative emotions. Harris's video confirms that he encountered errors on problem 4 because of which he felt frustrated. He eventually overcame the errors and successfully executed the code. However, he did not get a perfect score (see Table 3) on the problem because he ran out of time and could not write code corresponding to one of the learning objectives (LO 4.3. Format plots for technical presentation – see Table 2).

They Did Not Know How to Solve the Problem

Students experienced negative emotions when they realized they did not know how to solve a certain problem. The negative emotions included hopelessness, frustration, and shame. It must be noted that the same situation (not knowing how to solve a problem) may induce different emotions in different students. Literature suggests that people may experience a similar event differently, hence experiencing different emotions (Barrett, 2006). The following excerpts from Jack, Rachel, and Randy explain the negative emotions they experienced when they realized that they did not know how to solve a problem.

> *At the beginning of the segment I was feeling hopelessness, because after reading problem 3 thoroughly I decided that I don't think I can do this, so I won't attempt it until I solve the other three.*
>
> *Jack, Problem 3*
>
> *I was kind of hoping that I would have had that 'aha' moment, I would remember something, I would think of something, some way how to do it or I look back at it, I'd read it again and interpret it in a different way. I was kind of hoping for something like that which is why I moved to the next problem. Didn't let it get to me too much then, but I did feel frustrated that I didn't initially know how to do it.*
>
> *Rachel, Problem 3*
>
> *Kind of shameful that I didn't know how to do that. In general, like after this whole programming task, I just kind of felt a lot of shame because I felt like a lot of these problems were really easy compared to like what I'm normally given and I flat out didn't know how to do half of them so that was kind of frustrating and a little bit of shame.*
>
> *Randy, Problem 3*

All three students experienced negative emotions upon realizing that they did not know how to solve problem 3. However, their emotional reactions were different.

Jack felt hopeless when he realized he could not solve a problem. However, instead of dwelling on the problem, he decided to work on the other problems on the task and return to this problem later. Jack's experience conforms with the control-value theory which suggests that students experience hopelessness when they are certain they will fail on an academic task (Pekrun, 2006). Jack's video data confirms his feeling of certainty that he would not be able to solve this problem. Jack failed to score any marks on problem 3, even though he spent about 8 minutes trying to solve this problem. Additionally, video data confirms that Jack returned to this problem twice after his first attempt (see Table 3).

Rachel on the other hand, experienced frustration when she realized she could not solve the problem. However, she hoped to figure out the solution sooner or later, hence, she moved on to the next problem. Rachel's experience aligns with the control-value theory, which suggests that students experience frustration when they encounter activities that are not sufficiently controllable

(Pekrun, 2006; Pekrun & Perry, 2014). Rachel spent a total of 3 minutes working on problem 3, but she did not achieve anything (see Table 3). Rachel's video shows that she came back to problem 3 for a short duration, but to no avail. Rachel may have felt out of control because of her inability to make progress on problem 3, hence she felt frustrated.

Finally, Randy felt ashamed after the programming task because he did not know how to do most of the problems on the task. Randy's experience varies slightly from the other two students. While Jack and Rachel did not know how to work on one problem, leading to negative emotions, Randy felt ashamed because of the entire task. According to the control-value theory, shame is experienced when failure is judged to be caused by oneself (Pekrun, 2006). Randy attributed his failure on the task to himself. He explained that most of the problems on the task were easy, even then he was unable to do them successfully, hence he experienced shame. Randy's feeling of shame and his inability to solve seemingly simple problems is confirmed by his performance on the programming task. That is, Randy partially completed three problems on the programming task and was unable to achieve a perfect score on any problem (see Table 3).

Like Jack, Rachel, and Randy, most students were unable to solve problem 3 completely (see Table 3). Problem 3 involved writing nested loops. Since students had recently learned nested loops in class and may not have had enough practice with homework problem sets, they struggled with problem 3 during the programming task. Moreover, loops are hard for novices to learn (Guzdial, 2015; Soloway et al., 1983), and that would have added to their struggle with problem 3.

Forgot How Something Worked

Some students experienced negative emotions like frustration and anger, when they realized they had previously worked on something, but forgot how to use their previous learning during the programming task. Randy's excerpt encapsulates his experience of forgetting the MATLAB function for exponent.

I couldn't remember that e was exp and then I tried e in the command window, saw it didn't work and still put it in anyway so yeah, a lot of frustration and kind of anger towards myself because I was angry at myself for not remembering what it *was. Have I used e before? Yeah, absolutely. I'm realizing now that on the exam, I had forgot it too but it's fine so yeah, kind of anger and frustration in this moment.*

Randy, Problem 4

Randy's experience can be explained by the control-value theory which suggests that negative emotions like anger and frustration are experienced when students fail on the task they undertake (Pekrun, 2006). Randy's video shows that he was unable to write code corresponding to one of the learning objectives for problem 4 (LO 4.1. Use built-in functions to perform algebraic and trigonometric calculations – see Table 2). Since he had forgotten how to use the exponential function in MATLAB, he hardcoded the constant value of exponential (e = 2.7) to code the function, hence he did not get a perfect score on the problem (see Table 3). After hardcoding the value for exponential, Randy worked on fixing syntax errors in the code. When he executed the code, his code ran without any errors, but as soon as he executed the code, the time for the programming task finished.

The control-value theory distinguishes between frustration and anger as activity and outcome emotions (Pekrun, 2006; Pekrun & Perry, 2014). In other words, frustration is experienced while students work on the academic task, and anger is experienced at the end of the task. Both frustration and anger are experienced when students experience failure. Even though Randy did not experience failure per se (the output for his code was correct), he experienced frustration and anger at himself for forgetting the built-in function for exponential.

Dealing with Difficulties

In the "Dealing with Difficulties" stage, most students engaged in trial-and-error to resolve the errors they encountered. When they were unable to resolve errors using trial-and-error, they used resources such as MATLAB and online help. Some students did not engage in trial-and-error but sought help as soon as they realized they had encountered an error. Interestingly, these data show that most students persevered through the programming task and did not give up despite the challenges they faced. However, some students abandoned individual problems, with the intention of returning to them later.

Engaged in Trial-and-Error to Overcome Errors

All students who participated in the study encountered syntax errors while working on the programming task. However, different students dealt with the errors differently. The most common strategy that the students adopted was trial-and-error. While engaging in trial-and-error, students went back and forth over the work they had written and tried to identify errors in their code. This process induced negative emotions like frustration in students, as shown in the following excerpt by Becky.

> *So, at this point I made up a random vector to test [whether] it was MATLAB screwing up or if I was doing something wrong. So, I purposefully used negatives and positives so I could check what it was doing so yeah. So, then I got the same error in Y and then that's when I realized that I was doing something wrong. At this point, it was pretty neutral, slightly frustrated just 'cause it was not working. […] Just using trial an error and then figuring out what was happening, so it was pretty neutral at this point.*
>
> *Becky, Problem 2*

Becky encountered syntax errors while writing code. To resolve the errors, she engaged in the process of trial-and-error. When she was unable to fix errors, Becky felt frustrated. According to the control-value theory, students experience frustration when they experience failure (Pekrun, 2006). Literature suggests that novices normally do not have the developed mental models to correct errors in their code (Robins et al., 2003). Like Novices, Becky probably lacked the mental models required to correct errors in her code, and hence she felt frustrated. Becky's performance on problem 2 confirms that despite her trial-and error efforts, she was unable to successfully complete the problem. She completed only one of the three sub-problems and spent most of the 9 minutes figuring out how to write code to calculate the running sum (see Table 3).

Looked up Help or Online Resources

Students looked at help when they encountered syntax errors, or if they did not know the syntax of a function. According to the instructions provided, students could raise their hand and ask the researcher questions, but none of the students took that option. While seeking help, students had two types of experiences. Sometimes they were able to resolve errors by looking up help. In this case, students experienced pride, relief, and joy. At other times, they were unable to resolve errors, despite spending time looking up resources. In this case, students experienced shame and frustration.

Successfully Overcame Errors Using Help

When students encountered syntax errors, they looked at either local MATLAB or online help resources. Many times, students were able to resolve errors by looking at help. In this case, students experienced pride, joy and relief, as explained in the following excerpts by Anna and Ella.

> *I usually start with the help functions in MATLAB and then I'll go to Google if I need more help. Oh, now I'm figuring it out, here we go. […] Yep, now I feel better, I was proud thereafter, I think I kind of nodded to myself even, I do that a lot, I'm like okay, you got it, you're good. There's a lot of giving myself pep talks when I'm coding.*
>
> *Anna, Problem 1*
>
> *Once I googled it and like the first thing that came up, like it said, like cumulative sum, I'm like, oh, that makes sense. And then I'm sure I felt relief and joy. I was like, oh good. Then I'm. So, I clicked on it and then I read through it briefly and then it was exactly what I wanted it to. So, I put in my code and it worked.*
>
> *Ella, Problem 2*

According to the control-value theory, pride is control-dependent and is induced when success is attributed to one's own actions (Pekrun, 2006). Anna felt proud because she was able to successfully resolve errors by herself. Although Anna felt proud that she successfully completed problem 1, she spent more than half the time of the programming task on problem 1 (see Table 3). Her video shows that she repeatedly looked at help resources to search for correct syntax for the if/elseif/else/end structure and logical operators. Anna's emphasis is on completing individual problems on the task instead of completing the entire programming task, hence, she may not be concerned about finishing all four problems on the task.

Ella on the other hand experienced joy and relief. According to the control-value theory, joy is experienced when students are in control of the activity (Pekrun, 2006), and relief is experienced when failure is averted (Pekrun, 2006). Ella averted failure by searching for a correct solution online. Interestingly, Ella experienced two emotions at the same time: joy and relief. According to the control-value theory, joy and relief are seen as functions of outcome expectancy (Pekrun, 2006; Pekrun & Perry, 2014). In other words, joy and relief are experienced when there is a high probability of success and non-occurrence of failure. Table 3 confirms Ella's help seeking behaviors for the running sum problem. She spent 7 minutes on problem 2, which is slightly less than the average time students spent on this problem. Ella is one of the three students who earned a perfect score on problem 2.

Could Not Resolve Errors Even After Looking at Help

Some students were unable to resolve syntax errors in their code, even after consulting MATLAB or online help. These students experienced shame, as shown by an excerpt from Christina.

> *I actually tried looking up help in MATLAB and I found results but I couldn't really figure it out from that so that was when I experienced a little shame because you would think, if I do help and you get a sample code, you can figure it out but I just, for whatever reason, didn't get it. And then jumped to hard coding it which was really bad programming practice was but I guess it made me feel a little better because they give you the vector so you know which values you should get.*
>
> *Christina, Problem 2*

According to the control-value theory, shame is induced when failure is judged to be caused by oneself (Pekrun, 2006). Christina felt that she should have been able to resolve errors in her code, but she was unable to. She attributed the failure to herself, hence experienced shame. However, Christina did not give up even though she could not find a solution online. She resorted to hardcoding the function in her code, expecting to come back if she found a better solution. Christina's video confirms her inability to use help to calculate the running sum of the vector. She looked at help a few times and even though she was able to find the cumsum function online, she could not figure out how to correctly use the function. Her inability to correctly use the cumsum function is further confirmed by her score on problem 2 (see Table 3). She successfully completed the first two sub-problems, but hardcoded the solution for running sum, before moving on to the next problem.

Succeeding

Students reached the "Succeeding" stage when they had successfully finished a problem on the programming task. After encountering and dealing with difficulties, many students successfully resolved the errors. Once they realized their code executed successfully, they experienced positive emotions like relief, hope, and pride. The following excepts by Jack and Anna explained the emotions they experienced when they successfully fixed the problem with their code.

> *I felt some relief when I remembered the function for E was EXP. So, at this moment the graph worked and I felt more hopefulness and here I just probably felt happy because after so many errors I was finally able to make the code work properly.*
>
> *Jack, Problem 4*
>
> *I was proud of myself because I had successfully figured out how to do something that I didn't know how to do. And that always gives into pride and almost, something like relief, faith in my abilities to do the coding.*
>
> *Anna, Problem 2*

Jack and Anna's experience corroborates with the control-value theory, which explains that students experience positive emotions like joy, pride, and relief when they have high control of the activity they engage with, and experience hope when they have moderate control over the activity (Pekrun, 2006; Pekrun & Perry, 2014). Moreover, the control-value theory explains that students experience relief, when they are able to prevent failure (Pekrun, 2006). Jack spent just one-minute writing code for problem 4 (see Table 3), during which he recalled the MATLAB exponential function, hence, he felt relieved. On the other hand, Anna experienced relief because she was able to successfully do something she did not know, that is, she figured out how to calculate the running sum successfully. For most students, calculating running sum was a difficult undertaking. Anna is one of the four students who received a perfect score on problem 2 (see Table

3). Since Anna was able to successfully solve a problem she did not know how to do, she felt pride in her abilities (Pekrun, 2006).

Stopping

During the "Stopping" stage, students encountered three types of events. Students decided to move on to the next problem, when they did not understand a problem. In this case, students experienced frustration and hopelessness. When students successfully finished one problem, they experienced positive emotions like pride, relief, and joy. However, some students experienced negative emotions like disappointment because they had taken too long to finish one problem. Students experienced a range of emotions when the programming task ended. The predominant emotions students experienced during this stage were negative, that is, shame, frustration, hopelessness, annoyance, and disappointment. However, some students also reported feeling relieved because the task was over.

Could Not Understand the Problem, so Decided to Move On

When students were unable to successfully complete one problem, they decided to move on to the next problem, instead of spending time trying to figure out a solution for the problem. In this case, students experienced frustration and hopelessness, as described by an excerpt from Jack.

> *I remember thinking that problem 3 is just hopeless. I am going to attempt it last if I ever get to it. After reading problem 3 thoroughly I decided that I don't think I can do this, so I won't attempt it until I solve the other three.*
>
> *Jack, Problem 3*

Jack experienced hopelessness because he realized he could not do one problem on the task. Jack's experience aligns with the control-value theory, which suggests that hopelessness is experienced when students have low expectancy of success (Pekrun, 2006). Since Jack was not expecting to succeed on a problem, he felt hopeless, and decided to move on to another problem, expecting to come back to it later. However, Jack's video data shows that he came back to problem 3 twice after first attempting it, but he was still unable to successfully complete it (see Table 3).

Finished a Problem

When students successfully finished one problem, they felt relieved. They also felt proud of their capabilities. However, students felt annoyed when they had spent too much time on one problem. These experiences are explained by the excerpts from Ella and Sarah.

> *I think it was nice to get the first one done too. So, I probably felt I'm proud of my capabilities of coding. So, I probably felt relief that it's something that wasn't too challenging for me. I would say I was proud of my work as I was coding. It's pretty straightforward, because I know what I was doing then it was like the code works.*
>
> *Ella, Problem 1*
>
> *Here I felt a little relief because I finished the first problem. I didn't really know how the time had gone by, but I was just glad I was past the first problem. I was a little disappointed that I took so long but I kind of pushed that aside because moving on the next problem was like Okay, well I finished one, which means I can finish another one, I can do this now.*
>
> *Sarah, Problem 1*

According to the control-value theory, pride is experienced when success is judged by one's own actions, while relief is experienced when failure is averted (Pekrun, 2006). While Ella was proud of her capabilities after coding a problem, she also felt relieved knowing that the problem was not too challenging for her. Ella's video data shows that she completed problem 1 in 8 minutes, which is below the average time (12.6 minutes) it took students to complete the problem (see Table 3). Since Ella finished problem 1 in a short amount of time, it indicates that she did not find the problem challenging. Sarah was relieved because she finished the first problem on the task, but she also added that she was disappointed because she took longer than expected on one problem. According to the control-value theory, the non-occurrence of expected success instigates disappointment (Pekrun, 2006). Sarah expected to complete one problem in less time, but she spent almost the entire duration of the programming task on problem 1.

Reaching the End of the Task

Most students experienced a mix of positive and negative emotions, when the task ended after 30 minutes. Almost all students were relieved that the task was over. However, students experienced a variety of negative emotions at the end of the task, primarily frustration, shame, and disappointment. The following excerpts by John and Rachel show students' mixed emotions at the end of the task.

> *When the screen goes black. I suppose its frustration because I never got to figure out that last problem. I'm a little bit of shame because I wasn't able to do it and I feel like I could have done it easily. Okay. Um, but also relief because the session was over.*
>
> *John, Problem 3*
>
> *So, I think when it did get over and I realized the thirty minutes had passed that quickly, kind of frustrated that I wasn't able to get the third, relieved that I don't have to try anymore.*
>
> *Rachel, Problem 3*

Most students felt relieved after the task finished. According to the control-value theory, when students work on a relatively unimportant task, that is, they do not value the task, they experience relief (Pekrun, 2006). Since this task was low stakes for the students, with no grade attached to their performance, students did not value the task. Hence, most students felt relieved when the task ended. At the same time, these students experienced negative emotions because they were not satisfied with their performance on the task. Both John and Rachel were frustrated at the end of the task, because they were unable to successfully complete the last problem they were working on. John and Rachel's experience corroborates with the control-value theory, which explains that students experience frustration when they deal with difficult tasks, or when they experience failure (Pekrun, 2006). John also experienced shame because he was unable to successfully complete the problem. His experience corroborates with the control-value theory, which suggests that students experience shame when they think failure was caused by themselves (Pekrun, 2006).

Research Question 2

How do student emotions change as a result of working on programming problems?

The primary data used to answer this research question were the retrospective think-aloud interview responses that were prompted by students' scores on the Before- and After-AEQp surveys. These data are supplemented by students' responses from the Before- and After-AEQp (see Table 11 in Appendix L for individual Before- and After-AEQp scores), and electrodermal activity data. Some emotions do not have items either in both Before-AEQp and After-AEQp (relief, anxiety, hope and hopelessness), and hence, change in these emotions were not measured. The average values of before and after responses for each emotion are stated in Table 8 (shown as a dotted line on Figures 14, 15, 16, and 17). The findings of this research question are summarized in Table 8.

Table 8: Change of Emotions Before and after the Programming Task

Category	Emotion	Change	Reason
Positive Activating	**Enjoyment**	Decreased	Faced challenges while programming
			Could not transfer knowledge from class to the programming task
			Felt they had limited time
		Increased	Found the problem on the programming task to be familiar
		No Change	Felt neutral about the programming task
	Pride	Decreased	Fell short of their performance expectations
		Increased	Met or exceeded their own expectations
			Fixed errors on their own
		No Change	Felt no pride in their coding skills
Negative Activating	**Anger**	Increased	Encountered unanticipated obstacles during the task
			Fell short of the standards they set for themselves
		Decreased	Experienced the task to be easier than they had originally anticipated
	Shame	Increased	Fell short of the standards they had set for themselves
			Known how to solve majority of the problems on the task
			Couldn't figure some things on the programming task
			Couldn't finish all four problems
		No Change	Felt no embarrassment asking for help because they are not the best programmer
			Participating in the research study

Positive Activating Emotions

The following section elaborates on how participants' positive activating emotions (enjoyment and pride) changed as a result of working on the programming task.

Enjoyment

Findings from this study suggests that for most participants, enjoyment decreased between the beginning and the end of the programming task. However, some participants reported an increase in enjoyment, while some others did not experience any change in enjoyment. In

Figure 14, each line connects the Before- and After-AEQp score for one student and Table 11 in Appendix L shows individual Before- and After-AEQp scores. The difference in these scores were used to qualitatively infer whether the student's perceived enjoyment increased or decreased between the beginning and the end of the programming task. The dotted line in

Figure 14 connects the average of the participants' scores for enjoyment on the Before-AEQp with their average score for enjoyment on the After-AEQp. These descriptive data are supplemented by participants' interview responses from the retrospective think-aloud interview.

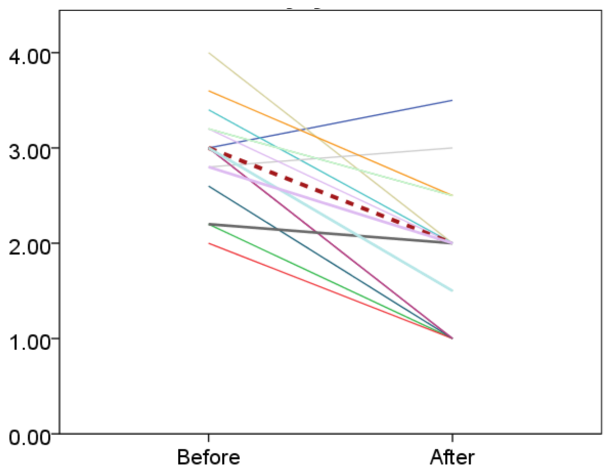


Figure 14. Change in Enjoyment Between the Beginning and the End of the Programming Task

Enjoyment Decreased

During the think-aloud interview, participants provided multiple reasons for their decreased enjoyment.

Faced Challenges While Programming

Some students perceived a decrease in enjoyment between the beginning and the end of the programming task because they felt frustrated by the challenges that they encountered. This group of students mentioned that they disliked programming. A combination of the dislike for programming and the occurrence of challenges led to decreased enjoyment between the beginning and the end of the task for some students. For instance, both Lilly and Sarah explained their dislike for programming, and the challenges they faced during the programming task.

> *I really do not enjoy programming at all. It's really not something I'm very good at it and I usually enjoyed things that I'm good at because I like being good things and I just get so frustrated all the time.*
>
> *Lilly*
>
> *I don't like programming in general, but I think it went down because problem 1 gave me more errors than I anticipated ... It went down because I wasn't enjoying myself because of all the errors even though I knew it's one of those things where you know you can do it but because there are so many things in the way, you get more and more frustrated so you don't enjoy it.*
>
> *Sarah*

Both Lilly and Sarah mentioned feeling frustrated in addition to decreased enjoyment between the beginning and the end of the programming task. According to the control-value theory of achievement emotions, failure instigates frustration (Pekrun & Perry, 2014). Table 3 shows that Lilly's performance was above average on the programming task (42 out of a total of 56), where the average score for all study participants is 31.4. Despite her above average performance, Lilly experienced decreased enjoyment and reported that she disliked programming. The last problem Lilly worked on before the programming task ended was problem 3, and she had struggled on this problem. Lilly's struggle on problem 3 may have resulted in decreased enjoyment when the task ended. On the other hand, Sarah did not perform adequately (28 out of a total of 56) primarily because she spent 27 minutes working on problem 1. Her video shows that she struggled significantly on problem 1. She encountered frequent syntax errors because of which she kept on referring to help resources and hence she experienced decreased enjoyment between the beginning and end of the task.

Could Not Transfer Knowledge from Class to the Programming Task

Often students fail to transfer the knowledge learned in one context (classroom) to another context, in this case a test-like programming task. Christina explained how she felt when she was unable to transfer what she had learned in class, to another setting.

> *The biggest one for me is only really feeling confident about one of the four problems but then ending on a problem where like, I did not get anywhere. Do I find that fun, no. I think especially because, I do learn things in class but it's frustrating and discouraging when I cannot show that. And then I feel like if I can't have pride in my education, do I find that enjoying? No.*
>
> *Christina*

Christina explained that she had learned concepts in class, but she was unable to transfer that knowledge to another context, thus inducing negative emotions like frustration. Literature suggests that novices generally lack the ability to transfer knowledge from one context to another because of which they had difficulty completing academic assignments successfully (Perkins, 2010); hence, failure to perform successfully leads to frustration (Pekrun, Frenzel, Goetz, & Perry, 2007; Pekrun & Perry, 2014). Christina's video data confirms her comment about feeling confident about one problem (problem 1) on the task. Christina spent 11 minutes on problem 1 and her score on this problem was 15 out of 16 (see Table 3). On the other hand, Christina's video data shows that she spent 7 minutes working on problem 4, but she struggled to transfer the knowledge from class to solving this problem, hence, she received a low score on this problem (3 out of 16). Since problem 4 was the last problem Christina worked on before 30 minutes ended, she experienced decreased enjoyment and frustration when the task ended.

They Had Limited Time

Some students experienced lower enjoyment at the end of the task than at the beginning, because they thought there wasn't enough time to complete all four problems on the task. Harris said that he could not finish the problems due to lack of time.

> *I think just the fact that I kind of went in with the thought that I'll probably finish these four problems. I'm just thinking that they were going to be easier. The first two were quite easy, the second two were pretty challenging and I was a little frustrated with myself, so I don't think the task itself was that enjoyable.*
>
> *Harris*

Harris had positive expectations about the task; that is, the problems were easy, and he would be able to finish all four of them. However, at the end of the task he was frustrated that his performance did not meet his expectations and hence he did not enjoy the task. Harris attempted all four problems on the task. He achieved a perfect score on problem 1 and 2, and an above average score on problem 4, but he did not complete problem 3 before the 30-minute period ended (see Table 3). Harris's experience is consistent with the control-value theory, which suggests that success is attained when students have enough control over the activity and its outcomes (Pekrun et al., 2007; Pekrun & Perry, 2014). Since he encountered challenges while working on the task, and found the problems to be hard, he felt frustrated and experienced decreased enjoyment between the beginning and end of the task.

Enjoyment Increased

Some students anticipated that they would enjoy the task, and they did report enjoyment when the task ended. For instance, Mark's excerpt explains that he enjoyed the task because one of the problems was familiar to him and that he had done that problem before.

> *It could have been because it was more fresh in my mind, when I look the before one, the last problem that I had done was before spring break. And I felt like I did*

a pretty good job on it. So, I enjoyed that one a little more than when I was working on this one.

Mark

Mark enjoyed working on one programming problem on the task, because he had previously done one of the four problems in class. Since the problem was familiar to him, he felt confident that he would be able to successfully complete that problem, hence, he felt in control of the situation. Mark's experience conforms with the control-value theory, which explains that people feel in control when they are able to successfully initiate or perform an activity (Pekrun et al., 2007; Pekrun & Perry, 2014). Table 3 shows that Mark worked on only three problems on the programming task (problem 1, 3, and 4). His overall performance was far below average (14 out of 56, average = 31.4), because he failed to save the files for problems 3 and 4. Hence, his performance does not reflect his actual ability. However, in the retrospective think-aloud interview, Mark reported an overall increase in enjoyment because he said was familiar with one problem on the task.

No Change in Enjoyment

Some participants experienced no change between the beginning and end of the task. Emily felt neutral while she worked on the task and her enjoyment stayed the same before and after the programming task.

It's not necessarily something that I would ever choose to do if it wasn't an assignment but at the same time, I'm not dreading it. I'm very neutral. It's not something I look forward to but it's not something that I dread doing it […] there's like a lot of other things I would do instead of coding but at the same time, it doesn't stress me out.

Emily

As a conscientious student, Emily undertakes academic tasks assigned to her, irrespective of the domain. Table 3 shows that Emily is one of the highest achieving students in this study (45 out of 56, average = 31.4). Despite her high achievement, she did not experience a change in her enjoyment at the end of the task because she had low value for programming. Emily's low value for programming could be explained by the control-value theory which explicates that students' emotions are influenced by the value they give to an activity (Pekrun & Perry, 2014).

Pride

During the retrospective think-aloud interview, participants reported that their pride in themselves after finishing the programming task either decreased, increased, or there was no change. In

Figure 15, each line connects the Before- and After-AEQp score for one student. The difference in these scores was used to qualitatively infer whether the student's perceived pride increased or decreased between the beginning and the end of the programming task. The dotted line in

Figure 15 connects the average of participants' scores for pride on the Before-AEQp with their average score for pride on the After-AEQp. Table 11 in Appendix L shows individual Before- and After-AEQp scores. These descriptive data are supplemented by participants' interview responses from the retrospective think-aloud interview.

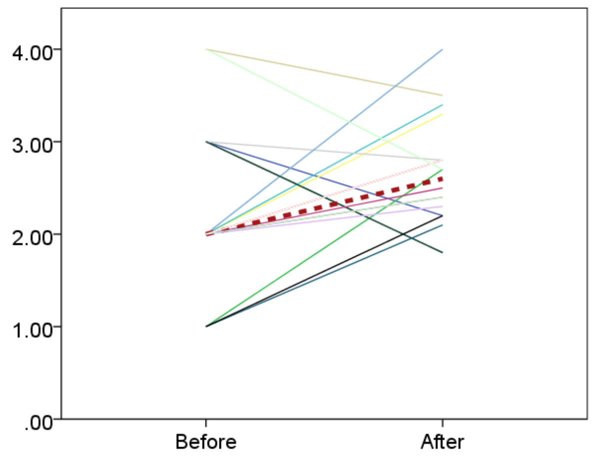


Figure 15. Change in Pride Between the Beginning and the End of the Programming Task

Pride Decreased

Positive activating emotions like pride relate positively with effort students expend in an academic task (Pekrun et al., 2011). Hence, students feel proud if the outcomes of their efforts meet their expectations, and vice versa. Students experienced a decline in their perceived pride when their performance did not meet their expectations. Christina explained why her pride declined.

> *How can I feel good about myself if I literally just failed, especially the red messages and stuff, the error messages, like no one likes getting an error but it does really hurt your pride because what you thought was going to work didn't so now it's up to you to try to figure out how to fix it but if you don't necessarily know how to do that.*
>
> *Christina*

Christina thought the code she had written would work, but she encountered errors that she could not fix. When she realized that she did not know how to fix the errors, her pride decreased.

Students also experienced decreased pride, when they thought they had not met the standards they had set for themselves. For instance, Harris explained how his pride decreased because he had not met his own standards.

> *Just because I think I hold myself to fairly high standards and so when I don't kind of reach those standards, I get pretty upset with myself. Yeah! So, I think again, like I was pretty frustrated with myself.*
>
> *Harris*

He thought he had enough time to finish all the problems on the programming task, but his inability to finish the problems made him feel frustrated. Table 3 shows that Harris is one of the high achievers in this group of students, but because he was unable to complete problem 3, he thought he had not met his own standards. Harris's experience can be explained by the social cognitive theory, which explains that people react negatively when they fall short of, or violate personal standards (Bandura, 1991).

Pride Increased

Pride positively relates to the effort students expend and the self-regulation of their learning (Pekrun et al., 2011). Pride is control dependent, that is, it is induced by self-related factors that influence outcomes, for instance, ability or lack thereof (Pekrun et al., 2007; Pekrun & Perry, 2014). Students feel proud when they succeed in academic tasks, thus leading to enhanced motivation (Oades-Sese, Matthews, & Lewis, 2014).

The overarching theme from these data is that students felt proud of their performance if they met or exceeded the expectations they had from the task. Data from the study uncovered different ways in which students met or exceeded their expectations, thus leading to increased pride. Moreover, it is important to mention that every student had different conception of what success meant to them. For instance, Anna explained that her pride increased because she had achieved more than what she had expected before the task started.

> *I was pretty proud of myself, I knew I had done more than I thought I was going to do, and especially because I did finish the one, completely […] I find a lot of times I get stuck is with the first one where we all have one error somewhere and I can't find that one error and that is the most frustrating thing in the world.*
>
> *Anna*

Generally, when Anna encounters errors on a programming problem, she gets frustrated trying to fix the errors and then she cannot complete other problems that she is supposed to work on. During the programming task, Anna was able to avoid this hindering behavior, hence, she achieved more than her expectations, leading to increased pride. Anna's overall performance on the programming task is below average because she did not attempt problem 3 and did not have enough time to complete problem 4 (see Table 3). However, she achieved a perfect score on problems 1 and 2. She spent almost the entire duration of the programming task (26 minutes) on the first two problems because of the challenges she encountered. However, she overcame the challenges, hence, leading to increased pride at the end of the task. Anna's experience conforms with the control-value theory, which suggests that pride is induced when success is perceived due to internal control by the student (Pekrun et al., 2007, 2011). In this case Anna evaluated the factors that hampers her success, overcame those factors, and achieved something, leading to a sense of increased pride (Oades-Sese et al., 2014; Pekrun & Perry, 2014).

In ENGR 132, students mostly work in pairs on their programming assignments. They also have help readily available (e.g., access to teaching assistants). Sarah explained that this programming task gave her the opportunity to test her programming skills in an environment that did not provide any help in the form of peers and teaching assistants. In such an environment, she was able to fix syntax errors and successfully execute programs, and that resulted in increased pride.

> *I don't know if it was pride but it made me feel kind of better about myself 'cause normally my programs don't run as I want them to and then normally I ask my peers but in this case I had to fix it all by myself. So, after I fixed it all and it ran, it made me feel better because I was like, I finished this all by myself. So, this is something I can do, which was not something I tend to feel when I'm done programming, like tasks and problem sets.*
>
> *Sarah*

As a novice, when Sarah encountered errors in her class, she relied on her teammates or teaching assistants for help. However, during this task, she fixed errors by herself. Since, she was able to fix errors by herself, that is, she felt in control of her circumstances, hence, she experienced increased pride between the beginning and end of the programming task. Sarah's experience conforms with the control-value theory, which suggests that students experience positive emotions when they feel in control of their circumstances (Pekrun et al., 2011).

No Change in Pride

Some participants experienced no change in pride between the beginning and the end of the task. For instance, Emily's pride did not change because she generally does not take pride in her coding skills.

> *I mean coding isn't something that I necessarily take pride in. It's not that I'm bad at it. It's just there are a lot of other things that I am much prouder of, that I just value more than my skills related to coding.*
>
> *Emily*

The control-value theory suggests that students' emotions are influenced by the value they give to an activity (Pekrun & Perry, 2014). Emily explained that she valued other things more than programming because of which she did not experience any change in her pride.

Negative Activating Emotions

Negative activating emotions (anger, anxiety, and shame) have a mixed effect on students. On one hand, these emotions could seriously undermine intrinsic motivation, thus hampering learning and performance (Pekrun et al., 2011). On the other hand, the activating nature of these emotions could encourage students to deal with the events that caused the emotions, thus strengthening extrinsic motivation (Pekrun et al., 2011, 2002). Research suggests that negative emotions may be most beneficial for tasks that require careful attention to details, in order to prevent or to detect errors (Pekrun et al., 2011). Hence, negative activating emotions may not necessarily impair student learning and performance. The following sections describe how students' anger and shame changed between the beginning and end of the programming task. Students did not elaborate on their experiences of anxiety during the retrospective think-aloud interview, hence, anxiety is not explained in the findings.

Anger

During the retrospective think-aloud interview, participants explained why their anger increased or decreased between the beginning and the end of the programming task. In

Figure 16, each line connects the Before- and After-AEQp score for one student. The difference in these scores were used to qualitatively infer whether the student's perceived anger increased or decreased between the beginning and the end of the programming task. The dotted line in

Figure 16 connects the average of the participants' scores for anger on the Before-AEQp with their average score for anger on the After-AEQp. Table 11 in Appendix L shows individual

Before- and After-AEQp scores. These descriptive data are supplemented by participants' interview responses from the retrospective think-aloud interview.

Anger Increased

Before the programming task started, most participants anticipated that they would not feel angry. However, their perceived anger increased between the beginning and the end of the task. These students encountered unanticipated obstacles during the task, which frustrated them, subsequently leading to increased anger. According to the control-value theory, frustration and anger are likely to occur together (Pekrun & Perry, 2014). Anger and frustration are aroused when working on a task requires so much effort that the resulting experience becomes detrimental for the student (Pekrun & Perry, 2014). The following excerpts by Sarah and Ella shows why they experienced increased anger.

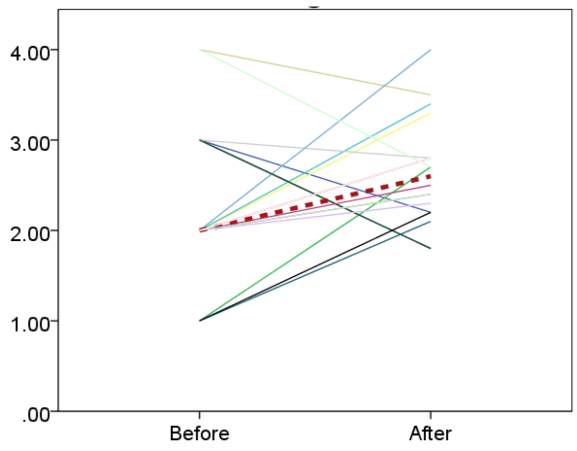


Figure 16. Change in Anger Between the Beginning and the End of the Programming Task

Well I was annoyed, I was frustrated. As I said before, it was such a simple task, but I couldn't finish it in the way I wanted to, or in the speed I wanted to. In like every error that kept on coming up I was like Again? Or like Come On! I thought this was as simple as I made it out to be but it's not.

Sarah

I think personally not knowing how to do something, it's very frustrating to me. So, coming across that question where I had no idea how to even start it, I was probably

> *pretty upset about that. I just wasn't happy about the last question, but I think that's why I saved it for the end is because I knew like I didn't want to.*
>
> *Ella*

Sarah and Ella's example can be explained by the control-value theory of achievement emotions. The control-value theory explains that students experience negative emotions when they do not feel in control over their actions and outcomes (Pekrun & Perry, 2014). Sarah explained that she was expecting the task to be simple, but it was not. Moreover, she kept on encountering errors while coding. Hence, she did not feel in control of her actions (coding), and outcomes (could not finish the task due to errors), and hence experienced frustration, leading to increased anger after the task finished. Sarah's struggle on the programming task is confirmed by data in Table 3, which shows that Sarah spent 27 minutes on problem 1, which took students an average of 12.6 minutes to complete.

Ella on the other hand felt out of control for a different reason. She did not know how to start working on the last problem (problem 3 for Ella – see Table 3). Hence, she was unable to complete it, leading to increased anger. Even though Ella is the highest achieving student in this study (see Table 3), she still experienced increased anger at the end of the task, and frustration while working on the programming task, because she was unable to finish the last problem on the task. Ella's experience could also be explained by the peak and end rule, which posits that a person judges their overall experience based on how they felt at the peak of their experience and at the end of the experience (Kahneman, 2000).

Some students had set certain standards for their performance, when they were unable to meet those standards, they experienced increased anger after the task ended. Anna mentioned that she sets high standards for herself, and when she does not match those standards, she gets angry.

> *I was upset that my program had errors but it's always more anger with myself I think than it is at anyone else. I'm a pretty self-motivated learner and I usually feel like the standard I perform at is my responsibility so that there's no excuse if, I can sit here and tell you the way they teach doesn't match with my learning style but in the end, that means that I should be going out and searching for, I should be finding a way to teach it to myself that matches up with my learning style because I do*

realize that you have a room full of a hundred people, you can't teach it to fit everyone's style.

Anna

Anna's experience can be explained by the social cognitive theory, which explains that people experience negative emotions when they fall short of their self-set personal standards (Bandura, 1991). She also added that as an intrinsically motivated learner, she is responsible for her own learning. Hence, she does not like to blame external factors for her lack of learning. She would rather search for material and teach herself. Anna's attainment of her goal or self-set standard was interrupted, and because she does not attribute her learning to external factors (including the class and teachers), she feels angry at herself for not reaching her standards. Anna's example contradicts the attribution theory, which states that when we fail or make mistakes, we attribute the cause to external factors instead of blaming ourselves (Weiner, 1972).

Anger Decreased

Some participants were expecting the programming task to be hard, and hence their expectation of anger was high. However, at the end of the task they realized the task was easier than they had anticipated. Hence, their anger at the end of the task was less than what they had expected to experience before the task started, as shown by an excerpt from Lilly.

I was expecting the task to be a lot harder. Okay. Because like some of them I found easier and some of them were harder, so I was like really annoyed at that. But then afterwards it was a lot better.

Lilly

Before starting the programming task, Lilly was expecting the programming problems to be hard. She did find some problems to be hard, but some problems were also easy. Overall, she explains that the problems were easier than her expectations and hence, her anger decreased coming out of the task. Her performance on the programming task confirms her claim that some problems were easy, and some were hard. Table 3 shows that Lilly successfully finished problems 1 and 4, received partial credit on problem 2 because she struggled on the running sum sub-problem.

However, she was unable to complete problem 3. Lilly's performance on these problems (see Table 3) suggest that although she found some problems harder than other problems, she was eventually successful at most problems, hence, her anger decreased between the beginning and end of the task.

Shame

During the retrospective think-aloud interview, participants explained why their shame increased or why there was no change in shame before and after the programming task. In

Figure 17, each line connects the Before- and After-AEQp score for one student. The difference in these scores were used to qualitatively infer whether the student's perceived shame increased or decreased between the beginning and the end of the programming task. The dotted line in

Figure 17 connects the average of the participants' scores for shame on the Before-AEQp with their average score for shame on the After-AEQp. Table 11 in Appendix L shows individual Before- and After-AEQp scores. These descriptive data are supplemented by participants' interview responses from the retrospective think-aloud interview.

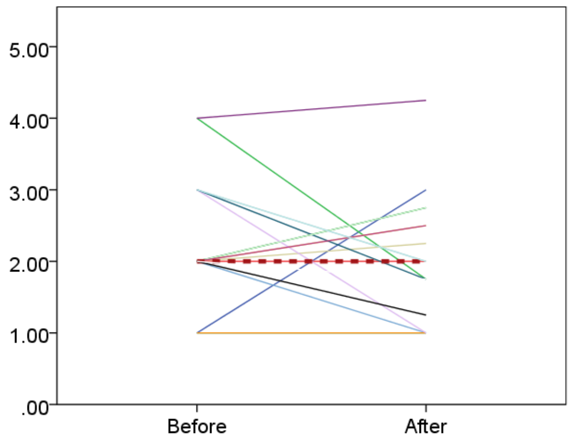


Figure 17. Change in Shame Between the Beginning and the End of the Programming Task

Shame Increased

Most participants experienced shame because they experienced failure, and they attributed the failure to themselves instead of external causes, as shown in the excerpts by Randy and Christina.

> *Oh, absolutely because I felt like I should have known the majority of this and I flat out didn't so yeah, it was very embarrassing to myself [...] I mean I'm being recorded, I'm being tested and so, performing the way I did, made me feel pretty bad.*
>
> *Randy*
>
> *I just look dumb, I couldn't figure some of them out especially I think the plotting one, its literally plotting, and I looked up the help function, I still couldn't get it so that was embarrassing. I don't like when things don't work, I mean, because all thing has been covered. I should know how to do it, but I couldn't figure it out so yeah, I felt that's embarrassing.*
>
> *Christina*

Randy said, "I should have known the majority of this"; Christina said, "I should know how to do it." These experiences of Randy and Christina conform with the control-value theory, which suggests that shame is induced when failure is judged to be caused by oneself (Pekrun, 2006). Randy and Christina's struggle on the programming task is evident by the fact that both did not receive a perfect score on any problem on the programming task. Both received near average or slightly above average scores on problems 1 and 2 (see Table 3), but below average scores on problems 3 and 4. Both attributed the cause of failure to themselves. However, Christina explicitly called herself stupid because she was unable to successfully solve the problem despite using help.

No Change in Shame

Some participants experienced no change in shame between the beginning and end of the programming task. Although these students were not successful at completing the programming task, they did not attribute their failure to themselves. Hence their perceived shame did not increase from their anticipated shame. Erica explained the reason why she did not feel ashamed:

I'm not the worst programmer in the world but I'm not the best programmer in the world, and I know that so I don't feel like embarrassed asking for help because I'd rather ask for help and get it than sit there quiet and not understanding it.

Erica

I did not feel ashamed as to my MATLAB performance. As I said I've never had experience with MATLAB before this semester. So, I thought for being a beginner this was alright. There is nothing to be ashamed about.

Jack

Erica and Jack's experiences are in accordance with the control-value theory, which suggests that shame is induced when failure is judged to be caused by oneself (Pekrun, 2006). Both Erica and Jack are not attributing their failure to themselves, and hence they did not experience any change in their perceived shame.

Electrodermal Activity Data and Emotions

For the purpose of analysis, I used electrodermal activity (EDA) data from 16 students, because one student's EDA data was not captured by the shimmer device. For each student, the EDA data is divided into different sections, depending on the part students engaged with. There were five parts in the programming task. The first was the baseline, where students sat in front of the computer with their heads on the table and did nothing for four minutes. After baseline, students filled the Before-AEQp survey, (two minutes on an average). After the Before-AEQp, students worked on the programming task (MATLAB session) for exactly thirty minutes. Once the programming task was over, students completed two surveys: After-AEQp (two minutes on an average) and the Big-Five Neuroticism sub-scale (one minute on an average).

The EDA data consists of two components: tonic and phasic (Boucsein, 2012). The phasic component of the EDA signal refers to a short, fast emotional or cognitive response (Boucsein, 2012), hence, we used phasic EDA for the analysis. To account for interindividual differences in the EDA, I calculated range correction, which normalizes the phasic EDA between 0 – 1 (Boucsein, 2012; Villanueva et al., 2018). I then calculated the mean and standard deviation for each event, for each student, as shown in Table 9.

Literature suggests that students' EDA may decrease after they have worked on a cognitively challenging task (Villanueva et al., 2018). Hence, I calculated the Wilcoxon signed rank test between the range-corrected EDA of baseline and After-AEQp. However, the results suggest a non-significant difference between the two variables (Z = 33.5, p = 0.13), as shown in Figure 18.

Students' EDA data could be explained by considering the progression of parts shown in Table 9 and by individual participants' data. Different students handle cognitive and emotional stimuli differently, as evidence by the EDA data shown in Table 9, and as explained in the following three examples.

Table 9. Mean and Standard Deviation of Range-Corrected Phasic EDA

Pseudonym	**Baseline**		**Before-AEQ**		**MATLAB Session**		**After-AEQ**	
	M	**SD**	**M**	**SD**	**M**	**SD**	**M**	**SD**
Andrew	0.509	0.073	0.548	0.096	0.391	0.046	0.552	0.131
Anna	0.605	0.140	0.547	0.191	0.490	0.150	0.423	0.195
Becky	0.650	0.068	0.484	0.063	0.484	0.063	0.552	0.128
Christina	0.526	0.095	0.562	0.144	0.455	0.093	0.529	0.139
Ella	0.544	0.080	0.530	0.101	0.528	0.088	0.592	0.123
Emily	0.597	0.096	0.529	0.144	0.529	0.144	0.523	0.125
Erica	0.650	0.062	0.537	0.133	0.549	0.037	0.529	0.113
George	0.551	0.051	0.502	0.147	0.542	0.082	0.470	0.122
Harris	0.620	0.077	0.381	0.161	0.595	0.060	0.633	0.126
Jack	0.491	0.137	0.458	0.171	0.576	0.102	0.525	0.153
John	0.503	0.125	0.388	0.174	0.533	0.138	0.476	0.141
Lilly	0.498	0.096	0.476	0.166	0.520	0.149	0.539	0.213
Martha	0.556	0.108	0.489	0.160	0.455	0.123	0.508	0.142
Rachel	0.491	0.597	0.504	0.138	0.585	0.112	0.503	0.129
Randy	0.543	0.077	0.548	0.112	0.499	0.086	0.497	0.149
Sarah	0.490	0.051	0.584	0.119	0.514	0.092	0.490	0.139

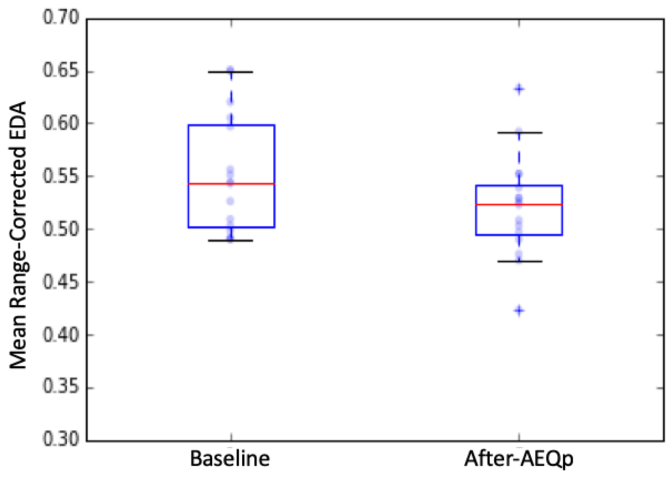


Figure 18. Baseline vs. After-AEQp Range-Corrected EDA

Compared to the baseline (*M*=0.509, SD=0.073), Andrew's mean range-corrected phasic EDA for the pre-survey was *M*=0.584, SD=0.096. Since the mean values were so close, I inferred that Andrew experienced no arousal between the two events (baseline and pre-survey). However, during the MATLAB session, there was substantial decrease in Andrew's arousal response (*M*=0.391, SD=0.046), indicating most likely a physiological arousal due to cognitive activation as Andrew was engaged in high-level algorithmic thinking. Literature suggests that programming involves higher learning gains (Robins et al., 2003), hence, Andrew experienced low arousal and high cognitive activation during the MATLAB session. Immediately after the MATLAB session, Andrew's arousal levels increased substantially (*M*=0.522). Finding from Andrew suggests an increased sympathetic activity (fight and flight response) during the application of a mental load compared to the baseline (Cacioppo, Tassinary, & Berntson, 2007; Picard, 2016; Setz et al., 2010). Andrew's video data confirms these findings as Andrew struggled while working on these problems. He went back and forth between writing code and looking at help resources many times in an attempt to solve the problems on the task.

Becky's baseline values were much higher than any of the other events that followed (*M*=0.650, SD=0.068). This difference suggests that Becky was undergoing emotional or cognitive stressors before the start of the experiment. Similar work published by Villanueva et al., (2018) indicates the importance of anticipatory responses in electrodermal activity data. Becky's video suggests that Becky had an uneasy disposition which affected the way she wrote code and made incidental mistakes along the way. Becky kept on switching between scripts unnecessarily and she also executed code multiple times without making any change to the code, pointing to the possibility that she may have been emotionally distressed before starting the programming task.

Finally, John began with a high arousal value for the baseline (*M*=0.503, SD=0.125) compared to the pre-survey (*M*=0.388, SD=0.174) and this level did not change much compared to the post-survey. This suggests a decreased recovery of the EDA data, and subsequently a potential inhibitory process of sympathetic activity during the recovery phase (Visnovcova, Mestanik, Gala, Mestanikova, & Tonhajzerova, 2016).

Research Question 3

What self-regulation strategies do students use to cope with different emotions during the programming task?

During the programming task, the study participants engaged in the three stages of self-regulation, to attain the goal of completing the programming problems.

Figure 19 summarizes the findings for research question 3 and the following sections describe the self-regulation strategies students adopted in each stage.

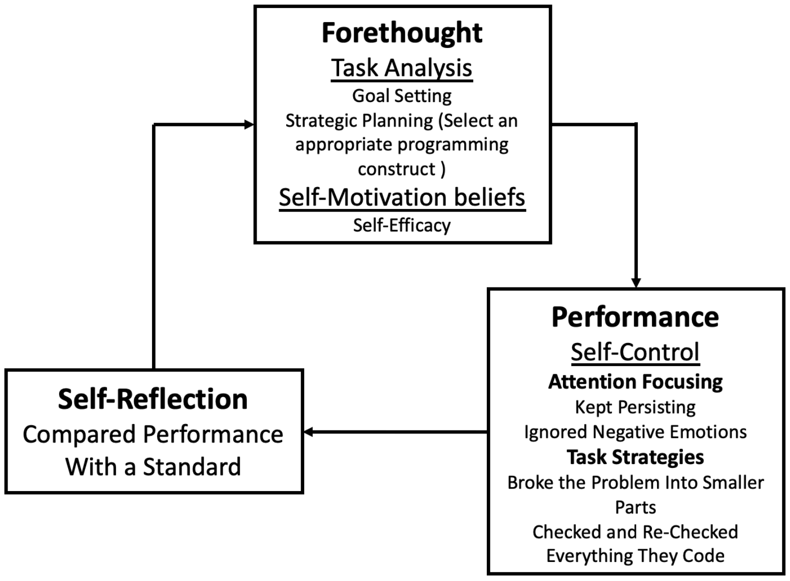

Forethought
Task Analysis
Goal Setting
Strategic Planning (Select an appropriate programming construct)
Self-Motivation beliefs
Self-Efficacy
Performance
Self-Control
Attention Focusing
Kept Persisting
Ignored Negative Emotions
Task Strategies
Broke the Problem Into Smaller Parts
Checked and Re-Checked Everything They Code
Self-Reflection
Compared Performance With a Standard

Figure 19. Self-Regulation During Programming Tasks

Forethought

In the forethought stage, students prepare for the actions that they expect to perform (Schutz & Davis, 2000; Zimmerman, 2002). This stage comprises two processes: task analysis and self-motivation. The following sections describe how the participants in this study experienced each of these processes during the programming task.

Task Analysis

During task analysis, students set specific goals and make strategic plans to help them achieve those goals (Zimmerman, 2005).

Goal Setting

Goal setting is a type of task analysis during which students decide upon the intended outcomes of their performance (Zimmerman, 2005). An overarching goal may be divided into multiple sub-goals. Achieving multiple sub-goals lead to the attainment of the overarching goal (Schutz & Davis, 2000; Zimmerman, 2002). The overarching goal for all students in the study was the same and was defined for them by the design of the study: solve four programming problems in thirty minutes. However, different students may set smaller sub-goals during the thirty-minute programming task, to attain the overarching goal. The following excerpts by George and Harris explain how they set smaller sub-goals during the 30-minute programming task.

> *I'm just reading through the first problem and I think I take a second and read through the rest of them too to just see what I want to start with. I'm just going for the easy one first. Just try to get as much done as possible.*
>
> *George, Problem 1*
>
> *I would see if problem 4 was any easier and if it's like, it kind of get a head start on there, rather than just waste a bunch of time on problem 3.*
>
> *Harris, Problem 4*
>
> *I guess I was analyzing a lot of the problems, trying to analyze which one would be the easier to start with, figuring out which one I was going to have issues with.*
>
> *Rachel, Problem 1*

George, Harris, and Rachel set a sub-goal for themselves, that is, to decide which problem to work on at different times during the programming task. They thought that working on an easy problem would enable them to complete that problem in minimum amount of time, hence, helping them attain the overarching goal of completing all four problems in thirty minutes. Students' goal setting behaviors conform with literature, which suggests that self-regulation involves comparing between the goal and where students perceive themselves to be in relation to that goal, and then making decisions accordingly (Schutz & Davis, 2000; Zimmerman, 2002).

Table 3 shows that all students started working on problem 1. For instance, both George and Rachel decided to work on problem 1 at the beginning of the programming task, because they thought problem 1 was easier than the rest of the problems. Most of the decision making about problem selection happened after students had completed problem 2 (see Table 3). Many students decided to move to problem 4 instead of working on problem 3, because they thought problem 4 was easier than problem 3. For instance, Harris's excerpt explains that he decided to work on problem 4 instead of problem 3 because he thought problem 4 was easier than problem 3.

Strategic Planning

During strategic planning, students plan the actions that they expect to perform (Zimmerman, 2002; Zimmerman, 2005). After the students had chosen a problem, they began to plan the code. During the planning process, they needed to select an appropriate programming construct. Mark explained how he decided to select a loop to solve problem 3.

> *I think I was looking for the FIND function, not a loop and because what I wanted to do was find the indexes of ones that are less than 0.2 and then the next line, just make it new array where the values are the same as the given array except indices from the FIND function, those values are changed to zero. But, when I read systematically, I thought oh, that's a loop because it systematically goes through and tests each one.*
>
> *Mark, Problem 3*

Mark explained he was able to figure out which construct to use (loop vs. FIND function) by reading the instructions systematically. Mark evaluated different cases to see which one would work best, a strategy called heuristic (Perkins, 2010; Polya, 1988). Table 3 shows that Mark spent 16 minutes working on problem 3, suggesting that he did a lot of trial-and-error in trying to figure out which programming construct to use. However, he did not save his file, so his performance on this problem could not be graded.

Emily explained how she decided on a programming construct for problem 2, which required calculating a running sum of integers in a one-dimensional vector.

> *I think I just figured it out, there's obviously another way and I think I was like oh, I could use kind of the similar things like a for loop? If there a function that does the running sum for you, that would probably be a lot easier.*
>
> *Emily, Problem 2*

In this case, Emily wondered if there was a function that implemented the running sum. She thought it would be easier to use a built-in function, instead of using a for loop. However, she was not sure if such a function existed in MATLAB. Emily's video shows that she was unable to find the cumsum function in MATLAB and she ended up trying to write code for running sum. Table 3 shows that she got partial credit for problem 2 because she did not successfully implement running sum.

Self-motivation Beliefs

Students' actions during the forethought stage were driven by some key motivation beliefs of the students (Zimmerman, 2005), for example, students' self-efficacy beliefs. A person's self-efficacy belief is their belief in their own capability to succeed at certain tasks (Bandura, 1977; Zimmerman, 2005). Self-efficacy is task dependent, implying that a person may have high self-efficacy beliefs about one task but not about another. For instance, in the context of programming, a student may have high self-efficacy beliefs about implementing if/elseif/else/end structures correctly, but not about implementing loops.

In this study, most students had high self-efficacy beliefs about their capability to solve some problems on the programming task, because they had worked on similar problems in their ENGR 132 class. For instance, Becky explained that since she knew how to solve problem 1, she felt relaxed.

> *So, this one, I knew was a logic problem and we had worked on those for a while in our engineering course. And at that point I was like ok so this one I know how to do. I can relax now. I got this. SO that made me feel better just I knew how to do what I was supposed to […] Now I am just checking to be sure that I remember the commands, so we have not used them in some time.*
>
> *Becky, Problem 1*

Becky had high self-efficacy beliefs about this problem because she has had previous mastery experience with similar type of problems. According to Bandura, when people judge themselves certain of handling a situation or activity, they behave assuredly, and get involved in that activity (Bandura, 1977).

In the following excerpts, Rachel and Emily mentioned they had confidence in their abilities.

> *I kind of felt relieved that the first problem wasn't too bad, that I knew I could do it. So once I started putting down words, I definitely felt more relieved, kind of felt like I could do it. Kind of felt confident putting down those words.*
>
> *Rachel, Problem 1*
>
> *I was pretty relaxed, I'm just kind of getting a feel for what the problem is asking. I like making functions and just seeing what are the inputs, what are the outputs so then those are easy to find. I'll now just go execute it but I was pretty confident, I knew I needed to use an if elseif statement, like that kind of statements. So, I knew what I was doing.*
>
> *Emily, Problem 1*

Rachel and Emily mentioned that they had the ability to successfully solve the problems (Rachel said *I knew I could do it* and Emily said *I knew I needed to use an if elseif statement*). In light of the theory, Rachel and Emily's experiences could be interpreted as their high self-efficacy beliefs (Pajares, 1996; Schunk, 1991).

Another interesting observation from these data is that the three participants (Becky, Rachel and Emily) had not performed the action (written or executed code), even then they were feeling relaxed or relieved. It is because they had high self-efficacy beliefs in their ability, that is, they knew what they were supposed to do, to solve the problem they were working on. Hence, they feel relieved or relaxed before they start working on the problem. Their high self-efficacy beliefs about problem 1 translated into high performance on the problem. Table 3 shows that all three students received a perfect score on problem 1.

Performance

In the performance stage, students work on the actual task at hand (Zimmerman, 2002). During this stage, students engaged in two types of processes, self-control and self-observation, which affect their motivation and learning (Pintrich & Schunk, 2001).

Self-Control

During self-control, students implemented the methods and strategies that they had selected during the forethought stage. These methods and strategies helped students focus on the task and optimize their performance (Zimmerman, 2005).

Attention Focusing

Students use attention focusing strategies to focus on the task and divert all external distractions while working on an academic task (Zimmerman, 2005). During the programming task, students adopted the following attention focusing strategies to remain attentive to the task.

Kept Persisting

As one attention focusing strategy, students kept persisting despite the challenges they faced or distractions they encountered while working on the programming task. For instance, Erica and Ella both stated that they kept working on the problem to stay motivated about the task.

> *Yeah, I would say concentrated, focused, I wasn't overjoyed at the thought of coding, but it wasn't super strenuous [...] I am definitely still figuring out what I was doing wrong and how to get it right. Probably a little frustrated that it wasn't working to a hundred percent that I wanted it to but overall, okay, I just kind of kept working and used it as motivation to keep going.*
>
> *Erica, Problem 2*
>
> *I think the frustration helped me focus in on the error that I had made. I don't know, I feel like sometimes it helps when you've done something wrong to be able to [in your moment of frustration], you're able to like really like zone in on where the mistake was made. And I think it helped me figure it out pretty quickly. I think in,*

> *in general, like when I get frustrated with something, I probably try harder at it first before I ask for help. It's like I like go rock climbing, which I guess is somewhere I get frustrated a lot. When I'm working on climbing a certain problem, I'll probably try at it for another half hour and then if it still doesn't work, I usually ask for help or just stop working on it and come back to it later.*
>
> *Ella, Problem 1*

Erica and Ella's behavior conforms with literature, which suggests that student persistence is a key indicator of motivation (Pintrich & Schunk, 2001). That is, students who are motivated tend to persist on academic tasks, despite facing challenges. Erica's persistence is confirmed by the video data which shows that Erica was unable to understand running sum despite looking at help. However, she did not give up. Instead, she first hardcoded the solution to understand how running sum worked. After observing the output, she started writing a loop to implement running sum. Table 3 further confirms Erica's comment that she was unable to complete the problem despite persisting. She received partial credit for problem 2 because she produced incomplete code for the problem.

Ella experienced frustration because she encountered an error, but this feeling of frustration increased her focus on the task. To explain her feeling of frustration, she used an analogy. Ella explained that during her rock-climbing sessions, whenever she felt stuck, she experienced frustration. However, she used frustration as a fuel to remain focused on the climbing task. According to the control-value theory, frustration is induced when failure is experienced (Pekrun, 2006). Although Ella experienced frustration because of the syntax errors, her reaction to the frustration was quite positive. Table 3 shows that Ella spent 8 minutes on problem 1, despite encountering and fixing syntax errors and achieved a perfect score. Overall, Ella was the highest achieving student in this study, with a total score of 51 out of 56.

Ignored Negative Emotions

As another self-control strategy, students ignored the negative emotions they experienced. Jack explained how he tried not to think about the anxiety that he experienced while working on the programming problems.

> *So, may be a bit of anxiety because I would end up forgetting the instructions over and over. So, I had to look back and look over and over again between MATLAB and word. Other than that, I was just calm. I try not to think about it [...] I noticed that the elseif statements, I am pretty sure would have done the job but I felt it was not the easiest way and then I began to get anxious [thinking] is the rest of my code doing it in the simplest way or am I just over complicating things so I started writing in comments to remind myself what each line does.*
>
> *Jack, Problem 1*
>
> *Probably getting to the point where I'm getting a little bit of, I don't want to say, not quite as strong as anxiousness but may be like nervous, [thinking] is it going to run? Moment of truth coming here so probably just ignoring it as much as possible, focusing on the more positive emotions and trying to ignore the negative ones.*
>
> *Anna, Problem 1*

Jack felt anxious because he kept on forgetting the instructions, hence, he had to go back and forth between instructions and the MATLAB window. In order to stay focused, Jack stated that he tried not to think about the negative emotions he was experiencing. Moreover, Jack switched to another, low stress activity (commenting code), when he felt anxious about his code. Anna on the other hand felt proud that she had written the code for the problem she was working on. However, as she was about to execute the program, she experienced nervousness and anxiety, wondering if her code would run. However, she pushed the negative feelings aside and focused on the positive emotions that she experienced. Anna experienced both positive and negative emotions at the same time, but she chose to remain focused on the positive emotions, instead of dwelling on the negative ones. Focusing on positive emotions proved to be beneficial for both Jack and Anna, because they successfully implemented problem 1 and received a perfect score on the problem (see Table 3).

Task Strategies

During the self-control stage, students adopted task strategies which assisted them in their learning and performance (Zimmerman, 2005). The participants of this study engaged in the following task strategies, while working on the programming task.

Breaking the Problem into Parts

As one task strategy, students broke down the problem into smaller parts before writing code. For instance, Jack explained how he used this strategy to keep himself on-track.

> *I was just highlighting what I think would help me. At this moment, I am pretty sure I felt confident I knew what I was doing. I got back to the second statement. I do a little bit of thinking and then I decide to start coding again. I believe I was feeling pretty relaxed and neutral. I was reading the instructions and coding, reading the instructions and coding.*
>
> *Jack, Problem 2*

By highlighting useful material, he was breaking information into parts for better comprehension (Zimmerman, 2005). Once he had broken information into parts, he started writing code for the parts he had highlighted. He kept on going back and forth between highlighted instructions and writing code. This strategy was beneficial for Jack, because he was able to write code for problem 2 and also received a score of 11 out of 12 on the problem.

Checking Their Work

As another task strategy, students checked and re-checked everything that they wrote. For instance, Randy mentioned that while writing code, he kept on rechecking the code he was writing, just to ensure he was writing correct code.

> *I try to read the problem as if it was an if statement which I mean, it's kind of written as one but and then, I just feel like I triple check everything in my head in terms of like, what the bounds are for each condition. [...] Um right now, just kind of, I feel a little bit of pride because I was getting ready to test my code and I was pretty confident that it would work.*
>
> *Randy, Problem 1*

Randy needed to check and re-check to make sure that the code he wrote was correct. Because of this strategy, he felt confident that his code would work. Consequently, he felt pride in knowing that his code would execute correctly.

Anna also mentioned using this strategy. However, she explicitly mentioned the reason why she used this strategy.

> *You can see this is me just literally double checking literally everything too because I know from experience that if you mess like, one little tiny thing up, it's going to affect everything so there, I'm trying to figure out exactly what my bounds are and then trying to remember the asterisk sign is on the keyboard.*
>
> *Anna, Problem 1*

While recalling her previous programming experience, she explained that a small mistake while coding affects the whole program. Hence, she preferred to check and re-check while writing code over fixing errors after executing code.

Self-Reflection

Self-reflection is students' ability to analyze their experiences and think about their own thought process about those experiences (Bandura, 1986). By engaging in self-reflection, people can generate knowledge about themselves and about the world around them (Zimmerman, 2002). In the context of this study, students worked on a thirty-minute programming task, during which they mainly focused on finishing the four programming problems given to them. Students were not prompted to engage in any self-reflection, and they also did not engage in explicit self-reflection about their performance. Moreover, self-reflection is a trait of experts, and novices seldom engage in the process of self-reflection (Zimmerman, 2002).

However, data from the retrospective think-aloud interview showed a few instances of self-reflection that the students engaged in, while working on the programming task. One type of self-reflection is to compare what students are doing with a certain standard, as shown by the following excerpt by Emily.

> *So yeah, I'm just kind of happy that this is kind of related to what we're doing in class so I think I'm pretty confident that if I just go element by element and then, after that the ifelse statement is really easy so I think it's just knowing to go through element by element so I'm like cool, this is what we learned in class so it's nice to be able to apply it.*
>
> *Emily, Problem 1*

Emily explained that one problem on the programming task was related to what she was studying in class, and that made her feel happy and confident. This is a type of self-reflection, called self-evaluation, in which students compare their performance with their previous performance or another standard (Zimmerman, 2005).

CHAPTER 4: DISCUSSION

Research Question 1

What emotions do first-year engineering students experience while they work on a computer programming task, and what reasons do students describe for experiencing the different emotions?

To answer Research Question 1, I used the retrospective think-aloud interview as the primary data source. I analyzed these data using thematic analysis (Braun & Clarke, 2006) and interpreted the findings using the control-value theory of achievement emotions (Pekrun, 2006; Pekrun & Perry, 2014). Overall, during the programming task, the predominant emotions students experienced were frustration (24%), anxiety (10%), confusion (10%), neutral (9%), and relief (9%). All 17 students reported feeling frustrated, while 15 students reported feeling anxious at various times during the programming task. These findings corroborate with findings from literature, which suggests frustration as the most frequent emotion when students worked on programming tasks (Bosch et al., 2013; Lishinski et al., 2017), followed by anxiety and feeling neutral (Bosch et al., 2013). The current study and prior research reveal that the most frequent emotions are neutral and feeling confused (Bosch et al., 2013). However, the control-value theory does not explain confusion and neutral as emotional states (Pekrun & Perry, 2014). The following sections discuss the predominant emotions students experienced during the programming task.

Frustration

All students reported feeling frustrated at various points during the programming task. The overarching events that triggered frustration were when students got stuck somewhere (e.g., unable to overcome an error) or if they failed at something (e.g., they were unable to complete a problem successfully). These events mostly occurred during the "Encountering Difficulties" and "Dealing with Difficulties" stages, instigating frustration in students. Students experienced frustration in all stages except "Getting Started" and "Succeeding," as explained below.

The current study shows that many times, students had negative expectations about the outcome while they were writing code in the "Typing Code" stage. When their negative expectations became a reality, students experienced frustration. The process model in Kinnunen and Simon (2010) does not have a “Typing Code” stage because they could not observe students as they typed code. The current study adds to the process model a new stage which explains student behaviors and emotions as they typed code.

During the “Encountering Difficulties” stage, students experienced frustration when their code did not execute successfully, when they forgot something, or when they did not know how to solve a problem (e.g., problem 3 and running sum from problem 2). In other words, students struggled while working on problems on the programming task. Literature suggests that learning to program involves an interplay of complex cognitive activities, mental models, program design, understanding, modification, and debugging (Guzdial, 2015; Robins et al., 2003). Furthermore, novice programmers do not have the accurate mental models required to learn programming, and they use surface knowledge for problem-solving (Robins et al., 2003). Hence, when the difficulty of a task exceeds the competence of the novice programmers, they may experience negative emotions, like frustration (Pekrun & Perry, 2014).

Most students found problem 3 difficult, as it involved the use of nested loops (called complex loops). Learning to program loops is considered a hard undertaking for novices (Guzdial, 2015; Soloway et al., 1983). Moreover, nested loops were introduced to students just before the Spring break, while the data collection took place immediately after the Spring break. Students may not have had enough practice with nested loops to be able to perform adequately on problem 3, hence, they experienced frustration when trying to write code, or while resolving errors. Findings from this study agree with literature, which suggests that students experience frustration when they encounter difficulties with their code (Bosch & D’Mello, 2015; Kinnunen & Simon, 2010a, 2010b; Lishinski et al., 2017). For instance, according to Kinnunen and Simon (2010), students experience negative emotions like frustration when they encounter errors suddenly when they least expected (as if struck by lightning).

In the “Dealing with Difficulties” stage, students experienced frustration when they were unable to find a solution to their problem despite engaging in trial-and-error and looking at help resources. Findings by Kinnunen and Simon (2010) suggest that students experience negative emotions like despair, puzzlement, and confusion when they try to overcome difficulties while coding. However, the current study shows that students also experienced frustration while dealing with difficulties, which aligns with existing literature that suggests that students experience frustration when they experience persistent failure while writing code (Bosch & D’Mello, 2015).

In the “Stopping” stage, students felt frustrated when they could not understand the problem, or when the programming task ended, and they had failed to solve the problem. For instance, both John and Rachel felt frustrated because they were unable to complete the last problem they were working on. John and Rachel’s experiences align with the peak and end rule (Kahneman, 2000), which states that a person judges their overall experience based on how they felt at the peak of the experience and at the end of the experience. Jack and Rachel were working on a problem when the programming task ended, hence they experienced frustration.

Confusion

In the "Dealing with Difficulties" stage, some students felt both confused and frustrated when dealing with errors in their code. To resolve these errors, students engaged in trial-and-error or looked at help, like students in a previous study by Fitzgerald et al. (2008). Engaging in trial-and-error and looking up help are not mutually exclusive behaviors, and students can switch between these two activities. Students who switch between trial-and-error and help resources can become confused when they are unable to incorporate new information from help into existing code, thus reaching an impasse (Graesser & D’Mello, 2014). In short, students' confusion leads to frustration when they fail to overcome the impasse (Bosch & D'Mello, 2015; Kinnunen & Simon, 2010b, 2012).

Anxiety

Some students experienced anxiety in the “Getting Started,” “Typing Code,” and “Encountering Difficulties” stages mainly because of the negative expectations and uncertainty they had about problems on the programming task. However, in the subsequent stages, students’ anxiety dissipated and changed into either frustration/shame if their negative expectations were fulfilled, or joy/relief if the negative expectations were averted. These findings align with literature, which suggests that prospective anxiety is experienced when there is uncertainty and students focus on anticipated failure (Pekrun & Perry, 2014).

During the “Getting Started” and “Typing Code” stages, incidental events and challenges (e.g., students making spelling mistakes) induced anxiety in students. These incidentals are not related to programming per se, but since students experienced these challenges during the programming task, emotions caused by these incidental challenges may impact students’ performance on the programming task. These findings align with the control-value theory, which suggests that incidental events disrupt students' focus on the task, thus inducing emotions like anxiety (Pekrun & Perry, 2014).

Relief

Students experienced relief primarily when they overcame a challenging situation or when their negative expectations were averted. Students experienced relief in "Getting Started," "Dealing with Difficulties," "Succeeding," and "Stopping" stages. However, students did not experience relief in the "Typing Code" and "Encountering Difficulties" stage, because they were cognitively engaged in completing the task. Experiencing relief in the "Getting Started" stage means that students started the task with apriori expectations, based on their experiences from ENGR 132. When the task started and they realized that the problems were easier that what they get in their class, they experienced relief. However, for many students, the feeling of relief was short lived because they struggled on many various problems on the task. All students experienced relief when the programming task ended (in the "Stopping" stage). Literature suggests that students may have experienced relief because they had completed a cognitively engaging task and they could now focus on other things (Kinnunen & Simon, 2012b).

Pride and Shame

Findings from this study suggest that students experienced pride when they attributed their success to their own efforts, and they experienced shame when they attributed the cause of failure to themselves. These findings align with literature which suggest that shame and pride are self-conscious emotions; when students experience self-conscious emotions, they engage in self-evaluation to understand the reasons for their failure and success (Oades-Sese et al., 2014).

In this study, students experienced pride in "Typing Code", "Dealing with Difficulties", "Succeeding" and "Stopping" stages when they overcame certain challenges and attributed the cause of their success to their own efforts. For instance, some students experienced pride when they successfully executed code on the first attempt. These findings align with literature from computer science education, which suggest that students feel proud when they find a solution to a difficulty based on their own efforts (Kinnunen & Simon, 2012a). It is noteworthy that students experience pride only after they have achieved success in an endeavor. Hence, none of the students experienced pride in the "Getting Started" stage because they had not achieved anything yet, and in the "Encountering Difficulties" stage because they were faced with difficulties.

Students in this study experienced shame in the "Encountering Difficulties," "Dealing with Difficulties," and "Stopping" stages. Students experienced shame most commonly when they were unable to overcome challenges on a seemingly easy problem or after looking up resources. In such cases students attributed the cause of failure to themselves, hence, they experienced shame. For instance, Christina was unable to solve the running sum problem even after looking at help. She explained that she should have been able to solve the problem after looking at help, hence she experienced shame. Although shame is not a frequently reported academic emotion in the context of programming, experiencing shame may seriously hamper student learning and motivation, particularly for performance-oriented students because students tend to blame themselves for their failure (Oades-Sese et al., 2014).

Neutral

Students reported feeling neutral in the "Getting Started" stage, when students were not engaged in tasks that involved high cognitive load. The control-value theory of achievement emotions does not include neutral as an emotion. However, findings from this study conform with previous literature, which has identified neutral as an affective state where students experienced no apparent emotion (Bosch & D'Mello, 2015).

Multiple Emotions

Students experienced multiple emotions simultaneously or in different patterns during the programming task. In some instances, students reported a range of negative emotions (e.g., frustration, anxiety, and hopelessness), while in other instances, students experienced multiple positive emotions like joy and relief. These fluctuation in emotions are important dynamic patterns that illustrate how emotions play out during learning and engagement (Sansone & Thoman, 2005). That is, the dynamic patterns of positive and negative emotions at a given point during the task may be good or bad for learning (Sansone & Thoman, 2005).

Some students also experienced both negative and positive emotions because of the same event. For instance, both Jack and Rachel move to the next problem when they could not solve a problem, intending to come back to it later. However, both students experienced different emotions. While Rachel was hopeful, she could solve the problem when she comes back to it, Jack felt hopeless about the problem. These findings align with literature which suggests that similar experiences could be interpreted differently by different people, yielding different emotional reactions (Barrett, 2006).

Research Question 2

How do student emotions change as a result of working on programming problems?

To answer Research Question 2, the primary data source was students' interview transcripts prompted by their Before- and After-AEQp responses. I analyzed these data using thematic analysis (Braun & Clarke, 2006), and interpreted the findings using the control-value theory of achievement emotions (Pekrun, 2006; Pekrun & Perry, 2014). The secondary data included students' responses on the Before- and After-AEQp, video observations, performance, and mean range-corrected EDA.

The key finding from this research question is that student may experience increases in some emotions and decreases in other emotions in response to the same events. When students experienced decreased enjoyment between the beginning and the end of the task, they also experienced increased anger. Students experience decreased enjoyment and increased anger because they encountered numerous challenges during the programming task. Similarly, when students experienced decreased pride between the beginning and the end of the programming task, they also experienced increased shame, mainly because they fell short of the expectations or standards they had set for themselves. These findings align with literature which suggest that there is a consistent negative correlation between anger and enjoyment, and between pride and shame in academic settings (Pekrun et al., 2011). However, existing correlational research fails to describe the rich descriptions students provided in this study for the increase and decrease in emotions between the beginning and the end of the task.

Many students mentioned feeling frustrated in addition to decreased enjoyment between the beginning and the end of the task. Since this group of students failed at achieving what they set out to do, they experienced frustration. Students' frustration during the programming task are consistent with literature (Bosch & D'Mello, 2015; Kinnunen & Simon, 2012b). However, this study also explained how students' enjoyment decreased because of the frustration they experienced.

Many students in this study had positive expectations and high self-efficacy beliefs before starting the task. When their expectations did not yield positive outcomes, their enjoyment decreased between the beginning and the end of the task. However, findings from this study suggest that there were some exceptions, where students' positive expectations yielded positive outcomes, but they still experienced negative emotions and decreased enjoyment during the task. These students struggled significantly during the programming task and engaged in repeated trial-and-error before finally succeeding on a programming problem.

Findings from this study also suggest that different participants can experience the same event differently. For instance, one student may experience increased pride between the beginning and the end of the task because they were able to complete majority of the problems on the task. However, another student experienced decreased pride because they were unable to complete all the problems on the programming task. Literature suggests that similar experiences could be interpreted differently, with different emotional reactions (Barrett, 2006).

Many times, students experienced negative activating emotions like anger and shame because they attributed the causes to themselves. Students experienced anger and shame when they had set certain standards for themselves or they had positive expectations about their ability to successfully complete the task. These findings align with literature which suggests that people experience anger when they are prevented from reaching their goals (Berkowitz, 2012). Moreover, anger and frustration normally occur together. For instance, students reported feeling frustrated in addition to increased anger during the task. It is likely that students experienced frustration during the programming task, which led to increased anger. These findings agree with the control-value theory, which suggests that activities and outcomes are linked, and are influenced by emotions and the antecedents of emotions (Pekrun & Perry, 2014).

No Change in Emotion

Some students reported no change in some emotions between the beginning and the end of the task. For instance, Emily reported no change in either enjoyment or pride, because she was ambivalent towards programming. She explained that programming was something she did not like, but she also did not dread programming. Jack and Erica did not experience any change in shame at the end of the task, because they thought they had performed adequately considering they were novices and had not had much prior experience with programming. Alternatively, these students may not have experienced a change in emotion because they were working on a low stakes task, with no grade attached to it. Some students who reported no change in emotion also reported feeling neutral during the programming task.

Emotions and Electrodermal Activity

Overall, findings suggest that there were no significant differences in students' range-corrected EDA values between the baseline and the After-AEQp periods. However, the three student cases explained in the findings point to the fact that there are individual differences in how students physiologically respond to a cognitively challenging task like programming. Hence, an in-depth qualitative analysis of individual students' EDA supplemented by their retrospective think-aloud interview responses may provide rich narratives about how students experience the programming task.

Research Question 3

What self-regulation strategies do students use to cope with different emotions during the programming task?

The self-regulated learning framework is a cyclical process that involves three stages (Zimmerman, 2002, 2005; Zimmerman & Campillo, 2003): preparation for the academic task (forethought), engagement with the academic task (performance), and reflection about their performance on the academic task (self-reflection). It is generally accepted that self-regulation is a trait of experts, and novices fail to engage in high-quality self-regulation during academic tasks (Zimmerman, 2002). However, students in this study engaged in some types of self-regulation during the programming task. Students' self-regulation behaviors constituted of minimal planning (if at all) and use of superficial problem-solving strategies. These findings conform with literature, which suggests that while novices engage in some basic form of self-regulation, their self-regulation behaviors do not match self-regulation behaviors of experts (Eteläpelto, 1993; Loksa & Ko, 2016).

Most literature about self-regulated learning focuses on how students regulate their learning during the forethought and self-reflection stages (Schutz & Davis, 2000). There is very little attention on students' self-regulated learning experiences during the performance stage because it is hard to observe students while they work on the task (Schutz & Davis, 2000). This study provides a unique opportunity to understand students' self-regulation as they worked on programming problems.

Forethought

Forethought stage is the preparatory part that students engage in before they start working on the academic task (Schutz & Davis, 2000). In this study, most students began engaging with the programming problems as soon as the task started, without much preparation and planning. The self-regulated learning theory suggests that novices do not engage in high-quality forethought, and hence fail to set specific goals for their learning or performance (Zimmerman, 2002; Zimmerman & Campillo, 2003). Like novices in other studies, students in the current study failed to set goals.

An important aspect of planning is goal setting (Schutz & Davis, 2000; Zimmerman, 2002). Findings from this study suggest that most students did not set any major goal before writing code, because the overarching goal was already set for them, because of the design of the study (complete all four problems on the task). All the students started working on problem 1 as soon as the programming task started. After completing this problem, some students picked a problem they perceived as the easiest. These small sub-goals help students towards the attainment of the overarching goal of completing all four problems in thirty minutes. These findings align with prior work on students' self-regulation during programming tasks, which suggests that novice programmers typically fail to set goals or neglect to monitor their progress toward those goals (Eteläpelto, 1993). These findings are also supported by the self-regulated learning theory, which posits that lack of goal setting is a typical behavior exhibited by novices (Zimmerman, 2002).

In this study, most students showed minimal planning behaviors during the programming task. Most students started writing code for a problem, soon after selecting the problem from the list of problems. However, there is some evidence of planning, when students decided which programming construct to use. The main decision here was between using a programming construct or a MATLAB built-in function to perform a task. These findings align with literature, which suggests that novice programmers do very little pre-planning before they start writing code for a problem (Pea & Kurland, 1984; Robins et al., 2003). In MATLAB, programmers can use not only basic syntax to construct code for a program, they can also use functions from MATLAB libraries to implement their programs. It is obvious that using functions is much quicker and easier than writing code from scratch, but students may be unaware that the desired function already exists in the MATLAB libraries.

Throughout the retrospective think-aloud interviews, students referred to their confidence as their self-efficacy beliefs about a certain problem they were working on. At various points during the retrospective think-aloud interview, students referred to their ability to successfully complete certain problems. Students provided multiple reasons to feel confident about successfully writing code for a problem (e.g., they had done the problem before and they knew they could do it). Students' self-efficacy beliefs during the task fluctuated (Bandura, 1991), depending on how confident they felt about different problems on the task. Consequently, students' emotions also fluctuated throughout the programming task. For instance, if students had high-self-efficacy about a successfully completing a problem, but they were unable to do so, students experienced negative emotions like frustration. Previous studies about students' self-efficacy about programming mainly focused on understanding the relationship between student's self-efficacy beliefs about programming and other constructs, e.g., prior programming experience or gender (Ramalingam, LaBelle, & Wiedenbeck, 2004). However, there is not much literature that understands the relationship between students' task related self-efficacy and students' emotions. Findings from this study points towards the notion that students' self-efficacy beliefs affect their emotions during programming tasks. Hence, an in-depth investigation is required to understand how students' self-efficacy beliefs influence their emotions as they work on programming problems.

Performance

Data from the retrospective interviews showed students' perseverance during the programming task. All students reported negative emotions while working on the task, especially while they encountered errors, or if they got stuck on a problem. Some students reported pushing through the task, even though they experienced negative emotions. This group of students used negative emotions as fuel to persist through the adverse circumstances they experienced. These students gave up only when they could not find any solution to the problem. Second, some students gave up and moved to the next problem, as soon as they realized the problem was too hard, and they would not be able to complete the problem. Literature categorizes these two group of students as “movers” and “stoppers” respectively (Perkins, 2010; Robins et al., 2003).

However, there are differences between how existing literature describes these groups of students and students’ behaviors and emotional reactions in this study. According to Robins et al., (2003), students are likely to become stoppers if they experience negative emotions while writing code. In this study, some students kept pushing through even though they felt frustrated during the programming task. Students' persistence through challenges indicated the positive role that negative emotions can play in students’ learning and motivation. According to the control-value theory, students experience frustration when they fail at an academic task (Pekrun, 2006; Pekrun & Perry, 2014). In this case, most students experienced frustration because they failed at the task (encountered errors that they could not fix), but their reaction to frustration is different. The movers kept pushing through, despite experiencing frustration. Understanding these persistence behaviors may help educators distinguish between students who endeavor to overcome their challenges and those who give up as soon as they encounter difficulty. These findings may be particularly useful to understand students’ long-term persistence in engineering and computing.

Some students deliberately ignored the negative emotions they experienced during the programming task. Students tried to focus on the positive emotions during the task, while ignoring the negative emotions (mostly anxiety). Negative activating emotions like anxiety may be detrimental for learning and performance (Pekrun & Perry, 2014). Hence, students consciously try to ignore their feeling of anxiety, so that they remained on track with the programming problems. Another observation from the findings is that some students used comments to divert their attention from challenging tasks, especially when they experienced negative emotions like anxiety.

Self-reflection

Students who participated in this study did not explicitly engage in self-reflection during the thirty-minute programming task. This finding conforms with literature, which suggests that novices usually do not engage in self-reflection while working on academic tasks (Zimmerman, 2002). However, it must be noted that the programming task was just thirty minutes, and most students focused on completing as many problems as they could in the given timeframe. All students were working on programming problems when the thirty minutes ended, bringing the programming task to an end. Hence, the design of the study did not allow time for self-reflection after the completion of the task. Moreover, before the end of the task, students could have reflected at the end of each problem, but there is no evidence in the retrospective think-aloud interview and video observations.

Emotions and Gender

The focus of this study is to understand a range of emotions students experienced during programming tasks and is not on the comparative difference in which different genders experienced emotions, hence, there are no definitive findings. However, there are some observations from the findings about the motivation and emotions experienced by women in this study.

Lilly and Sarah (both female) reported their dislike for programming (see Lilly and Sarah's excerpts in the "Enjoyment Decreased" sub-section of Research Question 2 in the Findings chapter). Lilly believed she was not good at programming. As a consequence, she did not enjoy the programming task. She doubted her ability and self-efficacy beliefs about programming. Research suggests that women generally have lower self-efficacy beliefs in programming courses than their male counterparts (Askar & Davenport, 2009; Ramalingam & Wiedenbeck, 1998). However, these low self-efficacy beliefs may not be gender related, as shown in Sarah's example. She perceived that she had the ability to do programming (high self-efficacy), but because she experienced hindrances along the way (encountering errors), she was unable to perform as she would have liked. Although Lilly and Sarah experience low and high self-efficacy beliefs respectively, both students experience decreased enjoyment and frustration. Hence, decreased enjoyment and frustration may not be attributed to self-efficacy.

Randy and Christina experienced increased shame at the end of the task because of similar reasons (see Randy and Christina's excerpts in the "Shame Increased" sub-section of Research Question 2 in the Findings chapter). However, Christina also felt stupid at her inability to solve the problem, resulting in her feeling inadequate. Randy on the other hand did not express feeling stupid. Christina's experience conforms with prior literature, which suggests that women generally undermine their programming abilities (Hutchison-Green, Follman, & Bodner, 2008; Ramalingam & Wiedenbeck, 1998).

The Programming Task

The programming task comprised four problems that covered different concepts. Since the problems ranged from easy to difficult, students' performance on these problems ranged from poor to excellent. As a consequence, students experienced a variety of emotions.

Problem 1 required students to write an if/elseif/else/end structure using logical conditions. Students were taught these concepts during the second week of the semester; hence, they had had ample exposure and practice with these concepts. Additionally, most students reported they had seen problem 1 in their ENGR 132 problem sets. Hence, most students achieved the highest score on this problem.

Problem 2 required students to use MATLAB built-in functions to compute the sum, sine, and running sum of a one-dimensional vector. Almost all the students were able to compute sum and sine, but most of the students struggled to compute the running sum, primarily because they had not had practice with the built-in function that computes running sum (cumsum) in their ENGR 132 class. Only four participants were able to find the cumsum function using help and used it correctly, hence receiving a perfect score. The students who received partial credit on this problem could not find the function. Instead, they ended up writing code for computing running sum. None of these students were able to write correct code to calculate the running sum of the vector. Findings from this problem suggest that these students were successful at using built-in functions they were familiar with but struggled to write code for a concept they had not previously seen in class.

Problem 3 required students to used nested loops (called complex loops) to traverse a two-dimensional vector and replace all values less than 0.2 in the vector. Students had studied complex loops just before the Spring break started, and the data collection took place soon after the Spring break ended. Students had not had ample practice with complex loops when the data collection happened. In addition to lack of practice, it is well-documented that loops are hard concepts to master for novices (Soloway et al., 1983). Hence, most of the students preferred to work on problem 4 instead of problem 3. The students who attempted problem 3 struggled significantly and were unable to successfully complete the problem. Problem 4 required students to plot a function. After problem 1 and 2, most students were successful at solving this problem.

It is noteworthy that students experienced syntax errors in their code and did not experience any logical errors, because of two reasons. These students were novices, and novices face challenges while fixing syntax errors (Denny, Luxton-Reilly, & Tempero, 2012). Even if students had resolved syntax errors, they likely introduced logical errors in their code without being aware that they had added more errors in their code (Kohn, 2019). Moreover, during the retrospective think-aloud interview, students mentioned encountering syntax errors and did not mention encountering logical errors. As novices, they are likely to be inexperienced about identifying logical errors (Kohn, 2019; Smith & Rixner, 2019). Students may have introduced logical errors in their code without knowing, because of which they did not mention logical errors during the retrospective think-aloud interview. While grading the scripts, I also did not find any logical errors in the code written by students.

Transferability of Findings

Transferability of findings is an aspect of trustworthiness of a qualitative study (Walther et al., 2017, 2013). For the purpose of discussion, I will discuss four types of transferability: from lab setting to the classroom, across different skill levels, across programming environments, and across other majors and institutions.

Transferability from Lab Setting to the Classroom

There were multiple instances in the data where students connected experiences during the programming task with their actual experiences in their ENGR 132 class, indicating some transferability from the lab study to the classroom. The following excerpt by Anna provides evidence that students who participated in the study connected their experiences in the study with their experiences in the ENGR 132 class.

> *Relieved! That's probably because I know I'm going to have to go and sit down and do it all over again. Just because I have one programming task over, I know that like right now, I have another one to go and sit and work on so it's just like a constant stressor, that you don't really get away from.*
>
> *Anna*

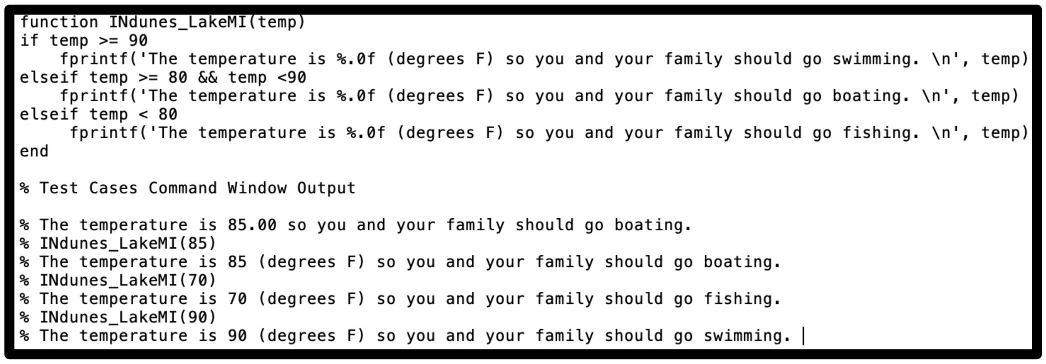
Anna did not feel relieved after the programming task ended, because she knew that programming was not over for her and that she would have to go and do more programming for her class. Although this task was part of a research study and was not related to the ENGR 132 course, she connected the experience of the programming task for this study with the programming she does for the class.

Moreover, Emily alluded to the fact that she treated the lab study as a test. Drawing a parallel, Emily mentioned using the same methods for solving problems that she used in an actual test for her ENGR 132 class, as shown in Emily's excerpt.

> *I mean the nice thing was that it didn't really matter how well I did so I wasn't stressed about. I mean, I still wanted to do well, and I still wanted to figure it out. It was an extremely low stress, low pressure thing but I treated it as a test in the way like, my methods for solving everything were similar so it wasn't like, oh it doesn't matter, I'm not going to care about coding or anything, it's okay I am going to try to do well.*
>
> *Emily*

Although Emily used the same problem-solving strategies as she used in ENGR 132, the lab study differed from an actual test because there was no grade assigned for the lab study, but tests are graded. Hence, findings of this study are transferrable only to a certain extent to a test-like setting. However, more research needs to be conducted in different contexts (e.g., lecture, homework assignments) to understand student emotions in those contexts.

In ENGR 132, when students submit solutions to their problem sets, they are required to copy the output of their code inside their code scripts as evidence that their code executes and that the students executed their code with all the test cases. Some students in this study also copied the output of their code as comments in the code scripts (see Figure 20). These students were not required to copy the output of their code as comments in the scripts, but they chose to work on the programming task as they would on their ENGR 132 problem sets, which alludes to some transferability.



```
function INdunes_LakeMI(temp)
if temp >= 90
    fprintf('The temperature is %.0f (degrees F) so you and your family should go swimming. \n', temp)
elseif temp >= 80 && temp <90
    fprintf('The temperature is %.0f (degrees F) so you and your family should go boating. \n', temp)
elseif temp < 80
     fprintf('The temperature is %.0f (degrees F) so you and your family should go fishing. \n', temp)
end

% Test Cases Command Window Output

% The temperature is 85.00 so you and your family should go boating.
% INdunes_LakeMI(85)
% The temperature is 85 (degrees F) so you and your family should go boating.
% INdunes_LakeMI(70)
% The temperature is 70 (degrees F) so you and your family should go fishing.
% INdunes_LakeMI(90)
% The temperature is 90 (degrees F) so you and your family should go swimming.
```

Figure 20. Code Snippet of Problem 1 by Ella

Transferability Across Different Programming Environments

MATLAB was used in the current study because MATLAB was used in ENGR 132, the class from which we recruited students. MATLAB is significantly different from other programming languages like C or Python (Fangohr, 2004), hence findings from this study might not be transferable to other programming languages (e.g., Java and Python). For instance, MATLAB is an interpreted language, which means that it executes code line by line. In contrast, programming languages like C and Java are compiled, which means that the code executes only after the compiler has ensured that there are no syntax errors in the code. MATLAB requires the programmers to save the scripts in the correct folder before the code is executed, otherwise it generates an error *Undefined function or variable ‹FunctionName›*. Students in this study encountered this error numerous times during the programming task and it took them some time to figure out how to save files in the correct location.

Transferability Across Different Skill Levels

The study focused on understanding emotions of novices as they worked on programming problems. This study also revealed the behaviors novices adopt while self-regulating during a programming task. However, these findings might not be transferable across people with different skill levels (e.g., experts). According to other research, experts normally do not adopt trial-and-error strategies to fix errors in code; they adopt more targeted strategies to fix errors. Experts also do not read or write code line by line. Instead they comprehend code in chunks (Robins et al., 2003). However, there is not much understanding about the emotions experts experience while working on programming. Moreover, there is no research that compares the emotions experienced by novices and experts. Comparative studies about novice and expert emotions and behaviors may provide an understanding on designing different strategies to help novices and experts deal with emotions as they work on programming.

Transferability Across Majors and Institutions

Findings from this study may be transferable to a certain extent to students from other majors and institutions. The following excerpt by George explains potential differences between engineering and computing students and reasons why their emotional experiences may vary.

> *Yeah, I mean I knew that you are not looking for people who are CS majors and have done coding all through high school and you know, so I know you're not expecting me to come in and bang out all four in 10 minutes. So, I didn't have this real high expectations going in and you know, I was able to finish quite a bit. didn't feel bad about any of that.*
>
> *George*

It is interesting that George perceives that CS majors are expected to do well in programming, while engineering majors are not. It is probably a widespread belief among engineering students that programming is meant for CS majors, because of which they are supposed to be very good at it.

It is established that students from different majors and institutions experience negative emotions while working on programming (Bosch et al., 2013; Kinnunen & Simon, 2010b, 2012b; Lishinski et al., 2017). However, it is important to note that students in different majors may have different motivations to take programming course(s). Additionally, different introductory courses might be taught with different instructors, different examples, and different pedagogies. Hence, different students may experience emotions differently. It is thus worth investigating the different emotions that students from different majors and institutions experience.

How Does this Study Advances Literature?

Methodological Contribution

Kinnunen and Simon interviewed students over the course of the semester. Although, they revised their interview protocol, their interviews were unable to fully capture students' experiences of working on the programming problems. Specifically, they acknowledge that their research design did not have the ability to observe students as they worked on programming problem (Kinnunen & Simon, 2010b). Furthermore, the authors also pointed out that their study participants had difficulty recalling their experiences of working on programming problems.

The inability to observe students as they wrote code was addressed by Bosch et al., (2015). In this study, the authors invited students to work on programming problems in two phases. In the learning phase, students worked on programming problems. In the retrospective judgment phase, students assigned emotion labels to their facial expressions during the learning phase at random points. Hence, Bosch et al., (2015) not only neglect data from other parts of students programming experience, but they were also unable to capture descriptive details about students' experiences. The current study used a retrospective think-aloud interview, which enabled me to produce rich descriptions about students' experiences during the programming task, hence, adding nuanced details to the emotion labels students assign to their experiences.

Use of Theory to Inform Different Stages of Research

Previous studies about student emotions in the context of programming have either not used any theory or have used theory superficially. For instance, Kinnunen and Simon used a grounded theory methodology to understand students' experiences while they worked on programming assignments (Kinnunen & Simon, 2010a, 2010b, 2011b, 2012b). Hence, their work is not backed by existing theory. Similarly, Lishinski et al., (2016) used Kinnunen and Simon's work as a base to design their study on student emotions in a programming class. On the other hand, Bosch et al., (2009) provided a list of emotions to the participants to help assign emotion labels to their facial expressions. A unique aspect of the current study is the use of theory to inform the research design, data analysis, and interpretation of findings. The two main theories used are control-value theory of achievement emotions and self-regulated learning (Pekrun & Perry, 2014; Zimmerman, 2002). Most findings from this study are consistent with the claims of control-value theory, that the mechanisms linking emotions to their antecedents and outcomes are universal across individuals, genders, subjects, and cultures (Pekrun & Perry, 2014). However, the current study provides rich descriptions and triangulation with secondary data.

Triangulation Between Multi-modal Data

Various researchers have advocated the use of multi-modal and multi-method paradigm to understand the complexity and richness of emotions in the context of education (Afzal & Robinson, 2015; Pekrun & Linnenbrink-Garcia, 2014a). For this purpose, researchers propose using both qualitative and quantitative data and methods. I followed these calls to take a multi-modal approach to data collection (Afzal & Robinson, 2015; Pekrun & Linnenbrink-Garcia, 2014a) and for triangulation purposes (Borrego, Douglas, & Amelink, 2009).

I designed a novel multi-modal data collection protocol which employed different sources of qualitative (interviews and observations) and quantitative (surveys, EDA, grades) data. The primary data used in this study were students' responses during the retrospective think aloud interview. These data indicate that frustration was students' most prominent emotion. Students reported many reasons for experiencing frustration during the task, but most frequently students experienced frustration when they encountered errors that they were unable to resolve. These findings are confirmed by students' interview responses prompted from students' Before- and After-AEQp responses. For instance, students' responses on these questionnaires showed that for most students' enjoyment decreased between the beginning and the end of the programming task ended. Students also mentioned frustration in addition to decreased enjoyment. I also checked students' reports of emotions by examining segments of the video, a form of observation. Finally, I sometimes used performance data, grades on the submitted scripts, to support students' claims about their emotions. Hence, all these sources of data confirm the occurrence of frustration as a predominant emotion students' experienced.

How this Study Advances Knowledge on Emotions in the Context of Programming

This study advances the literature on emotions in the context of programming in a few ways. There is a dearth of literature that understands emotions of engineering students as they learn programming (Secules et al., 2018). However, other studies have examined students from computer science and other majors (Bosch et al., 2013; Kinnunen & Simon, 2010b, 2012a;

Lishinski et al., 2017). This study provides a more detailed understanding of emotions experienced by first-year engineering students during programming tasks. For instance, previous studies show a negative correlation between pride and shame (Pekrun et al., 2011), but they fail to explain the events that trigger these emotions, and the reasons students describe for experiencing these emotions. In addition, this study examines how students cope with emotions during the performance stage of the self-regulation cycle.

Moreover, most studies on student emotions in programming focus on negative emotions like frustration, anxiety, and confusion (Bosch et al., 2013; Kinnunen & Simon, 2010b, 2012a; Lee et al., 2011; Lishinski et al., 2017; Mercedes, Rodrigo, Shaun, et al., 2009). However, this study identified both positive and negative emotions when students encountered difficulties (relief, pride, anxiety, shame, and hopelessness), which previous research did not mention or described only superficially. For instance, findings from this study explain students' experience of pride, relief, and enjoyment, the reasons for experiencing these positive emotions, and why these emotions change at the beginning and end of the programming task.

CHAPTER 5: CONCLUSION

Summary of Key Findings

I summarize some of the overarching findings from this study in this section. Students reported frustration more frequently than other emotions while working on programming problems. These findings align with literature on emotions in the context of programming. Some students also experienced multiple emotions (e.g., confusion leads to frustration, or frustration and anger occur together), and a mix of positive and negative emotions. For instance, one student reported feeling annoyed because she had made a mistake, but she also experienced joy when she was able to fix the mistake. Most students persevered when they encountered errors. When these students finally overcame the errors, they experienced joy and pride; otherwise, they experienced shame. Finally, students did not display sophisticated expert-like self-regulation, mainly because these students were novices and the programming task was just thirty minutes long.

Implications and Actionable Suggestions for Instructors

In recent years, there has been an increased awareness about the role of emotions in education (Brooks, 2019; Cavanagh, 2016; Swallow, 2018). Positive emotions generally promote learning and creativity, and negative emotions may help during tasks that require close attention (Pekrun & Linnenbrink-Garcia, 2014b). To capture students' attention, promote their learning, and foster their motivation, instructors should incorporate emotional impact in their teaching and curriculum. In particular, instructors teaching programming should foster students' perceptions of control and value of programming. The following section provides some suggestions for instructors. These ideas also align with the control-value theory, which suggests that perceived control over the task and its value influence student emotions (Pekrun & Perry, 2014).

The control-value theory implies that students who have positive values about an academic task also experience positive emotions (Pekrun, 2006; Pekrun & Perry, 2014). Instructors tend to directly or indirectly convey messages that may positively or negatively affect students' value for programming instruction. For instance, many introductory programming or CS1 classes have a reputation of "weed out" classes. Presenting CS1 as a "weed out" class creates a stressful learning environment for the students (Denton & McKinney, 2004), leading students to believe that they are "not cut out" for programming. Instructors must avoid implying that CS1 is a "weed out" course.

Instructors must also realize that their own emotional disposition may transfer to their students in the classroom, an idea called the instructor-to-student contagion (Cavanagh, 2016). For instance, the instructor's positive emotions may lead to greater enthusiasm in delivery. Instructor's enthusiasm creates a pleasant atmosphere in the class, leading to increase in students' feeling of control, subsequently impacting learning.

A shift to a team-oriented approach to instruction and community building may also foster positive emotions in students (Cavanagh, 2016). For this purpose, many instructors have implemented pair programming in their CS1 classes (Brougham et al., 2003). Although pair programming is implemented in ENGR 132, more CS1 instructors need to adopt this technique in their class to promote teamwork. During pair programming, both students provide feedback to each other about their work. Feedback plays an important role in students' learning and growth. Feedback that emphasizes punishment promotes anxiety (Zeidner, 2014). Hence, educators must focus on providing constructive feedback to students, that would foster their learning while promoting their emotional wellbeing.

Findings from this study and previous literature (Bosch & D'Mello, 2015; Kinnunen & Simon, 2012b; Lishinski et al., 2017) suggest that negative emotions like frustration are the most commonly occurring emotions in a programming class. Most students do not share these negative experiences and tend to deal with these emotions in isolation. As a result, students may become discouraged, and subsequently may drop out of the class or engineering. Instructors could periodically conduct live polls in their class asking questions about how students feel while working on programming problems. Based on the results of the poll (e.g., most students experienced frustration and anxiety), instructors could emphasize that most students feel frustrated while programming and that the instructional team is there to help them overcome their feelings of frustration.

Although self-regulation is a trait of experts (Zimmerman, 2002), some self-regulation strategies could be incorporated in a CS1 course to help students become more self-regulated learners. Instructors can promote self-regulation by providing more opportunities for planning by teaching logic building exercises, and by working with pseudocode and flowcharts. Instructors could introduce effective strategies for debugging to improve performance, and incorporate reflection activities after programming tasks to help students reflect on their experiences.

Findings from this study suggest that most students adopted a trial-and-error strategy to debug their code (loop between the "Typing Code," "Encountering Difficulties," and "Dealing with Difficulties" stages in

Figure 13. Sometimes students even engaged in repeated trial-and-error, called "hamster wheel" by Kinnunen & Simon (2011). This debugging behavior is typical of novices because they tend to process and plan in their minds and do not use "paper and pencil" or print statements to debug and trace their code (Murphy et al., 2008), as also evidenced in the current study. Murphy and colleagues suggest that debugging instruction should incorporate metacognition questions where students ask themselves questions like "What should I try?" and "What are the possible sources of bugs?" (Murphy et al., 2008).

Limitations of the Study

We have identified several limitations in methodology and methods, and limitations in data and findings.

Limitations in the Methodology and Methods

It is generally understood that the controlled laboratory setting is less authentic than a higher stakes activities student engage with in a class, e.g., tests, assignments, and homework (Pekrun, 2006). Hence the emotions students experienced in this study may have been less intense than emotions that they experience in real academic settings (Picard, 2016). This limitation cannot be eliminated entirely. However, findings from the study suggest that students experienced negative emotions when faced with programming challenges. Moreover, students regularly connected their experiences in the lab setting with their experiences in the ENGR 132 class. The intensity of students' emotions may increase in a high-stakes task like a programming quiz. However, findings from this study provides valuable insights on how students might behave emotionally when faced with similar tasks in class.

During the retrospective think-aloud interview, students were given a list of emotions to assist them in labeling the emotions that they experienced. The list may have restricted the students to use only the emotions listed provided to them. However, in this study, some students mentioned emotions which were not present on the list (e.g., confusion, overwhelmed).

Some students mentioned during the retrospective think-aloud interview that template scripts are provided to students for pair programming activities in the class. While designing the programming task, I interviewed an ENGR 132 instructor and discussed the level of difficulty of the problems on the task, and how the course is conducted. However, the idea of providing templates to students never came up in the conversation. To simulate the class environment more closely, template scripts could have been provided to the students in the study.

Throughout the study, students mentioned their self-efficacy beliefs about certain problems on the task and about their ability to successfully complete the problems. However, self-efficacy was not explicitly incorporated in the study design. That is, there was no instrument used to measure self-efficacy, and the interview protocol asked no questions about self-efficacy.

In this study, students' responses from the AEQp have been used to ask interview questions about how students' emotions changed as a result of working on the programming task. It is possible that students' interview responses could have been influenced by looking their scores on the Before- and After-AEQp. However, there is sufficient qualitative data that explains students' experience of working on the programming problem, to confirm that their emotions changed between the beginning and the end of the task.

The Before-AEQp and After-AEQp measure the same emotion with different items. For example, the Before-AEQp has five items for enjoyment, and the After-AEQp has only two items for enjoyment. There no evidence that these subscales are directly comparable. My analysis assumes that the average score for enjoyment on the Before-AEQp is directly comparable to the average score for enjoyment on the After-AEQp.

Limitations in the Data and Findings

It must be noted that the participants did not elaborate on change in certain emotions (anxiety, hope and hopelessness) during the retrospective think-aloud interview. Hence changes in these emotions are not explained in the findings. Moreover, some participants found it hard to articulate their emotions and change in emotions. For instance, Erica started out by saying that she was not sure why there was a decrease in her enjoyment after the task ended, but then she articulated the reason. Students also conflate between different emotions. For instance, Mark used “annoyance” and “frustration” interchangeably, and Christina used “shame” and “embarrassment” synonymously.

Engineering population is different from computer science population. This difference is especially important to discuss in the context of learning programming. Computer science students may be more motivated than engineering students to learn programming as they know that they need programming for their future courses and career. On the other hand, engineers may or may not use programming, hence, their motivation to learn programming may not be as high as computer science students. However, it is also important to recognize that engineering students at Purdue University are normally high achievers and self-driven. Data from the study show that some students wanted to take other programming courses in the future, because they realized programming is a key skill that they would need later, or they knew they wanted to pursue careers that require programming skills.

Students in this study encountered syntax errors only, and no logical errors. Hence, this study does not provide any understanding about the emotion’s students experience when they encounter logical errors, what reasons students provide for experiencing those emotions, and how they deal with the emotions and the logical errors. A plausible explanation is that novices normally make syntax errors because they only have superficial understanding and mental models about programming (Secules et al., 2018). However, a more detailed investigation is needed to understand students’ emotions when they encounter and deal with logical errors. In the same vein, there is limited data about students’ behaviors while they executed test cases to test their code. One of the four problems on the programming task provided test cases in the problem description. Students executed the test cases provided to them and did not test code for the other three problems on the task. It may be because they had limited time to finish four problems, and their focus was to finish as many problems as possible, instead of testing the code for multiple test cases.

EDA data has many limitations. First, there are many prerequisites that students need to meet before EDA data collection. For instance, the experiment needs to control for students’ caffeine and medicine intake before collecting EDA data (Villanueva et al., 2018). However, these pre-requisites were not considered, because of the average range-corrected EDA for the baseline was higher than the average range-corrected EDA for the Before-AEQp. Second, a detailed event-based analysis of EDA could not be done because students worked on programming problems in the order of their choice and switched between problems as they deemed fit.

Future Directions

Two types of data from this study remained un-analyzed because of limited time: the short post-task interview conducted immediately after the programming task and the data from the Big-Five neuroticism sub-scale. Hence, the immediate future direction includes analyzing these data. I plan to examine the post-task interview using thematic analysis and corroborate the findings with those explained here. I also want to explore the causal relationship between student emotions in programming and their levels of neuroticism/emotional stability. For this purpose, I want to design a large scale multi-institutional mixed methods study. Data from this study also point towards the relationship between students' self-efficacy beliefs and their emotions. However, self-efficacy was not explicitly measured in this study and hence, no strong claims can be made about the connection between self-efficacy and emotions.

Research in educational neuroscience suggests that cognition, motivation, and emotions are essential components of education. Computing and engineering education researchers have been trying to understand motivation and cognition in the context of learning. However, there is a dearth of research on emotions in computing and engineering education. Hence, I want to extend my dissertation research on emotions in the following directions.

First, I want to investigate how emotions affect student persistence in learning computer programming. For instance, the preliminary findings from my dissertation research suggest that students feel frustrated when they are stuck while solving a programming problem. In this scenario, either they give up and move on to the next problem, or they push through and finish the problem. More research on scenarios like these may help educators design appropriate interventions to help students persist instead of giving up while working on programming problems.

Second, I want to understand women's emotions while they learn programming. The literature suggests that even when women have the same ability as men, they may have lower self-efficacy beliefs because of which they may experience more intense negative emotions than men. Preliminary findings from my dissertation research also suggest that when women experience negative emotions, they tend to doubt their abilities. These negative emotions and low self-efficacy beliefs may subsequently lead them to drop out of computing or engineering altogether. I aim to conduct a detailed investigation of how women’s emotions and self-efficacy beliefs in introductory programming courses can lead them to leave computing or engineering.

Third, I want to investigate emotions of teachers of introductory programming courses. There is very little research on teachers’ emotions, but that research suggests that teachers’ emotions in the classroom may impact their students. In this context I am interested in how teachers mediate the effects of students’ emotions while they learn programming. I am also interested in how teachers’ emotions impact their performance while teaching a programming course.

Fourth, the AEQp was primarily used to prompt interview responses from students. Hence, the questionnaire was not validated using statistical techniques. However, this instrument needs to be validated before significant amounts of quantitative data can be collected using the AEQp.

REFERENCES

Afzal, S., & Robinson, P. (2015). Emotion data collection and its implications for affective computing. In Rafael A. Calvo, S. K. D'Mello, J. Gratch, & A. Kappas (Eds.), *The Oxford Handbook of Affective Computing*. doi:10.1093/oxfordhb/9780199942237.013.002.

Aronson, J. (1995). A pragmatic view of thematic analysis. *The Qualitative Report*, 2(1), 1-3. Retrieved from https://nsuworks.nova.edu/tqr/vol2/iss1/3.

Askar, P., & Davenport, D. (2009). An investigation of factors related to self-efficacy for java programming among engineering students. *Turkish Online Journal of Educational Technology*, *8*(1), 26–32.

Atiq, Z. (2018a). *Emotions experienced by first-year engineering students during programming tasks*. Proceedings of the 2018 ACM Conference on International Computing Education Research, 258–259. doi: https://doi.org/10.1145/3230977.3231014.

Atiq, Z. (2018b). *Work in progress: A multi-modal method for assessing student emotions during programming tasks*. Proceedings of the 2018 ASEE Annual Conference & Exposition, Salt Lake City, Utah. Retrieved from https://peer.asee.org/31264.

Baker, R. S. J. d, Corbett, A. T., Roll, I., & Koedinger, K. R. (2008). Developing a generalizable detector of when students game the system. *User Modeling and User-Adapted Interaction*, *18*(3), 287–314. doi: https://doi.org/10.1007/s11257-007-9045-6.

Bandura, A. (1977). Self-efficacy: Toward a unifying theory of behavioral change. *Psychological Review*, *84*(2), 191–215. doi: https://doi.org/10.1037/0033-295X.84.2.191.

Bandura, A. (1986). *Social foundations of thought and action: A social cognitive theory*. Englewood Cliffs, NJ, US: Prentice-Hall, Inc.

Bandura, A. (1989). Human agency in social cognitive theory. *American Psychologist*, *44*(9), 1175–1184. doi: https://doi.org/10.1037/0003-066X.44.9.1175.

Bandura, A. (1991). Social cognitive theory of self-regulation. *Organizational Behavior and Human Decision Processes*, *50*(2), 248–287. doi: https://doi.org/10.1016/0749-5978(91)90022-L.

Barrett, L. F. (2006). Solving the emotion paradox: Categorization and the experience of emotion. *Personality and Social Psychology Review*, *10*(1), 20–46. doi: https://doi.org/10.1207/s15327957pspr1001_2.

Berkowitz, L. (2012). A different view of anger: the cognitive-neoassociation conception of the relation of anger to aggression. *Aggressive Behavior*, *38*(4), 322–333. doi: https://doi.org/10.1002/ab.21432.

Booth, J. M. J., Doyle, T. E., & Musson, D. M. (2013). *Influence of learning preference on self-efficacy and performance in mixed-modality first-year engineering design*. Proceedings of the Canadian Engineering Education Association (CEEA). École Polytechnique de Montréal.

Borrego, M., Douglas, E. P., & Amelink, C. T. (2009). Quantitative, qualitative, and mixed research methods in engineering education. *Journal of Engineering Education*, *98*(1), 53–66. doi: https://doi.org/10.1002/j.2168-9830.2009.tb01005.x.

Bosch, N., & D'Mello, S. (2015). The affective experience of novice computer programmers. *International Journal of Artificial Intelligence in Education*, 1–26. doi: https://doi.org/10.1007/s40593-015-0069-5.

Bosch, N., D'Mello, S., & Mills, C. (2013). What emotions do novices experience during their first computer programming learning session? In H. C. Lane, K. Yacef, J. Mostow, & P. Pavlik (Eds.), *Artificial Intelligence in Education* (pp. 11–20). doi: https://doi.org/10.1007/978-3-642-39112-5_2.

Boucsein, W. (2012). *Electrodermal activity*. Springer Science & Business Media.

Boulay, B. D. (1986). Some difficulties of learning to program. *Journal of Educational Computing Research*. doi: https://doi.org/10.2190/3LFX-9RRF-67T8-UVK9.

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology*, *3*(2), 77–101. https://doi.org/10.1191/1478088706qp063oa.

Brooks, D. (2019). Opinion | Students Learn from People They Love. *The New York Times*. Retrieved from https://www.nytimes.com/2019/01/17/opinion/learning-emotion-education.html

Brougham, J., Freeman, S., & Jaeger, B. (2003). *Pair programming: More learning and less anxiety in a first programming course*. Proceedings of the American Society of Engineering Education Annual Conference.

Cacioppo, J. T., Tassinary, L. G., & Berntson, G. (2007). *Handbook of Psychophysiology*. Cambridge University Press.

Calvo, R. A., & D'Mello, S. (2010). Affect detection: An interdisciplinary review of models, methods, and their applications. *IEEE Transactions on Affective Computing*, *1*(1), 18–37. doi: https://doi.org/10.1109/T-AFFC.2010.1.

Cavanagh, S. R. (2016). *The spark of learning: Energizing the college classroom with the science of emotion*. West Virginia University Press.

Cheng, K., Husman, J., Fishman, E., & Barnes, M. E. (2015). *Expanding Control-Value Theory: Relations between thinking about the future, positive emotions and biological markers of stress*. Presented at the The Society for Personality and Social Psychology. Retrieved from https://osf.io/3wxcs/.

Connolly, C., Murphy, E., & Moore, S. (2008). Programming anxiety amongst computing students—A key in the retention debate?. *IEEE Transactions on Education*, *52*(1), 52-56. doi: https://doi.org/10.1109/TE.2008.917193.

Cooper S., Forbes J., Fox A., Hambrusch S., Ko A., & Simon B. (2016). The Importance of computing education research: A white paper prepared for the Computing Community Consortium committee of the Computing Research Association. Retrieved from http://arxiv.org/abs/1604.03446.

Creswell, J. W., & Clark, V. L. P. (2011). *Designing and Conducting Mixed Methods Research*. SAGE.

Denny, P., Luxton-Reilly, A., & Tempero, E. (2012). *All syntax errors are not equal.* Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education, 75–80. https://doi.org/10.1145/2325296.2325318.

Denton, L. F., & McKinney, D. (2004). *Affective factors and student achievement: A quantitative and qualitative study*. In 34th Annual Frontiers in Education, 2004. FIE 2004. (pp. T1G-6). IEEE.

Denzin, N. K., & Lincoln, Y. S. (2011). *The SAGE Handbook of Qualitative Research*. SAGE.

D'Mello, S., Picard, R. W., & Graesser, A. (2007). Toward an affect-sensitive AutoTutor. *IEEE Intelligent Systems*, *22*(4), 53-61.

Doyle, E., Stamouli, I., & Huggard, M. (2005, October). *Computer anxiety, self-efficacy, computer experience: An investigation throughout a computer science degree*. In Proceedings Frontiers in Education 35th Annual Conference (pp. S2H-3). IEEE. doi: https://doi.org/10.1109/FIE.2005.1612246.

Elfenbein, H. A., & Ambady, N. (2002). On the universality and cultural specificity of emotion recognition: a meta-analysis. *Psychological Bulletin*, *128*(2), 203–235. doi: https://doi.org/10.1037/0033-2909.128.2.203.

Elling, S., Lentz, L., & de Jong, M. (2011, May). *Retrospective think-aloud method: Using eye movements as an extra cue for participants' verbalizations*. In Proceedings of the SIGCHI conference on human factors in computing systems (pp. 1161-1170). ACM.

Endler, N. S., & Okada, M. (1975). A multidimensional measure of trait anxiety: The SR inventory of general trait anxiousness. *Journal of Consulting and Clinical Psychology*, *43*(3), 319.

Eteläpelto, A. (1993). Metacognition and the expertise of computer program comprehension. *Scandinavian Journal of Educational Research*, *37*(3), 243-254. doi: https://doi.org/10.1080/0031383930370305.

Fangohr, H. (2004, June). *A comparison of C, MATLAB, and Python as teaching languages in engineering*. In International Conference on Computational Science (pp. 1210-1217). Springer, Berlin, Heidelberg.

Fitzgerald, S., Lewandowski, G., McCauley, R., Murphy, L., Simon, B., Thomas, L., & Zander, C. (2008). Debugging: finding, fixing and flailing, a multi-institutional study of novice debuggers. *Computer Science Education*, *18*(2), 93–116. doi: https://doi.org/10.1080/08993400802114508.

Frega, R. (2011). *Pragmatist Epistemologies*. Lexington books.

Frenzel, A. C., Pekrun, R., & Goetz, T. (2007). Girls and mathematics —A "hopeless" issue? A control-value approach to gender differences in emotions towards mathematics. *European Journal of Psychology of Education*, *22*(4), 497. doi: https://doi.org/10.1007/BF03173468.

Graesser, A. C., & D'Mello, S. (2014). Confusion. In R. Pekrun & L. Linnenbrink-Garcia (Eds.), *International Handbook of Emotions in Education*. doi: https://doi.org/10.1080/02667363.2014.994350.

Guzdial, M. (2015). Learner-centered design of computing education: Research on computing for everyone. *Synthesis Lectures on Human-Centered Informatics*, *8*(6), 1-165.

Haak, M. van den, Jong, M. D., & Schellens, P. J. (2003). Retrospective vs. concurrent think-aloud protocols: Testing the usability of an online library catalogue. *Behaviour & Information Technology*, *22*(5), 339–351. doi: https://doi.org/10.1080/0044929031000.

Hawi, N. (2010). Causal attributions of success and failure made by undergraduate students in an introductory-level computer programming course. *Computers & Education*, *54*(4), 1127–1136. doi: https://doi.org/10.1016/j.compedu.2009.10.020.

Husman, J., Cheng, K. C., Puruhito, K., & Fishman, E. J. (2015, June 14). *Understanding engineering students' stress and emotions during an introductory engineering course.* Proceedings of the American Society of Engineering Education Annual Conference.

Hutchison-Green, M. A., Follman, D. K., & Bodner, G. M. (2008). Providing a voice: qualitative investigation of the impact of a first-year engineering experience on students' efficacy Beliefs. *Journal of Engineering Education*, *97*(2), 177–190. doi: https://doi.org/10.1002/j.2168-9830.2008.tb00966.x.

iMotions. (2016). *Facial Expression Analysis: The Complete Pocket Guide*. Retrieved from https://imotions.com/blog/facial-expression-analysis/.

iMotions. (2016). *Galvanic Skin Response (GSR): The Definitive Pocket Guide.* Retrieved from: https://imotions.com/blog/galvanic-skin-response/.

Jack, R., Garrod, O., Yu, H., Caldara, R., & Schyns, P. (2012). Facial expressions of emotion are not culturally universal. *Proceedings of the National Academy of Sciences*, *109*(19), 7241–7244. https://doi.org/10.1073/pnas.1200155109

Jamil, S. J., Saad, S., Jaafar, W. N. W., & Abdullah, N. (2011). *An of investigation of mathematics anxiety among diploma engineering students*. 2011 3rd International Congress on Engineering Education (ICEED), 180–183. doi: https://doi.org/10.1109/ICEED.2011.6235385

Kahneman, D. (2000). Evaluation by Moments: Past and Future. In A. Tversky & D. Kahneman (Eds.), *Choices, Values and Frames*. New York, NY: Cambridge University Press and the Russell Sage Foundation.

Kawulich, B. B. (2005). Participant observation as a data collection method. *Qualitative Social Research*, *6*(2).

Kellam, N. N., Costantino, T., Walther, J., & Sochacka, N. W. (2011, June 26). *Uncovering the role of emotion in engineering education within an integrated curricular experience*. Proceedings of the American Society of Engineering Education Annual Conference.

Kinnunen, P., & Simon, B. (2010a). *Building theory about computing education phenomena*. Proceedings of the 10th Koli Calling International Conference on Computing Education Research - Koli Calling '10, 37–42. doi: https://doi.org/10.1145/1930464.1930469.

Kinnunen, P., & Simon, B. (2010b). *Experiencing programming assignments in CS1: The emotional toll*. Proceedings of the 2010 ACM Conference on International Computing Education Research ICER'10, 77–85. doi: https://doi.org/10.1145/1839594.1839609.

Kinnunen, P., & Simon, B. (2011). *CS Majors ' Self-efficacy perceptions in CS1: Results in light of social cognitive theory*. Proceedings of the 2011 ACM Conference on International Computing Education Research ICER'11, 19–26. doi: https://doi.org/10.1145/2016911.2016917.

Kinnunen, P., & Simon, B. (2012). My program is ok – am I? Computing freshmen's experiences of doing programming assignments. *Computer Science Education*, *22*(1), 1–28. doi: https://doi.org/10.1080/08993408.2012.655091.

Kohn, T. (2019). *The error behind the message: Finding the cause of error messages in Python*. Proceedings of the 50th ACM Technical Symposium on Computer Science Education, 524–530. doi: https://doi.org/10.1145/3287324.3287381.

Lee, D. M. C., Rodrigo, M. M. T., Baker, R. S. J. d, Sugay, J. O., & Coronel, A. (2011). Exploring the relationship between novice programmer confusion and achievement. In S. D'Mello, A. Graesser, B. Schuller, & J. C. Martin (Eds.), *Affective Computing and Intelligent Interaction* (pp. 175–184). doi: https://doi.org/10.1007/978-3-642-24600-5_21.

Leppävirta, J. (2011). The impact of mathematics anxiety on the performance of students of electromagnetics. *Journal of Engineering Education*, *100*(3), 424–443. doi: https://doi.org/10.1002/j.2168-9830.2011.tb00021.x.

Leso, T., & Peck, K. L. (1992). Computer anxiety and different types of computer courses. *Journal of Educational Computing Research*, *8*(4), 469–478. doi: https://doi.org/10.2190/Q1TJ-8JCU-LDAP-84H8.

Lincoln, Y. S., & Guba, E. G. (1985). *Naturalistic inquiry*. SAGE.

Linnenbrink, E. A. (2006). Emotion research in education: Theoretical and methodological perspectives on the integration of affect, motivation, and cognition. *Educational Psychology Review*, *18*(4), 307–314. doi: https://doi.org/10.1007/s10648-006-9028-x.

Lishinski, A., Yadav, A., & Enbody, R. (2017). *Students' emotional reactions to programming projects in introduction to programming: Measurement approach and influence on learning outcomes*. In Proceedings of the 2017 ACM Conference on International Computing Education Research (pp. 30-38). ACM. doi: 10.1145/3105726.3106187.

Lishinski, A., Yadav, A., Good, J., & Enbody, R. (2016). *Learning to program: Gender differences and interactive effects of students' motivation, goals, and self-efficacy on performance*. In Proceedings of the 2016 ACM Conference on International Computing Education Research (pp. 211-220). ACM. doi: 10.1145/2960310.2960329.

Loksa, D., & Ko, A. J. (2016, August). *The role of self-regulation in programming problem solving process and success*. In Proceedings of the 2016 ACM Conference on International Computing Education Research (pp. 83-91). ACM. doi: 10.1145/2960310.2960334.

Marshall, B., Cardon, P., Poddar, A., & Fontenot, R. (2013). Does sample size matter in qualitative research?: A review of qualitative interviews in IS research. *Journal of Computer Information Systems*, *54*(1), 11–22. doi: https://doi.org/10.1080/08874417.2013.11645667.

Mauss, I. B., & Robinson, M. D. (2009). Measures of emotion: A review. *Cognition and Emotion*, *23*(2), 209–237. doi: https://doi.org/10.1080/02699930802204677.

May, G. S., & Chubin, D. E. (2003). A retrospective on undergraduate engineering success for underrepresented minority students. *Journal of Engineering Education*, *92*(1), 27–39. doi: https://doi.org/10.1002/j.2168-9830.2003.tb00735.x.

McCrae, R. R., & John, O. P. (1992). An introduction to the five-factor model and its applications. *Journal of personality*, *60*(2), 175-215. doi: https://doi.org/10.1111/j.1467-6494.1992.tb00970.x.

Rodrigo, M. M. T., Baker, R. S., Jadud, M. C., Amarra, A. C. M., Dy, T., Espejo-Lahoz, M. B. V., Lim, S. A. L., Pascua, S. A. M. S., Sugay, J. O., & Tabanao, E. S. (2009, July). *Affective and behavioral predictors of novice programmer achievement*. In ACM SIGCSE Bulletin (Vol. 41, No. 3, pp. 156-160). ACM. doi: 10.1145/1595496.1562929.

Mercedes, M., Rodrigo, T., Shaun, R., & Baker, J. (2009). *Coarse-grained detection of student frustration in an introductory programming course.* ICER ’09 Proceedings of the Fifth International Workshop on Computing Education Research Workshop. New York, NY. doi: 10.1145/1584322.1584332.

Mercedes, M., Rodrigo, T., Sugay, J. O., Baker, R. S., & Tabanao, E. (2009). *Monitoring Novice Programmer Affect and Behaviors to Identify Learning Bottlenecks*. In Philippine Computing Society Congress (pp. 1-7).

Meyer, M., & Marx, S. (2014). Engineering propouts: A qualitative examination of why undergraduates leave engineering. *Journal of Engineering Education*, *103*(4), 525–548. doi: https://doi.org/10.1002/jee.20054.

Morse, J. M. (2000). Determining sample size. *Qualitative Health Research*, *10*(1), 3–5. SAGE.

Oades-Sese, G. V., Matthews, T. A., & Lewis, M. (2014). Shame and Pride and their Effects on Student Achievement. In R. Pekrun & L. Linnenbrink-Garcia (Eds.), *International Handbook of Emotions in Education*. doi: https://doi.org/10.1080/02667363.2014.994350.

Ocumpaugh, J., Baker, R. S., & Rodrigo, Ma. M. T. (2015). *Baker Rodrigo Ocumpaugh Monitoring Protocol (BROMP) Training Manual 2.0*. New York, NY: Teachers College, Columbia University. Manila, Philippines: Ateneo Laboratory for the Learning Sciences.

Owolabi, J., Olanipekun, P., & Iwerima, J. (2014). Mathematics ability and anxiety, computer and programming anxieties, age and gender as determinants of achievement in basic programming. *GSTF Journal on Computing (JoC)*, *3*(4).

Pajares, F. (1996). Self-efficacy beliefs in academic settings. *Review of Educational Research*, *66*(4), 543–578. doi: https://doi.org/10.3102/00346543066004543.

Patton, M. Q. (2014). *Qualitative Research & Evaluation Methods: Integrating Theory and Practice* (4 edition). Thousand Oaks, California: SAGE Publications, Inc.

Paulhus, D. L., & Vazire, S. (2009). The self-report method. In R. W. Robins, R. C. Fraley, & R. F. Krueger (Eds.), *Handbook of Research Methods in Personality Psychology* (pp. 224–239). Guilford Press.

Pea, R. D., & Kurland, D. M. (1984). On the cognitive effects of learning computer programming. *New Ideas in Psychology*, *2*(2), 137–168. doi: https://doi.org/10.1016/0732-118X(84)90018-7.

Pekrun, R. (2006). The control-value theory of achievement emotions: Assumptions, corollaries, and implications for educational research and practice. *Educational Psychology Review*, *18*(4), 315–341. doi: https://doi.org/10.1007/s10648-006-9029-9.

Pekrun, R., Frenzel, A. C., Goetz, T., & Perry, R. P. (2007). The Control-Value Theory of Achievement Emotions: An Integrative Approach to Emotions in Education. In P. A. Schutz & R. Pekrun (Eds.), *Emotion in Education* (pp. 13–36). doi: https://doi.org/10.1016/B978-012372545-5/50003-4.

Pekrun, R., Goetz, T., Frenzel, A. C., Barchfeld, P., & Perry, R. P. (2011). Measuring emotions in students' learning and performance: The Achievement Emotions Questionnaire (AEQ). *Contemporary Educational Psychology*, *36*(1), 36–48. doi: https://doi.org/10.1016/j.cedpsych.2010.10.002.

Pekrun, R., Goetz, T., Titz, W., & Perry, R. P. (2002). Academic emotions in students' self-regulated learning and achievement: A program of qualitative and quantitative research. *Educational Psychologist*, *37*(2), 91–105. doi: https://doi.org/10.1207/S15326985EP3702_4.

Pekrun, R., Götz, T., & Perry, R. P. (2005). Achievement emotions questionnaire (AEQ). User's manual. *Department of Psychology, University of Munich, Munich, Germany*.

Pekrun, R., & Linnenbrink-Garcia, L. (2014a). Conclusions and future directions. In R. Pekrun & L. Linnenbrink-Garcia (Eds.), *International Handbook of Emotions in Education*. doi: https://doi.org/10.1080/02667363.2014.994350.

Pekrun, R., & Linnenbrink-Garcia, L. (2014b). Introduction to emotions and education. In R. Pekrun & L. Linnenbrink-Garcia (Eds.), *International Handbook of Emotions in Education*. doi: https://doi.org/10.1080/02667363.2014.994350.

Pekrun, R., & Perry, R. P. (2014). Control-value theory of achievement emotions. In R. Pekrun & L. Linnenbrink-Garcia (Eds.), *International handbook of emotions in education* (pp. 120–141). doi: https://doi.org/10.1080/02667363.2014.994350.

Perkins, D. (2010). *Making Learning Whole: How Seven Principles of Teaching Can Transform Education*. John Wiley & Sons.

Picard, R. W. (2016). Automating the recognition of stress and emotion: From lab to real-world impact. *IEEE MultiMedia*, *23*(3), 3–7. doi: 10.1109/MMUL.2016.38.

Pintrich, P. R., & Schunk, D. H. (2001). *Motivation in Education: Theory, Research, and Applications* (2 edition). Upper Saddle River, N.J: Prentice Hall.

Polya, G. (1988). *How to Solve It: A New Aspect of Mathematical Method* (Princeton Science Li edition). Princeton N.J.: Princeton University Press.

Ramalingam, V., LaBelle, D., & Wiedenbeck, S. (2004, June). *Self-efficacy and mental models in learning to program*. In ACM SIGCSE Bulletin (Vol. 36, No. 3, pp. 171-175). ACM.

Ramalingam, V., & Wiedenbeck, S. (1998). Development and validation of scores on a computer programming self-efficacy scale and group analyses of novice programmer self-efficacy. *Journal of Educational Computing Research*, *19*(4), 367–381. doi: https://doi.org/10.2190/C670-Y3C8-LTJ1-CT3P.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, *13*, 137–172.

Rogerson, C., & Scott, E. (2010). The fear factor: How it affects students learning to program in a tertiary environment. *Journal of Information Technology Education: Research*, *9*(1), 147–171.

Sansone, C., & Thoman, D. B. (2005). Does what we feel affect what we learn? Some answers and new questions. *Learning and Instruction*, *15*(5), 507–515. doi: https://doi.org/10.1016/j.learninstruc.2005.07.015.

Scherer, K. R. (2009). The dynamic architecture of emotion: Evidence for the component process model. *Cognition and Emotion*, *23*(7), 1307–1351. doi: https://doi.org/10.1080/02699930902928969.

Schukajlow, S., Rakoczy, K., & Pekrun, R. (2017). Emotions and motivation in mathematics education: theoretical considerations and empirical contributions. *ZDM*, *49*(3), 307–322. doi: https://doi.org/10.1007/s11858-017-0864-6.

Schunk, D. H. (1991). Self-efficacy and academic motivation. *Educational Psychologist*, *26*(3–4), 207–231. doi: https://doi.org/10.1080/00461520.1991.9653133.

Schutz, P. A., & Davis, H. A. (2000). Emotions and self-regulation during test taking. *Educational Psychologist*, *35*(4), 243–256. doi: https://doi.org/10.1207/S15326985EP3504_03.

Scott, M., & Ghinea, G. (2014). *Measuring enrichment: The assembly and validation of an instrument to assess student self-beliefs in CS1*. Proceedings of the Tenth Annual Conference on International Computing Education Research. doi: 10.1145/2632320.2632350.

Secules, S., Elby, A., & Gupta, A. (2016, June 26). *"Turning away" from the struggling individual student: An account of the cultural construction of engineering ability in an undergraduate programming class*. Proceedings of the American Society of Engineering Education Annual Conference.

Secules, S., Gupta, A., Elby, A., & Turpen, C. (2018). Zooming out from the struggling individual student: An account of the cultural construction of engineering ability in an undergraduate programming class. *Journal of Engineering Education*, *107*(1), 56–86.

Setz, C., Arnrich, B., Schumm, J., Marca, R. L., Tröster, G., & Ehlert, U. (2010). Discriminating stress from cognitive load using a wearable EDA device. *IEEE Transactions on Information Technology in Biomedicine*, *14*(2), 410–417. doi: https://doi.org/10.1109/TITB.2009.2036164.

Smith, R., & Rixner, S. (2019). The Error Landscape: *Characterizing the Mistakes of Novice Programmers*. Proceedings of the 50th ACM Technical Symposium on Computer Science Education, 538–544. doi: https://doi.org/10.1145/3287324.3287394.

Soloway, E., Bonar, J., & Ehrlich, K. (1983). Cognitive strategies and looping constructs: An empirical study. *Communications of the ACM*, *26*(11), 853–860.

Sun, W., & Sun, X. (2011, June 26). *Teaching computer programming skills to engineering and technology students with a modular programming strategy*. Proceedings of the American Society of Engineering Education Annual Conference.

Suresh, R. (2006). The relationship between barrier courses and persistence in engineering. *Journal of College Student Retention: Research, Theory & Practice*, *8*(2), 215–239. doi: https://doi.org/10.2190/3QTU-6EEL-HQHF-XYF0.

Swallow, J. R. (2018). Students today need colleges to value emotions as well as the intellect. *Inside Higher Ed*. Retrieved from https://www.insidehighered.com/views/2018/07/10/students-today-need-colleges-value-emotions-well-intellect-opinion.

Tufford, L., & Newman, P. (2012). Bracketing in qualitative research. *Qualitative Social Work*, *11*(1), 80–96. doi: https://doi.org/10.1177/1473325010368316.

Villanueva, I., Raikes, A., Ruben, N., Schaefer, S., & Günther, J. (2014). *The use of physiological tools to identify changes in affective responses for graduate students recently admitted into a scientific discipline*. 2014 IEEE Frontiers in Education Conference (FIE) Proceedings, 1–5. doi: https://doi.org/10.1109/FIE.2014.7044316.

Villanueva, I., Campbell, B. D., Raikes, A. C., Jones, S. H., & Putney, L. G. (2018). A multimodal exploration of engineering students' emotions and electrodermal activity in design activities. *Journal of Engineering Education*, *107*(3), 414–441. doi: https://doi.org/10.1002/jee.20225.

Villanueva, I., Valladares, M., & Goodridge, W. (2016). Use of galvanic skin responses, salivary biomarkers, and self-reports to assess undergraduate student performance during a laboratory exam activity. *JoVE (Journal of Visualized Experiments)*, (108), e53255–e53255. doi: https://doi.org/10.3791/53255.

Visnovcova, Z., Mestanik, M., Gala, M., Mestanikova, A., & Tonhajzerova, I. (2016). The complexity of electrodermal activity is altered in mental cognitive stressors. *Computers in Biology and Medicine*, *79*, 123–129. doi: https://doi.org/10.1016/j.compbiomed.2016.10.014.

Walther, J., Sochacka, N. W., Benson, L. C., Bumbaco, A. E., Kellam, N., Pawley, A. L., & Phillips, C. M. (2017). Qualitative research quality: A collaborative inquiry across multiple methodological perspectives. *Journal of Engineering Education*, *106*(3), 398–430.

Walther, J., Sochacka, N. W., & Kellam, N. N. (2013). Quality in interpretive engineering education research: Reflections on an example study. *Journal of Engineering Education*, *102*(4), 626–659. doi: https://doi.org/10.1002/jee.20029.

Weiner, B. (1972). Attribution Theory, Achievement Motivation, and the Educational Process. *Review of Educational Research*, *42*(2), 203–215. doi: https://doi.org/10.3102/00346543042002203.

Wiedenbeck, S., Labelle, D., & Kain, V. N. (2004). *Factors affecting course outcomes in introductory programming*. In Proceedings of the 16th Workshop of the Psychology of Programming Interest Group. Carlow, Ireland.

Wladis, C., Hachey, A. C., & Conway, K. M. (2014). The representation of minority, female, and non-traditional STEM majors in the online environment at community colleges A nationally representative study. *Community College Review*, 0091552114555904. doi: https://doi.org/10.1177/0091552114555904.

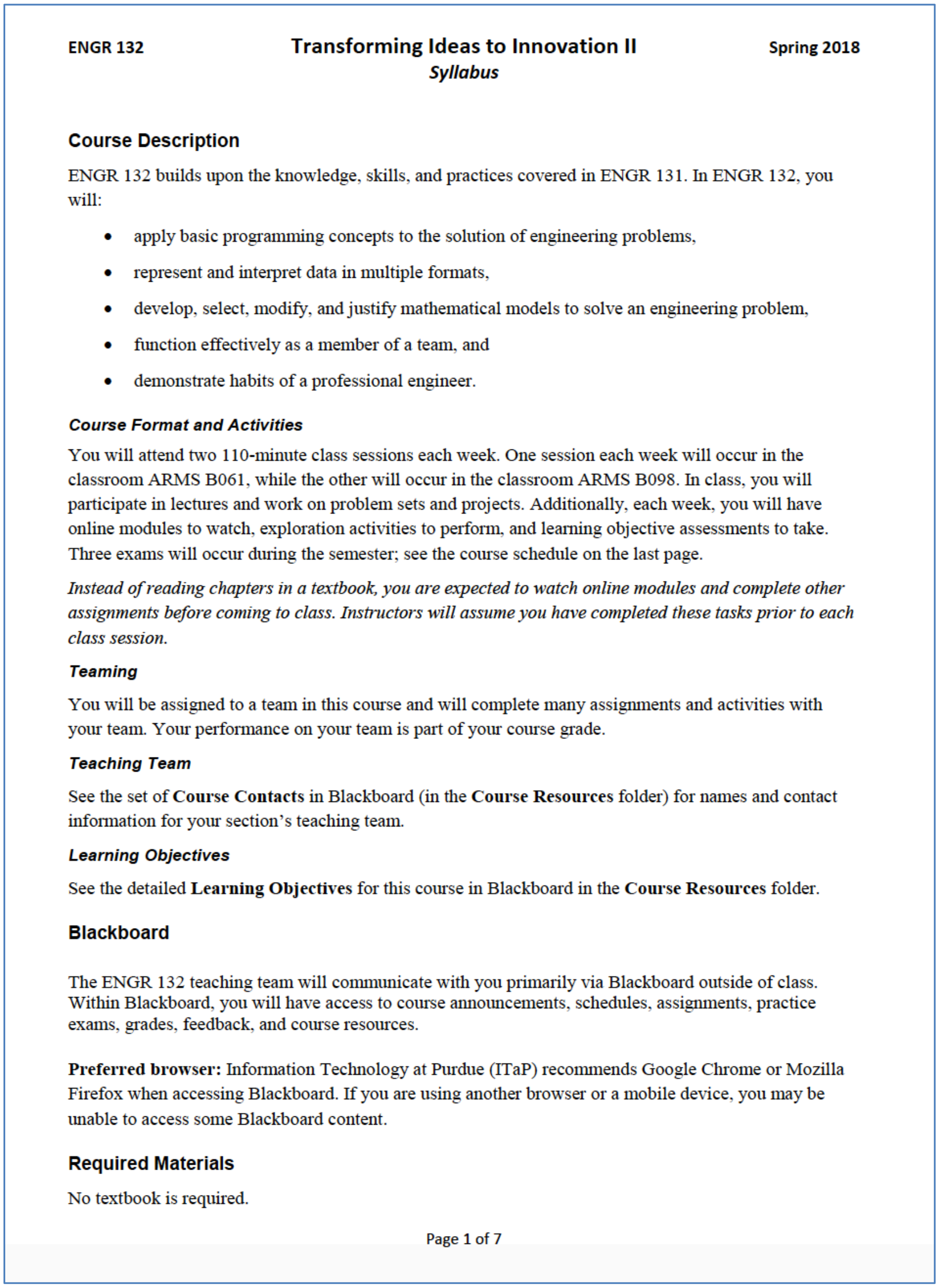
Zeidner, M. (2014). Anxiety in education. In *International handbook of emotions in education* (pp. 265–288). In R. Pekrun & L. Linnenbrink-Garcia (Eds.), *International handbook of emotions in education* (pp. 120–141). doi: https://doi.org/10.1080/02667363.2014.994350. Routelage.

Zimmerman, B. J. (2002). Becoming a self-regulated learner: An Overview. *Theory into Practice*, *41*(2), 64–72.

Zimmerman, B. J. (2005). Attaining self-regulation: A social cognitive perspective. In M. Boekaerts, P. R. Pintrich, & M. Zeidner (Eds.), *Handbook of Self-Regulation*.

Zimmerman, B. J., & Campillo, M. (2003). Motivating self-regulated problem solvers. In J. E. Davidson & R. J. Sternberg (Eds.), *The Psychology of Problem Solving*. Cambridge University Press.

APPENDIX A: SYLLABUS OF ENGR 132 (SPRING 2018)



ENGR 132 **Transforming Ideas to Innovation II** Spring 2018

Syllabus

Course Description

ENGR 132 builds upon the knowledge, skills, and practices covered in ENGR 131. In ENGR 132, you will:

- apply basic programming concepts to the solution of engineering problems,
- represent and interpret data in multiple formats,
- develop, select, modify, and justify mathematical models to solve an engineering problem,
- function effectively as a member of a team, and
- demonstrate habits of a professional engineer.

Course Format and Activities

You will attend two 110-minute class sessions each week. One session each week will occur in the classroom ARMS B061, while the other will occur in the classroom ARMS B098. In class, you will participate in lectures and work on problem sets and projects. Additionally, each week, you will have online modules to watch, exploration activities to perform, and learning objective assessments to take. Three exams will occur during the semester; see the course schedule on the last page.

Instead of reading chapters in a textbook, you are expected to watch online modules and complete other assignments before coming to class. Instructors will assume you have completed these tasks prior to each class session.

Teaming

You will be assigned to a team in this course and will complete many assignments and activities with your team. Your performance on your team is part of your course grade.

Teaching Team

See the set of **Course Contacts** in Blackboard (in the **Course Resources** folder) for names and contact information for your section's teaching team.

Learning Objectives

See the detailed **Learning Objectives** for this course in Blackboard in the **Course Resources** folder.

Blackboard

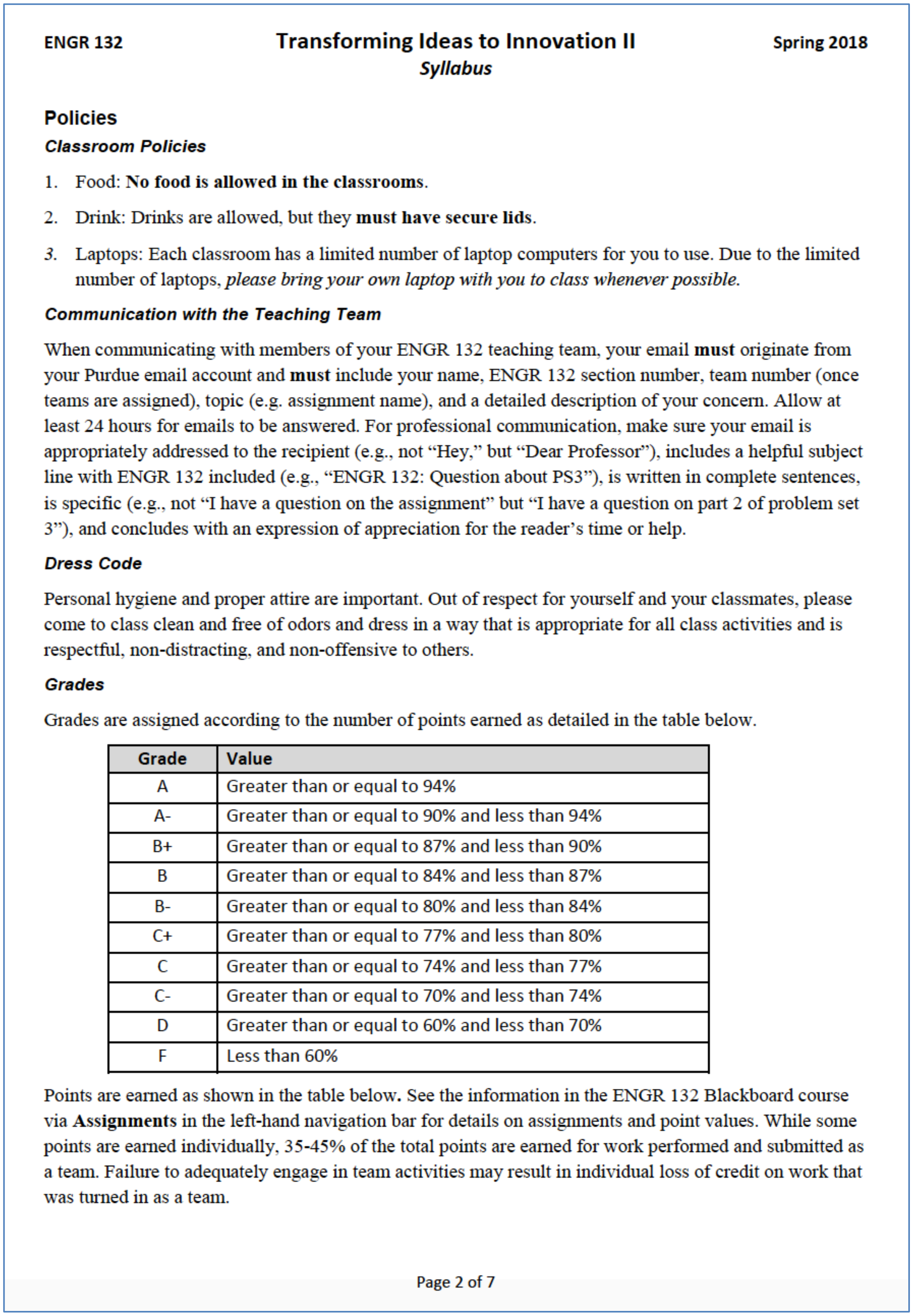
The ENGR 132 teaching team will communicate with you primarily via Blackboard outside of class. Within Blackboard, you will have access to course announcements, schedules, assignments, practice exams, grades, feedback, and course resources.

Preferred browser: Information Technology at Purdue (ITaP) recommends Google Chrome or Mozilla Firefox when accessing Blackboard. If you are using another browser or a mobile device, you may be unable to access some Blackboard content.

Required Materials

No textbook is required.

Page 1 of 7



Policies

Classroom Policies

1. Food: **No food is allowed in the classrooms**.
2. Drink: Drinks are allowed, but they **must have secure lids**.
3. Laptops: Each classroom has a limited number of laptop computers for you to use. Due to the limited number of laptops, *please bring your own laptop with you to class whenever possible.*

Communication with the Teaching Team

When communicating with members of your ENGR 132 teaching team, your email **must** originate from your Purdue email account and **must** include your name, ENGR 132 section number, team number (once teams are assigned), topic (e.g. assignment name), and a detailed description of your concern. Allow at least 24 hours for emails to be answered. For professional communication, make sure your email is appropriately addressed to the recipient (e.g., not "Hey," but "Dear Professor"), includes a helpful subject line with ENGR 132 included (e.g., "ENGR 132: Question about PS3"), is written in complete sentences, is specific (e.g., not "I have a question on the assignment" but "I have a question on part 2 of problem set 3"), and concludes with an expression of appreciation for the reader's time or help.

Dress Code

Personal hygiene and proper attire are important. Out of respect for yourself and your classmates, please come to class clean and free of odors and dress in a way that is appropriate for all class activities and is respectful, non-distracting, and non-offensive to others.

Grades

Grades are assigned according to the number of points earned as detailed in the table below.

Grade	Value
A	Greater than or equal to 94%
A-	Greater than or equal to 90% and less than 94%
B+	Greater than or equal to 87% and less than 90%
B	Greater than or equal to 84% and less than 87%
B-	Greater than or equal to 80% and less than 84%
C+	Greater than or equal to 77% and less than 80%
C	Greater than or equal to 74% and less than 77%
C-	Greater than or equal to 70% and less than 74%
D	Greater than or equal to 60% and less than 70%
F	Less than 60%

Points are earned as shown in the table below**.** See the information in the ENGR 132 Blackboard course via **Assignments** in the left-hand navigation bar for details on assignments and point values. While some points are earned individually, 35-45% of the total points are earned for work performed and submitted as a team. Failure to adequately engage in team activities may result in individual loss of credit on work that was turned in as a team.

Category	Total Points
Learning Objective Assessments	50
Problem Sets	120
Project	300
Exams	360
Team Member Effectiveness	150
Miscellaneous	20
Total	**1000**

To pass ENGR 132 and move on to an engineering school program, your final grade in ENGR 132 must be a C or better (i.e., at least 700 total points). Further, you must also demonstrate sufficient mastery of the first two course goals at the beginning of this syllabus by earning at least 50% of the possible points on the Exams (individual portion) and the Problem Sets, combined.

Grading

The purpose of grading is to assess your understanding and utilization of the concepts taught in the course, and to provide you with feedback about the strengths and weaknesses evident in your work. Full credit may be awarded on items that are mostly correct even if the work still contains errors in understanding. Therefore, it is important that you not only check your score on a particular assignment or exam, but also review the feedback provided by the graders. This feedback will help you improve your understanding of the concepts being assessed and, in turn, improve your performance on future work.

Concerns About Grading

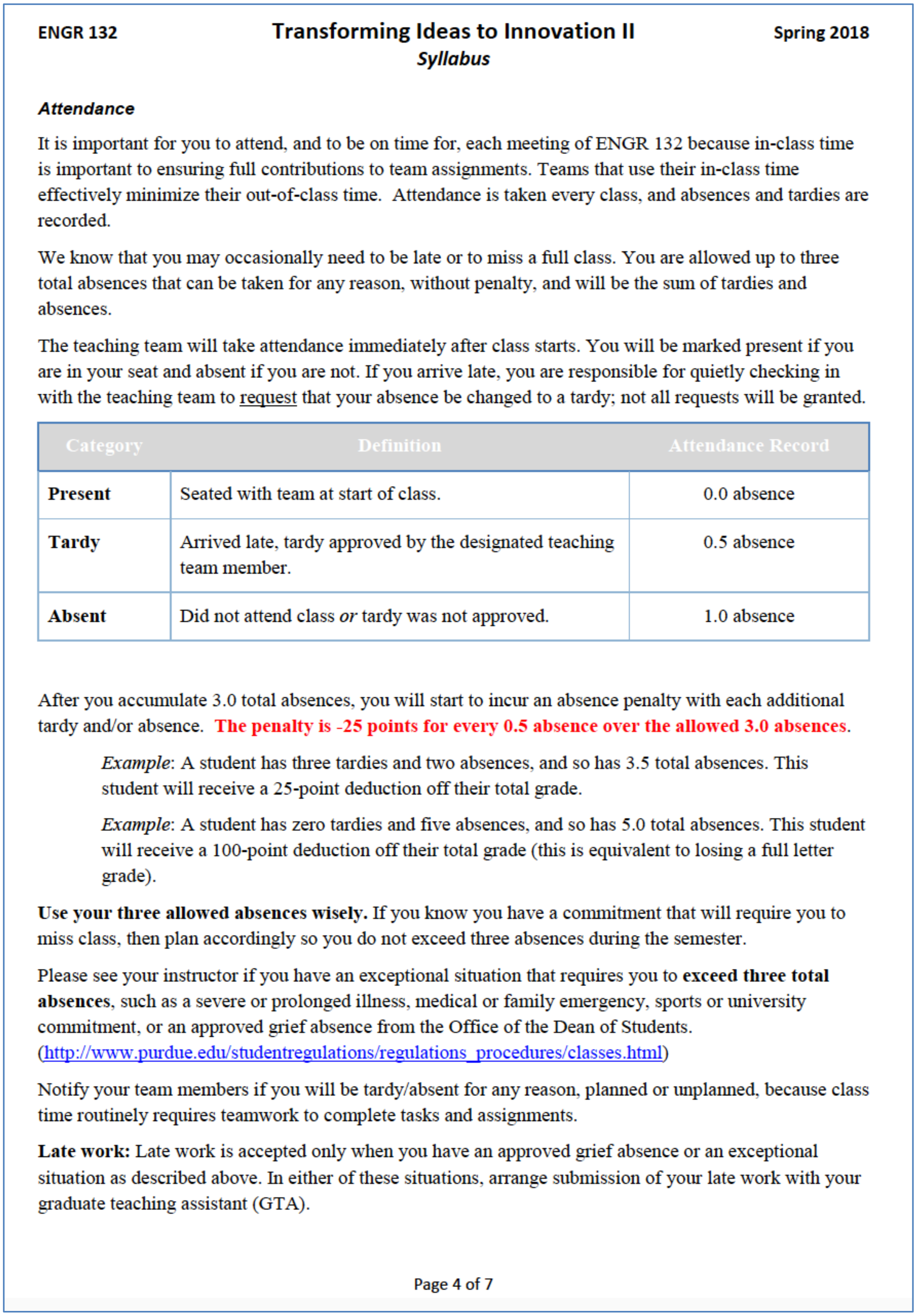
If you have concerns about how an assignment was graded, send an email to your graduate teaching assistant (GTA) with a detailed description of the concern within **seven days** after the graded assignment was revealed in Blackboard. Please see ***Communication with the Teaching Team*** (above) for proper email etiquette.

Professional Expectations

Professional expectations in ENGR 132 reinforce the idea that everyone in our learning environment helps shape the environment to be positive and productive for everyone. Behaving professionally includes arriving for class on time and being prepared; focusing during class on ENGR 132; controlling your behavior to minimize distractions to those around; and engaging with others in a respectful and professional manner. **Therefore, the instructional team will deduct points from your semester total for behavior that is disruptive to your class or to your team's dynamics and performance. Example deductions follow:**

Category	Deduction
Classroom Issues	• −5 for sleeping in class • −5 for using a computer or a cell phone for non-class purposes • −5 for behavior that is disruptive to the class, the team, or an inclusive classroom environment • Up to −25 for observed behavior disruptive to team dynamics/performance

Note: Additional *Classroom Issues* may be added. Your instructor will notify you of these.



Attendance

It is important for you to attend, and to be on time for, each meeting of ENGR 132 because in-class time is important to ensuring full contributions to team assignments. Teams that use their in-class time effectively minimize their out-of-class time. Attendance is taken every class, and absences and tardies are recorded.

We know that you may occasionally need to be late or to miss a full class. You are allowed up to three total absences that can be taken for any reason, without penalty, and will be the sum of tardies and absences.

The teaching team will take attendance immediately after class starts. You will be marked present if you are in your seat and absent if you are not. If you arrive late, you are responsible for quietly checking in with the teaching team to request that your absence be changed to a tardy; not all requests will be granted.

Category	Definition	Attendance Record
Present	Seated with team at start of class.	0.0 absence
Tardy	Arrived late, tardy approved by the designated teaching team member.	0.5 absence
Absent	Did not attend class *or* tardy was not approved.	1.0 absence

After you accumulate 3.0 total absences, you will start to incur an absence penalty with each additional tardy and/or absence. **The penalty is -25 points for every 0.5 absence over the allowed 3.0 absences**.

Example: A student has three tardies and two absences, and so has 3.5 total absences. This student will receive a 25-point deduction off their total grade.

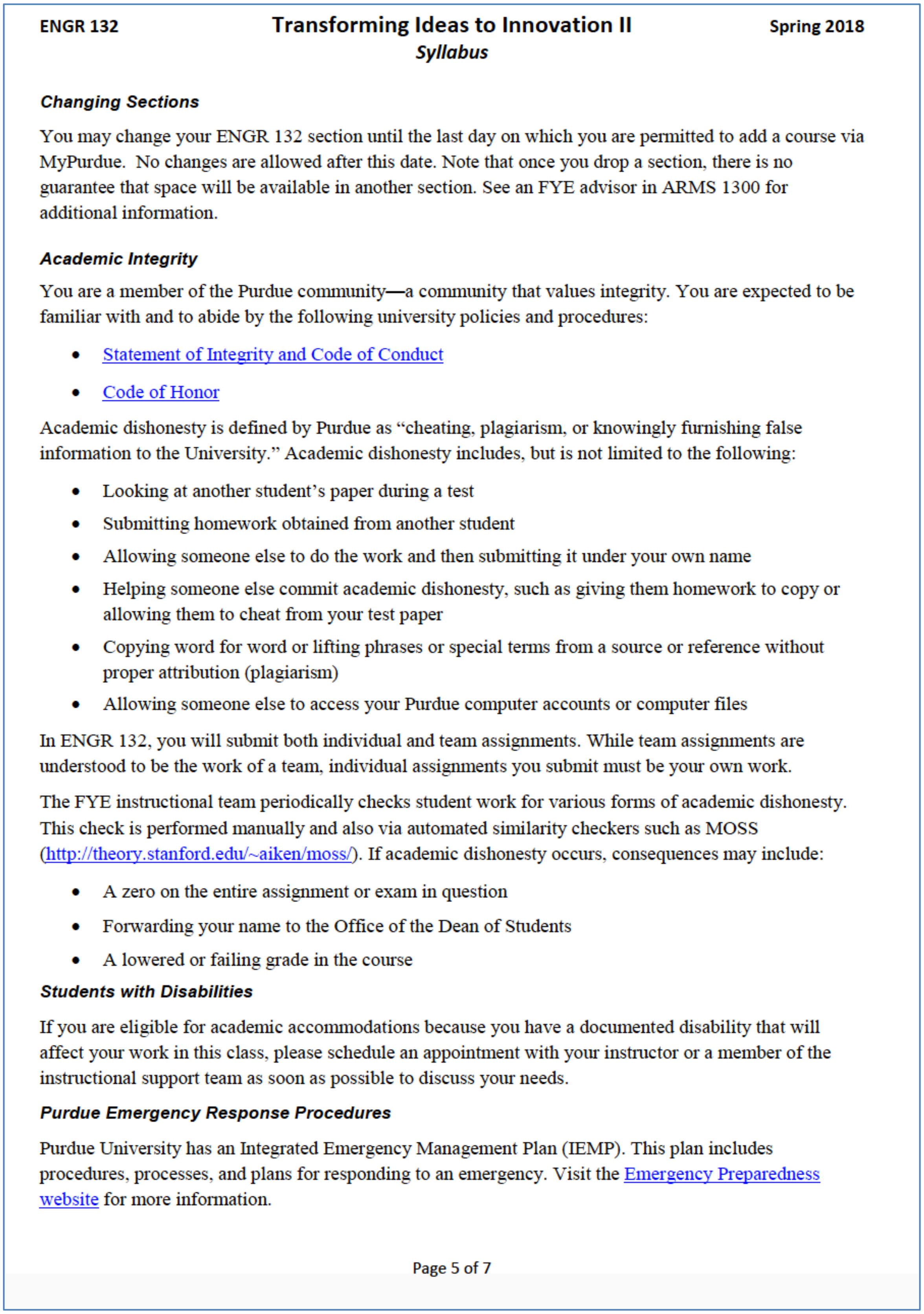
Example: A student has zero tardies and five absences, and so has 5.0 total absences. This student will receive a 100-point deduction off their total grade (this is equivalent to losing a full letter grade).

Use your three allowed absences wisely. If you know you have a commitment that will require you to miss class, then plan accordingly so you do not exceed three absences during the semester.

Please see your instructor if you have an exceptional situation that requires you to **exceed three total absences**, such as a severe or prolonged illness, medical or family emergency, sports or university commitment, or an approved grief absence from the Office of the Dean of Students. (http://www.purdue.edu/studentregulations/regulations_procedures/classes.html)

Notify your team members if you will be tardy/absent for any reason, planned or unplanned, because class time routinely requires teamwork to complete tasks and assignments.

Late work: Late work is accepted only when you have an approved grief absence or an exceptional situation as described above. In either of these situations, arrange submission of your late work with your graduate teaching assistant (GTA).



Changing Sections

You may change your ENGR 132 section until the last day on which you are permitted to add a course via MyPurdue. No changes are allowed after this date. Note that once you drop a section, there is no guarantee that space will be available in another section. See an FYE advisor in ARMS 1300 for additional information.

Academic Integrity

You are a member of the Purdue community—a community that values integrity. You are expected to be familiar with and to abide by the following university policies and procedures:

- Statement of Integrity and Code of Conduct
- Code of Honor

Academic dishonesty is defined by Purdue as "cheating, plagiarism, or knowingly furnishing false information to the University." Academic dishonesty includes, but is not limited to the following:

- Looking at another student's paper during a test
- Submitting homework obtained from another student
- Allowing someone else to do the work and then submitting it under your own name
- Helping someone else commit academic dishonesty, such as giving them homework to copy or allowing them to cheat from your test paper
- Copying word for word or lifting phrases or special terms from a source or reference without proper attribution (plagiarism)
- Allowing someone else to access your Purdue computer accounts or computer files

In ENGR 132, you will submit both individual and team assignments. While team assignments are understood to be the work of a team, individual assignments you submit must be your own work.

The FYE instructional team periodically checks student work for various forms of academic dishonesty. This check is performed manually and also via automated similarity checkers such as MOSS (http://theory.stanford.edu/~aiken/moss/). If academic dishonesty occurs, consequences may include:

- A zero on the entire assignment or exam in question
- Forwarding your name to the Office of the Dean of Students
- A lowered or failing grade in the course

Students with Disabilities

If you are eligible for academic accommodations because you have a documented disability that will affect your work in this class, please schedule an appointment with your instructor or a member of the instructional support team as soon as possible to discuss your needs.

Purdue Emergency Response Procedures

Purdue University has an Integrated Emergency Management Plan (IEMP). This plan includes procedures, processes, and plans for responding to an emergency. Visit the Emergency Preparedness website for more information.

ENGR 132 Classroom Emergency Response

Emergency: For ANY emergency, call 911 (fire, medical emergency, etc.).

- ARMS B061: Phone is in the corner of the room by the podium.
- ARMS B098: Phone is in the internal hallway by the printer.

Fire Alarm or Evacuation: Gather all critical personal belongings and exit the building using the stairs. Do not use the elevator.

- ARMS B061: Proceed up the stairs by Amelia's Cafe.
- ARMS B098: Proceed up the stairs by B122 (away from Amelia's).

Meet in the emergency assembly area in the grass between Civil Engineering and the Johnson Hall of Nursing.

Shelter in Place: Could occur due to tornado, accidental release of toxic chemicals, shots fired on campus, etc.

- **Tornado:** Stay in the classroom. In B098, move away from any glass and sit on the floor.
- **Other situations:** The course of action will depend upon the situation; FYE has an extensive emergency plan that will be put into action.
- **Recommendation:** Remain in the classroom and wait for further instructions.

Note: *In any situation, follow instructions from emergency response personnel (police, fire department, etc.) when they are present.*

In the event of a major campus emergency, course requirements, deadlines and grading percentages are subject to changes that may be necessitated by a revised semester calendar or other circumstances beyond the instructor's control. Information about any changes will be available in Blackboard.

Nondiscrimination

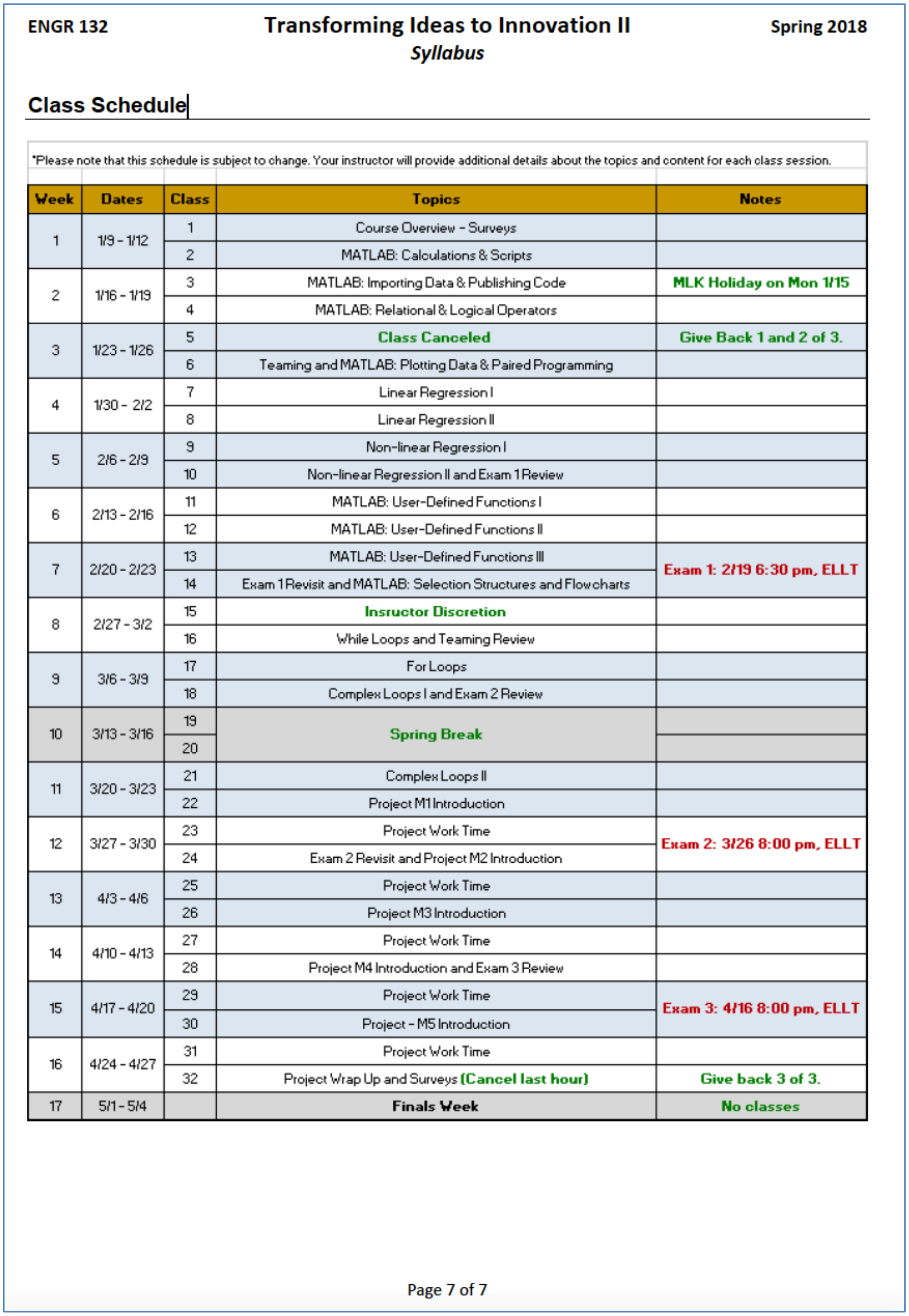
In this course, each voice in the classroom has something of value to contribute. Please take care to respect the different experiences, beliefs and values expressed by students and staff involved in this course. We support Purdue's commitment to diversity, and welcome individuals of all ages, backgrounds, citizenships, disability, sex, education, ethnicities, family statuses, genders, gender identities, geographical locations, languages, military experience, political views, races, religions, sexual orientations, socioeconomic statuses, and work experiences.

For additional information about diversity and nondiscrimination at Purdue, visit the following websites:

- Purdue nondiscrimination policy statement
- Purdue Division of Diversity & Inclusion

Purdue Policies

You are expected to comply with all policies on the University Policies website.



ENGR 132 **Transforming Ideas to Innovation II** **Spring 2018**

Syllabus

Class Schedule

*Please note that this schedule is subject to change. Your instructor will provide additional details about the topics and content for each class session.

Week	Dates	Class	Topics	Notes
1	1/9 - 1/12	1	Course Overview - Surveys	
		2	MATLAB: Calculations & Scripts	
2	1/16 - 1/19	3	MATLAB: Importing Data & Publishing Code	**MLK Holiday on Mon 1/15**
		4	MATLAB: Relational & Logical Operators	
3	1/23 - 1/26	5	**Class Canceled**	**Give Back 1 and 2 of 3.**
		6	Teaming and MATLAB: Plotting Data & Paired Programming	
4	1/30 - 2/2	7	Linear Regression I	
		8	Linear Regression II	
5	2/6 - 2/9	9	Non-linear Regression I	
		10	Non-linear Regression II and Exam 1 Review	
6	2/13 - 2/16	11	MATLAB: User-Defined Functions I	
		12	MATLAB: User-Defined Functions II	
7	2/20 - 2/23	13	MATLAB: User-Defined Functions III	**Exam 1: 2/19 6:30 pm, ELLT**
		14	Exam 1 Revisit and MATLAB: Selection Structures and Flowcharts	
8	2/27 - 3/2	15	**Insructor Discretion**	
		16	While Loops and Teaming Review	
9	3/6 - 3/9	17	For Loops	
		18	Complex Loops I and Exam 2 Review	
10	3/13 - 3/16	19	**Spring Break**	
		20		
11	3/20 - 3/23	21	Complex Loops II	
		22	Project M1 Introduction	
12	3/27 - 3/30	23	Project Work Time	**Exam 2: 3/26 8:00 pm, ELLT**
		24	Exam 2 Revisit and Project M2 Introduction	
13	4/3 - 4/6	25	Project Work Time	
		26	Project M3 Introduction	
14	4/10 - 4/13	27	Project Work Time	
		28	Project M4 Introduction and Exam 3 Review	
15	4/17 - 4/20	29	Project Work Time	**Exam 3: 4/16 8:00 pm, ELLT**
		30	Project - M5 Introduction	
16	4/24 - 4/27	31	Project Work Time	
		32	Project Wrap Up and Surveys **(Cancel last hour)**	**Give back 3 of 3.**
17	5/1 - 5/4		**Finals Week**	**No classes**

Page 7 of 7

APPENDIX B: RECRUITMENT EMAILS

Recruitment email

Subject: Request to participate in a research study "Emotions Experienced by First-Year Engineering Students during Programming Tasks"
Dear Students,
I am conducting a research study titled "Emotions Experienced by First-Year Engineering Students during Programming Tasks" for my Ph.D. dissertation. For this study, I am interested in understanding the emotions that students experience while they work on programming problems, the reasons for experiencing those emotions, and the strategies they adopt to deal with those emotions. For this purpose, I aim to recruit a total of 20 students during the Spring 2018 semester. I am aiming to recruit domestic students who have not taken ENGR 132 before, have NO prior programming experience of any kind, and are NOT currently enrolled in ANY OTHER programming course (e.g., CS 159). If you meet these criteria and are interested in participating in the study, please visit this link and sign-up: http://bit.ly/emotions_mar18. You will be required to give your consent before taking a background survey. The survey should take no more than 5 minutes.
If you are selected, you will be invited to participate in two sessions: 1) In the first session, you would work on a programming task in order to assess emotions (about 75 minutes). 2) Three to five days after performing the programming task, you will be invited to participate in a retrospective interview (about 60 minutes). You will be compensated for your time and effort. I will give you a $40 Amazon gift card after the completion of the study. For any questions, you can contact me at satiq@purdue.edu.
Thank you. I look forward to seeing you soon.
Zahra Atiq,
Ph.D. Candidate and Research Assistant,
School of Engineering Education, Purdue University.

Invitation to selected participants

Subject: Appointment details for the emotions research study
Dear <NAME>,
Thank you for volunteering to participate in my Ph.D. dissertation study ("Emotions Experienced by First-Year Engineering Students During Programming Tasks"). Please meet me at the front desk in WANG 3500 on <DATE AND TIME>. I will send you a reminder before the actual meeting. If you have any questions before then, feel free to contact me at satiq@purdue.edu or 7654913807. I look forward to seeing you in a few days.
Thank you and Regards
Zahra Atiq
Ph.D. Candidate and Research Assistant
School of Engineering Education, Purdue University.

APPENDIX C: BACKGROUND INFORMATION SURVEY

Purdue IRB Protocol #: 1712019990 - Expires on: 03-JAN-2019

What is this study about? My name is Zahra Atiq and I am a Ph.D. candidate at the School of Engineering Education at Purdue University. For my dissertation research, I am studying the emotions that first-year engineering students experience while they work on programming tasks. Subsequently, I am also hoping to understand the self-regulation strategies that students adopt to deal with those emotions. Since emotions influence learning and success, the findings of this study will help educators mitigate the effects of negative emotions and promote the effects of positive emotions on student learning and success. You are eligible to participate only if you are age 18 or older.

What will I be asked to do if I am in this study? You will be requested to come for two sessions. The first session will be 75 minutes and the second session will be 60 minutes. For the first session, you will work on a programming task, which consists of four programming problems. Before, during, and after this task, you will be prompted to respond to a few questions about the emotions you may be experiencing at that moment. Finally, you will be interviewed about your experiences with the programming task. For the second session, you will be invited to come for a retrospective think-aloud interview 3 – 5 days after the first session. During this interview, I will review the data collected during the first session. You will be asked about the emotions that you may have experienced while working on the programming problems. This session will last for almost one hour.

What types of data will be collected via this study? You will use a workstation that consists of a computer with a front camera and eye-tracker installed on it. This camera will record your facial expressions while you work on the problems, and the eye-tracker will record your eye-gaze. A screen capture software utility will record your work on the screen. You will be wearing a non-invasive device on your left foot. This device will record your body's electrodermal activity (EDA), which consists of the autonomic changes in the electrical properties of the skin. I am also providing scratch paper and pencil, in case you need it for scratch work. The scratch paper will be collected after you have completed the programming task. The interviews will be audio recorded. All sources of data will be used to assess the emotions that you may feel at a certain point in time.

Are there any benefits to me if I am in this study? There are no direct benefits to the participants of the study. However, by participating in this study, you will have a chance to reflect on your emotional experiences while working on programming tasks, the reasons for experiencing those emotions, and the self-regulation strategies you adopted to deal with those emotions. These reflections may help you understand your own emotions better and choose appropriate self-regulation strategies when needed.

Are there any risks to me if I am in this study? You may experience minor psychological or emotional discomfort. In the first session, you will be asked to work on a programming task, which may cause positive or negative emotions. However, the programming tasks require the knowledge that you have already learned. Hence, it is expected that the emotional discomfort will be minimal. Moreover, the tasks are low-stakes because they will not affect your course grade. During the second session, you will be asked to reflect on the emotions you experienced during the first programming session. Reflecting on your experiences may also cause minor discomfort.

<u>Are there any costs or payments for being in this study?</u> You will receive a $40 Amazon gift card via email. To receive the Amazon gift card, you must complete both sessions. By hitting the next button, you give consent to share your background information with the research team. For any questions, please do not hesitate to contact me: satiq@purdue.edu.

Q1 First Name

____

Q2 Last Name

____

Q3 Age

____

Q4 Major OR Intended Major

____

Q5 Email

____

Q6 Cell phone Number (Optional)

____

Q7 You are:
A Domestic Student
An International Student
Q8 Are you taking any other programming course while taking ENGR 132? If so, which one?
Yes ____
No
Q9 Are you repeating ENGR 132?
Yes
No
Q10 What gender do you identify with?
Male
Female
Prefer not to say
Other ____
Q11 What race/ethnicity do you identify with?
American Indian or Alaskan native
Asian
Black or African American
Native Hawaiian or other Pacific Islander
White
Hispanic or Latino/a
Multicultural
Other ____
Q12 What was your grade on the last homework of ENGR 132 (State in percentage)?

Q13 How many years of programming experience did you have before starting ENGR 132?
None
0 - 1 years
More than 1 year
Q14 What technology-related experiences have you had before starting ENGR 132 (choose all that apply)?
Build your own car (e.g., Super Mileage Challenge)
Robotics summer camp
AP Computer Science
Wrote/Manipulated macros in excel
Website development
Other ____
Q15 Please select all the times that are suitable for you. I will then email you with the time and venue of the appointment.
Monday March 19th, 2018 2:00 PM - 3:30 PM
Monday March 19th, 2018 4:00 PM - 5:30 PM
Monday March 19th, 2018 6:00 PM - 7:30 PM
Tuesday March 20th, 2018 8:00 AM - 9:30 AM
Tuesday March 20th, 2018 10:00 AM - 11:30 AM
Tuesday March 20th, 2018 2:00 PM - 3:30 PM
Tuesday March 20th, 2018 4:00 PM - 5:30 PM
Tuesday March 20th, 2018 6:00 PM - 7:30 PM
Wednesday March 21st, 2018 10:00 AM - 11:30 AM
Wednesday March 21st, 2018 12:00 PM - 1:30 PM
Wednesday March 21st, 2018 2:00 PM - 3:30 PM
Wednesday March 21st, 2018 4:00 PM - 5:30 PM
Wednesday March 21st, 2018 6:00 PM - 7:30 PM
Thursday March 22nd, 2018 8:00 AM - 9:30 AM
Thursday March 22nd, 2018 10:00 AM - 11:30 AM
Thursday March 22nd, 2018 12:00 PM - 1:30 PM
Thursday March 22nd, 2018 2:00 PM - 3:30 PM
Thursday March 22nd, 2018 4:00 PM - 5:30 PM
Friday March 23rd, 2018 12:00 PM - 1:30 PM
Friday March 23rd, 2018 2:00 PM - 3:30 PM
Saturday March 24th, 2018 10:00 AM - 11:30 AM
Saturday March 24th, 2018 12:00 PM - 1:30 PM
Saturday March 24th, 2018 2:00 PM - 3:30 PM
Saturday March 24th, 2018 4:00 PM - 5:30 PM
Saturday March 24th, 2018 6:00 PM - 7:30 PM
Monday March 26th, 2018 2:00 PM - 3:30 PM
Monday March 26th, 2018 4:00 PM - 5:30 PM
Monday March 26th, 2018 6:00 PM - 7:30 PM
Tuesday March 27th, 2018 8:00 AM - 9:30 AM
Tuesday March 27th, 2018 10:00 AM - 11:30 AM
Tuesday March 27th, 2018 2:00 PM - 3:30 PM

Tuesday March 27th, 2018 4:00 PM - 5:30 PM
Tuesday March 27th, 2018 6:00 PM - 7:30 PM
Wednesday March 28th, 2018 8:00 AM - 9:30 AM
Wednesday March 28th, 2018 10:00 AM - 11:30 AM
Wednesday March 28th, 2018 12:00 PM - 1:30 PM
Wednesday March 28th, 2018 2:00 PM - 3:30 PM
Wednesday March 28th, 2018 4:00 PM - 5:30 PM
Wednesday March 28th, 2018 6:00 PM - 7:30 PM
Friday March 30th, 2018 8:00 AM - 9:30 AM
Friday March 30th, 2018 10:00 AM - 11:30 AM
Friday March 30th, 2018 12:00 PM - 1:30 PM
Friday March 30th, 2018 2:00 PM - 3:30 PM
Other (Please mention other time slots in which you are available and if you get selected, I will accommodate you accordingly) ____

APPENDIX D: ACHIEVEMENT EMOTIONS QUESTIONNAIRE

Before-, and after-task AEQp

I am using the original test sub-scale of AEQ Most of the items in the test sub-scale have the terms *'test'* or *'exam'*, which have been replaced by *'the programming task'*. For instance, *'I look forward to the exam'* has been changed to *'I look forward to the programming task'*. In the Qualtrics survey, all the emotion labels (e.g., Task Enjoyment) will be removed.

Before-task AEQp

The following questions pertain to feelings you may experience BEFORE starting a programming task. Please indicate how you are feeling, before starting the programming task.

1 – Strong Disagree to 5 – Strongly Agree

		1	2	3	4	5
	Task Enjoyment					
1.	I am looking forward to the programming task					
2.	I am looking forward to demonstrating my knowledge					
3.	Because I enjoy preparing for the programming task, I'm motivated to do more than is necessary					
4.	Because I look forward to being successful, I have studied hard					
5.	Before starting a programming task, I sense a feeling of eagerness					
	Task Hope					
6.	I am optimistic that everything will work out fine					
7.	I have great hope that my abilities will be sufficient					
8.	I am quite confident that my preparation is sufficient					
9.	I am thinking about the programming task optimistically					
10.	I have studied for the programming task with great hope and anticipation					
11.	My confidence motivates me to prepare well					
	Task Pride					
12.	I am so proud of my preparation that I want to start the programming task now					
	Task Anger					
13.	I get angry over time pressures which don't leave enough time to prepare					
14.	I get angry about the amount of material I need to know					
	Task Anxiety					
15.	Before the programming task, I am feeling nervous and uneasy					
16.	I am worried whether I have studied enough					
17.	I am worried whether the programming task will be too difficult					
18.	I am so nervous I wish I could just skip the programming task					

19.	I am feeling sick to my stomach					
Task Shame						
20.	I can't even think about how embarrassing it would be to fail the programming task					
Task Hopelessness						
21.	I am feeling depressed because I feel I don't have much hope for the programming task					
22.	I have lost all hope that I have the ability to do well on the programming task					
23.	I am feeling so resigned about the programming task that I can't start doing anything					
24.	I'd rather not work on the programming task because I have lost all hope					
25.	My hopelessness is robbing me of all my energy					

After-task AEQp

The following questions pertain to feelings you may experience AFTER completing the programming task. Please indicate how you feel, typically after completing a programming task.

1 – Strong Disagree to 5 – Strongly Agree

		1	2	3	4	5
Task Enjoyment						
1.	My heart is beating faster with joy					
2.	I am glowing all over					
Task Pride						
3.	I am very satisfied with myself					
4.	I am proud of myself					
5.	Thinking about my success is making me feel proud					
6.	I am proud at how well I mastered the programming task					
7.	When my program ran correctly, my heart beats with pride					
8.	After the programming task I am feeling ten feet taller because I'm so proud					
9.	I am walking out of the programming task with the look of a winner on my face					
Task Relief						
10.	I feel relief					
11.	I feel freed					
12.	I am feeling very relieved					
13.	The tension in my stomach is dissipated					
14.	I finally can breathe easy again					
15.	I can finally laugh again					
Task Anger						
16.	I am fairly annoyed					

17.	I am angry because my program had errors					
18.	I wish I could yell at the compiler/interpreter					
19.	I wish I could freely express my anger					
20.	My anger is making the blood rush to my head					
21.	I am feeling so angry, I am feeling hot and flushed					
Task Shame						
22.	I am feeling ashamed					
23.	My performance on programming tasks is embarrassing me					
24.	When my program fails, I would prefer not to face my teacher again					
25.	When others find out about my poor programming ability, I start to blush					

APPENDIX E: PROTOCOL FOR THE PROGRAMMING TASK AND POST-TASK INTERVIEW

Total time: 75 minutes

- Introduction and signing the consent form: 5 - 10 minutes
- Programming task, Background information + before-, during-, and after-task Achievement Emotions Questionnaire (AEQp): 30 + 10 minutes
- Follow-up interview: 25 minutes

Introduction to the study, rapport building, and signing of consent form (5 – 10 minutes).

First of all, I want to thank you for participating in this study. Let me first give you a brief overview of this research study. I am interested in understanding the emotions students experience while they work on programming problems and the reasons for experiencing those emotions. Subsequently, I am also interested in the coping strategies students adopt to deal with those emotions. Since, emotions influence learning and success, the findings of this study will help educators mitigate the effects of negative emotions, and promote the effects of positive emotions on student learning and success.

This session is divided into two parts:

1. Programming Session (40 minutes)

1. You will be given about 3 minutes to complete the before-AEQp. In case you complete the questionnaire before time, please press F3 on the keyboard and I will move you on to the next screen.
2. You will then be prompted to start working on the programming task (four programming problems) for the next 30 minutes. The problems range from easy to hard, in terms of difficulty levels. You may chose to perform the problems in order, or select any problem you may want to work on. You should attempt to complete as many problems as you can during this time. You will also be provided with some hints, in case you feel stuck on a problem.
3. After 15 minutes of working on the programming problems, you will be prompted to answer a set of questions about the emotions you may be experiencing during the programming session. Answering these questions should take you about 3 minutes or less to complete. You will then resume the programming task and will continue working on it for the next 15 minutes.
4. After you have completed the task, you will be prompted to complete the after-AEQp, which should take you about 3 minutes or less to complete.
5. Finally, you will be asked to sign-up for the retrospective think-aloud interview, which will take place 3 – 5 days after completing this session.

You will use a workstation which consists of a computer with a front camera and eye-tracker installed on it. This camera will capture your facial expressions while you work on the problems, and the eye-tracker will capture your eye gaze. A screen capture software will also capture the work that you perform on the screen. You will be wearing a non-invasive device on one foot. This device will capture your electrodermal activity (EDA); that is, the device measures changes in the electrical properties of the skin. All sources of data will be used to assess the emotions you may be feeling at a certain point in time. You will be provided with scratch paper and pencil, if needed, which will be collected on completion of the programming task.

2. Post-Task Interview (25 minutes)

Now that you have worked on the given programming problems, I want to ask you some follow-up questions regarding your emotional experiences while you were working on the problems. The interview will be semi-structured in nature, which means that I may ask you probing questions, and you may add additional comments if deemed necessary.

General Questions/Examples

1. You were given 4 problems to work on. How many problems do you think were you able to solve successfully?
2. What are some challenges you encountered while working on the problems?
    a. How did the challenges make you feel?
    b. What are some of the events which are making you feel this way?
    c. Could you tell me how you attempted to overcome the challenges? Tell me more/Give a more specific example.
    d. How do you think these emotions will impact the future programming tasks that you are likely to work on? On your overall well-being?
3. Please explain what you do when you encounter a hard programming problem [Probe: Do you give up or work only on the easy parts?]
4. Please explain what you do when you find programming problems dull or uninteresting [Probe: Do you keep working on it until you finish? (Yes/No → Why?)]
5. Before you begin working on a programming problem, what are some of the things you think you will need to complete the programming problem?
6. When writing code, do you stop once in a while and go over what you have written? (Yes/No → Please explain why?)
7. Explain your experience of taking ENGR 132? What events in ENGR 132 have been emotionally intense for you? Please explain/Tell me more.
8. Do you feel optimistic that you will do well in your programming course?
9. Tell me more about your future plans as it relates to pursuing programming, programming-related tasks, and programming intensive courses?
10. Do you have any additional comments?

Concluding remarks
Now, I will compile all the data collected from the session. Next week, you will be invited for a retrospective think-aloud interview, in which you will view your own video/screen capture/EDA/scratch work and we will talk about the emotions you may have experienced at a certain point in time. We will also discuss the reasons as to why you may have felt that way at that time and how you dealt with those emotions.

APPENDIX F: RETROSPECTIVE THINK-ALOUD PROTOCOL

Total time: 60 minutes

Thank you for coming in for this follow-up retrospective think-aloud interview. In this session, we will go over your programming session data and talk about the emotions that you may have experienced while you were working on the programming task. We will also discuss the reasons as to why you may have felt that way and how you dealt with those emotions.
I have extracted segments of the recording of your programming task session. I will stop at every twenty seconds within each segment and ask you a few questions. I am also handing you a sheet of paper, which has a list of emotions. When I ask you about what emotion you were experiencing at a certain point, you can use this sheet to help you identify the emotion you think you were experiencing at that point during the programming session.

<u>**List of Emotions. Adapted from (Pekrun & Perry, 2014)**</u>

Enjoyment	Anger	Sadness
Hope	Anxiety	Disappointment
Joy	Shame	Contentment
Pride	Boredom	Relaxation
Relief	Frustration	Neutral
Gratitude	Hopelessness	<u>Others (Please define this emotion)</u>

<u>**Questions**</u>

- What emotion do you think you were experiencing here?
- What might have caused this emotion?
- How did you deal with this emotion? [Probes: Why did you choose this action? Did you consider other actions? What happened next?]

Now that we have gone over your programming task session, let's discuss your responses from the before-task, during-task, and after-task AEQ. Show them their responses in parallel and ask open-ended questions about why a certain emotion changed after completing the programming task? For example:

- Your responses show a dip in enjoyment after completing the programming task. Why do you think that happened?
- I see that your anger has gone up after completing the programming task. Could you explain the reasons for increase in anger?

APPENDIX G: BIG-FIVE NEUROTICISM SUB-SCALE

The following questions pertain to your tendency to experience unpleasant emotions (anger, anxiety, depression, and vulnerability). Please note that there are no right or wrong responses. Reflect on your current emotional state pertaining to each of these items, and then respond accordingly.

1 - Strongly Agree to 5 - Strongly Disagree

	1	2	3	4	5
I see myself as someone who Is depressed, blue	○	○	○	○	○
I see myself as someone who is relaxed, handles stress well	○	○	○	○	○
I see myself as someone who can be tense	○	○	○	○	○
I see myself as someone who worries a lot	○	○	○	○	○
I see myself as someone who is emotionally stable, not easily upset	○	○	○	○	○
I see myself as someone who can be moody	○	○	○	○	○
I see myself as someone who remains calm in tense situations	○	○	○	○	○
I see myself as someone who gets nervous easily	○	○	○	○	○

APPENDIX H: PROGRAMMING TASK AND RUBRIC

Programming Task

Instructions

1. You can work on any number of problems, in any order during the next 30 minutes. You don't have to complete all problems.
2. Please create a separate MATLAB script for each of the problems and save them in a folder "your_name" on desktop.
3. You can use MATLAB help or online help any time you want.

No.	Programming problem
1.	An engineering student and his family are planning activities for their summer vacation to the Indiana Dunes & Lake Michigan. The student needs a MATLAB function to help them decide what to do. Here are the criteria: If the temperature is greater than or equal to 90°F, they will swim. If the temperature is greater than or equal to 80°F and less than 90°F, they will go boating. If the temperature is less than 80°F, they will go fishing. Write the MATLAB function to determine which activity the family will do based on the outside temperature, and then display the decision to the MATLAB command window. Include appropriate comments with your code. Test your function using the temperatures 70°F, 85°F, and 90°F.
2.	Given the vector *x* = [1 8 3 9 0 1], write the MATLAB code necessary to: Add up the values of all of the elements Compute the sine of the given vector containing x-values (values may be assumed to be in radians) Compute a running sum. A running sum is the summation of a sequence of numbers that is updated every time a new number is added to the sequence. So the running sum of this x vector would be new vector with the values [1 9 12 21 21 22].
3.	Consider an *M*-by-*N* array of random numbers. Write the MATLAB code necessary to move systematically through the array, element by element, and set any value that is less than 0.2 to 0 and any value that is greater than (or equal to) 0.2 to 1.
4.	Write the MATLAB code necessary to evaluate and plot the function $f(t) = 4 - t^2e^{-3t}$ for the time range $0 \le t \le 3$ second. Use an appropriate step size for *t* to create a smooth curve. Label the axes on the plot appropriately.

Rubric for Grading the Programming Task

Problem	Learning Objective	Evidence of Proficiency	Rubric
1.	LO 1.1. Construct an appropriate function definition line	Use the correct syntax for a function: function [output1,…,outputN] = function_name(input1,…,inputM) Function starts with the keyword function Order is output arguments, equal sign, function name, input arguments	Proficient (4): All items are correct Developing (3): Do not use to assess work Emerging (2): Do not use to assess work Insufficient Evidence (1): Syntax is incorrect Not Attempted (0): Did not attempt this item for the problem
	LO 1.2. Code a selection structure	Begin a selection structure with an if Provide an if with a condition for which a true result corresponds to code that immediately follows Use elseif for a series of related conditions Provide each elseif with a condition for which a true result corresponds to code that immediately follows Leave no space between else and if - elseif is a single word Use an else to handle any condition(s) not addressed in the earlier parts of the selection structure and do not use else if no code is needed before the end Do not provide a condition with an else Use an end to terminate a selection structure Indent statements between the if, elseif, else, and end Write a selection structure to address all necessary paths for a given problem	Proficient (4): All items are correct Developing (3): 7 – 9 items are correct Emerging (2): 4 – 6 items are correct Insufficient Evidence (1): less than 4 items are correct Not Attempted (0): Did not attempt this item for the problem

Problem	Learning Objective	Evidence of Proficiency	Rubric
	LO 1.3. Construct relational and logical statements from English statements	Identify correct true/false pattern for the English statement Write a logical statement that is correct Show complete work when verifying that the logical statement results in the pattern identified Write a logical statement that is only as complicated as necessary	Proficient (4): All items are correct Developing (3): 3 items are correct Emerging (2): 2 items are correct Insufficient Evidence (1): Less than 2 items are correct Not Attempted (0): Did not attempt this item for the problem
	LO 1.4. Execute a user-defined function	Use the correct syntax to call a function: [output1,…,outputN] = function_name(input1,…,inputM) Call does not contain keyword function Order is output arguments, equal sign, function name, input arguments, with output arguments and equal sign being optional for a no-output function Functions with no inputs have no input list; use of ( ) is optional Functions with no output arguments have no output list and no equal sign List multiple output arguments inside square brackets and separate them by spaces or commas List multiple input arguments inside parentheses and separate them by commas Call the correct function filename Match the number of input arguments to the number required by the function Match the input argument list to that expected by the function definition line Match the number of output argument(s) to the number required by the function	Proficient (4): All items are correct Developing (3): 4 – 5 items are correct Emerging (2): 2 – 3 items are correct Insufficient Evidence (1): Less than 2 items are correct Not Attempted (0): Did not attempt this item for the problem

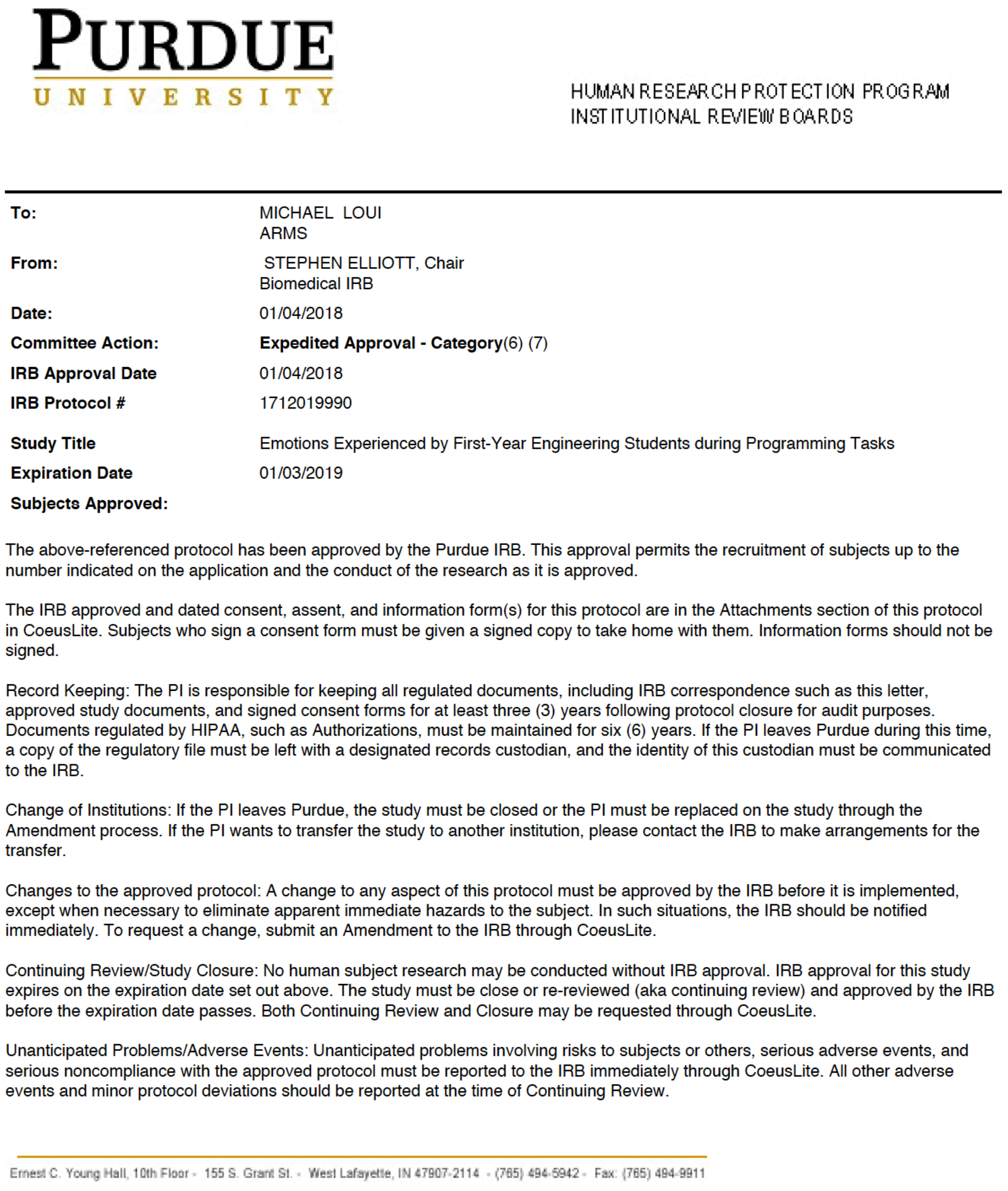
Problem	Learning Objective	Evidence of Proficiency	Rubric
		Match the output argument list to that expected by the function definition line	
2.	LO 2.1. Use built-in functions to perform algebraic and trigonometric calculations	Perform calculations using correct syntax for built-in mathematical function(s) Ex. abs(X), exp(X), sum(X), sin(X), cosd(x) Use appropriate units for the input argument of a built-in trigonometric function Verify the result using an alternative to MATLAB (e.g., calculating by hand or on a calculator), when appropriate	Proficient (4): Built-in functions sum and sin are implemented correctly Developing (3): Only one of the two functions are implemented (either sum or sin) Emerging (2): Do not use to assess work Insufficient Evidence (1): Do not use to assess work Not Attempted (0): Did not attempt this item for the problem
	LO 2.2. Code a definite looping structure that employs vector indexing (if cumsum function not used)	Begin a definite looping structure with a for Use the correct syntax to index a for: index = start_value:increment:end_value or index = vector Include a control statement vector that indexes the loop appropriately for the problem context Perform the correct operations within the definite looping structure Use end to terminate the definite looping structure Indent statements between the for and end	Proficient (4): All items are correct (give credit if cumsum function is correctly used) Developing (3): 4 – 5 items are correct Emerging (2): 2 – 3 items are correct Insufficient Evidence (1): Do not use to assess work Not Attempted (0): Did not attempt this item for the problem
	LO 2.3. Execute a script from the MATLAB Command Window	Call the script name to execute a script from the Command Window	Proficient (4): All items are correct Developing (3): Do not use to assess work Emerging (2): Do not use to assess work Insufficient Evidence (1): Do not use to assess work

Problem	Learning Objective	Evidence of Proficiency	Rubric
			Not Attempted (0): Did not attempt this item for the problem
3.	LO 3.1. Code nested structures	Select an appropriate outer structure for the problem context Select an appropriate inner structure for the problem context Initialize variables as appropriate for successful execution of the nested structure Update variables as appropriate for successful execution of the nested structure Use end to terminate each structure Indent code clearly to demarcate inner and outer structures	Proficient (4): All items are correct Developing (3): 4 – 5 items are correct Emerging (2) 2 – 3 items are correct Insufficient Evidence (1): Less than 2 items are correct Not Attempted (0): Did not attempt this item for the problem
	LO 3.2. Replace elements of arrays	Replace elements of an array using correct syntax: original_array(row range, column range) = replacement_values_array Original_array is to the left of the equal sign, The replacement values are to the right of the equal sign, Parentheses are used, The row and column ranges are separated by a comma Ex. A(1,1:3) = [0 0 0]; % replace row 1 col 1-3 with zeros Reference the correct array variable name in which to replace elements Reference the correct row range Reference the correct column range Correctly dimension the replacement values array Verify that the values in the replacement array are correct	Proficient (4): All items are correct Developing (3): 4 – 5 items are correct Emerging (2) 2 – 3 items are correct Insufficient Evidence (1): Less than 2 items are correct Not Attempted (0): Did not attempt this item for the problem
	LO 3.3. Execute a script from the	Call the script name to execute a script from the Command Window	Proficient (4): All items are correct Developing (3): Do not use to assess work

Problem	Learning Objective	Evidence of Proficiency	Rubric
	MATLAB Command Window		Emerging (2): Do not use to assess work Insufficient Evidence (1): Do not use to assess work Not Attempted (0): Did not attempt this item for the problem
4.	LO 4.1. Use built-in functions to perform algebraic and trigonometric calculations	Perform calculations using correct syntax for built-in mathematical function(s) Ex. abs(X), exp(X), sum(X), sin(X), cosd(x) Use appropriate units for the input argument of a built-in trigonometric function Verify the result using an alternative to MATLAB (e.g., calculating by hand or on a calculator), when appropriate	Proficient (4): All items are correct (exp function) Developing (3): Do not use to assess work Emerging (2): Do not use to assess work Insufficient Evidence (1): Do not use to assess work Not Attempted (0): Did not attempt this item for the problem
	LO 4.2. Create an x-y plot from a single data set	Use the correct syntax for the plot command: plot(x, y, 'line/marker formatting') Identify the independent (x) and dependent (y) variables correctly Select data markers and lines appropriately: data markers with no line (for raw data), line with no data markers (known model), data markers with overlaid line (for raw data with model)	Proficient (4): All items are correct (exp function) Developing (3): 3 items are correct Emerging (2): 2 items are correct Insufficient Evidence (1): Less than 2 items are correct Not Attempted (0): Did not attempt this item for the problem
	LO 4.3. Format plots for technical presentation	Use the correct syntax for title Use the correct syntax for xlabel Use the correct syntax for ylabel Provide a descriptive title that references the problem context, the independent (x) variable, and the dependent (y) variable	Proficient (4): All items are correct Developing (3): 7 – 9 items are correct Emerging (2): 4 – 6 items are correct

Problem	Learning Objective	Evidence of Proficiency	Rubric
		Provide a clear x-axis label with units Provide a clear y-axis label with units Turn the gridlines on Select color and marker/line style(s) that are as specified or distinctive (when multiple data sets) Properly format the legend, when multiple data sets and/or models Use a common x-axis scale range (when using subplots to compare data)	Insufficient Evidence (1): Less than 4 items are correct Not Attempted (0): Did not attempt this item for the problem
	LO 4.4. Execute a script from the MATLAB Command Window	Call the script name to execute a script from the Command Window	Proficient (4): All items are correct Developing (3): Do not use to assess work Emerging (2): Do not use to assess work Insufficient Evidence (1): Do not use to assess work Not Attempted (0): Did not attempt this item for the problem

APPENDIX I: IRB APPROVAL AND CONSENT FORM



HUMAN RESEARCH PROTECTION PROGRAM
INSTITUTIONAL REVIEW BOARDS

To:	MICHAEL LOUI ARMS
From:	STEPHEN ELLIOTT, Chair Biomedical IRB
Date:	01/04/2018
Committee Action:	**Expedited Approval - Category**(6) (7)
IRB Approval Date	01/04/2018
IRB Protocol #	1712019990
Study Title	Emotions Experienced by First-Year Engineering Students during Programming Tasks
Expiration Date	01/03/2019
Subjects Approved:	

The above-referenced protocol has been approved by the Purdue IRB. This approval permits the recruitment of subjects up to the number indicated on the application and the conduct of the research as it is approved.

The IRB approved and dated consent, assent, and information form(s) for this protocol are in the Attachments section of this protocol in CoeusLite. Subjects who sign a consent form must be given a signed copy to take home with them. Information forms should not be signed.

Record Keeping: The PI is responsible for keeping all regulated documents, including IRB correspondence such as this letter, approved study documents, and signed consent forms for at least three (3) years following protocol closure for audit purposes. Documents regulated by HIPAA, such as Authorizations, must be maintained for six (6) years. If the PI leaves Purdue during this time, a copy of the regulatory file must be left with a designated records custodian, and the identity of this custodian must be communicated to the IRB.

Change of Institutions: If the PI leaves Purdue, the study must be closed or the PI must be replaced on the study through the Amendment process. If the PI wants to transfer the study to another institution, please contact the IRB to make arrangements for the transfer.

Changes to the approved protocol: A change to any aspect of this protocol must be approved by the IRB before it is implemented, except when necessary to eliminate apparent immediate hazards to the subject. In such situations, the IRB should be notified immediately. To request a change, submit an Amendment to the IRB through CoeusLite.

Continuing Review/Study Closure: No human subject research may be conducted without IRB approval. IRB approval for this study expires on the expiration date set out above. The study must be close or re-reviewed (aka continuing review) and approved by the IRB before the expiration date passes. Both Continuing Review and Closure may be requested through CoeusLite.

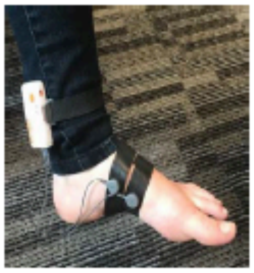
Unanticipated Problems/Adverse Events: Unanticipated problems involving risks to subjects or others, serious adverse events, and serious noncompliance with the approved protocol must be reported to the IRB immediately through CoeusLite. All other adverse events and minor protocol deviations should be reported at the time of Continuing Review.

Ernest C. Young Hall, 10th Floor - 155 S. Grant St. - West Lafayette, IN 47907-2114 - (765) 494-5942 - Fax: (765) 494-9911

RESEARCH PARTICIPANT CONSENT FORM
Emotions Experienced by First-Year Engineering Students during Programming Tasks
Dr. Michael C. Loui & Zahra Atiq
School of Engineering Education
Purdue University

What is the purpose of this study?
I am studying the emotions that first-year engineering students experience while they work on programming tasks. Subsequently, I am also hoping to understand the self-regulation strategies that students adopt to deal with those emotions. Since emotions influence learning and success, the findings of this study will help educators mitigate the effects of negative emotions and promote the effects of positive emotions on student learning and success. You are eligible to participate only if you are age 18 or older.

What will I do if I choose to be in this study?
You will be requested to come for two sessions. The first session will be 75 minutes long, and the second session will be 60 minutes.



For the first session, you will work on a programming task, which consists of four programming problems. Before, during, and after this task, you will be prompted to respond to a few questions about the emotions you may be experiencing at that moment. Finally, you will be interviewed about your experiences with the programming task. While working on the programming task, you will be required to wear a non-invasive device on your left foot (PICTURE SHOWN ON THE RIGHT). This device records your body's arousal and is hence crucial for assessing student emotions.
For the second session, you will be invited to come for a retrospective think-aloud interview 3 – 5 days after the first session. During this interview, I will review the data collected during the first session. You will be asked about the emotions that you may have experienced while working on the programming problems. This session will last for almost one hour.

How long will I be in the study?
You will be requested to come for two sessions, spaced three to five days apart. The first session will be around 75 minutes long, and the second session will be around 60 minutes.

What are the possible risks or discomforts?
There are no greater than the participant would encounter in daily life or during the performance of routine physical or psychological exams or tests. You may experience minor psychological or emotional discomfort. In the first session you will be asked to work on a programming task, which may cause positive or negative emotions. However, the programming tasks require the knowledge that you have already learned. Hence, it is expected that the emotional discomfort will be minimal. Moreover, the tasks are low-stakes because they will not affect your course grade. During the second session, you will be asked to reflect on the emotions you experienced during the first programming session. Reflecting on your experiences may also cause minor discomfort. Breach of confidentiality is a risk, however, we have safeguards in place to minimize this risk (see confidentiality section below).

Purdue IRB Protocol #: 1712019990 - Expires on: 03-JAN-2019

Are there any potential benefits?
There are no direct benefits to the participants of the study. However, by participating in this study, you will have a chance to reflect on your emotional experiences while working on programming tasks, the reasons for experiencing those emotions, and the self-regulation strategies you adopted to deal with those emotions. These reflections may help you understand your own emotions better and choose appropriate self-regulation strategies when needed.

Will I receive payment or other incentive?
You will receive a $40 Amazon gift card via email. If you do not complete both sessions, payment will be pro-rated as a fraction of the expected 135 minutes of data collection, based on the amount of time that you participated.

Will information about me and my participation be kept confidential?
Data collected will include video, audio, EDA, eye-gaze data, screen recording, background and survey data, and the scratch paper that you turn in. The data for this study will be kept confidential to the extent allowed by U.S. federal and state law. No published results will identify you by name, and your name will not be associated with the findings. Under certain circumstances, information that identifies you may be released for internal and external reviews of this project. Data collected during interviews will be de-identified, coded and the names of participants maintained separately from the data. The researcher will treat all data with utmost respect for participant privacy and work to create a safe environment in which participants can express their opinions openly and honestly, without fear that their responses or academic information would be disclosed publicly. To protect participants' identities, the researcher will use an identity mapping procedure, with the master key of identities held in the secure facility. The data itself will be shared in de-identified form only; de-identification will be performed by the researcher. The researcher will store the data in a secure institutional data repository. The data will be securely encrypted. Only the researcher(s) will have access to the confidential data. Interviews will be audio-recorded using digital audio recorders. Audio data will be kept confidential and will be used for transcription and coding purposes. Once the audio data is transcribed, the audio files will be deleted. However, the de-identified transcripts will be kept indefinitely. The video data (facial expressions and screen recordings), EDA, de-identified background survey data, and eye-gaze data will also be kept indefinitely. The results of this study, including pictures of participants' facial expressions may be published or presented at professional meetings, but the identities of all research participants will not be disclosed. To disguise the participants' identities, images of student facial expressions may be pixilated or converted to unidentifiable formats using image filters, or only sections of an image may be used. The project's research records may be reviewed by the Purdue University, responsible for regulatory and research oversight.

What are my rights if I take part in this study?
Your participation in this study is voluntary. You may choose not to participate or, if you agree to participate, you can withdraw your participation at any time without penalty or loss of benefits to which you are otherwise entitled. If you wish to withdraw anytime during the study, your data will not be included in the study and will be permanently deleted. If you have completed both sessions, your data will not be excluded from the study. This research is being conducted independently of the course (ENGR 132) you are currently enrolled in. Your participation in the study will have no effect on your participation or grade in ENGR 132.

Who can I contact if I have questions about the study?
If you have questions, comments or concerns about this research project, you can talk to one of the researchers. Please contact Dr. Michael Loui (mloui@purdue.edu). If you have questions about your rights while taking part in the study or have concerns about the treatment of research participants, please call the Human Research Protection Program at (765) 494-5942, email (irb@purdue.edu)or write to:

Human Research Protection Program - Purdue University
Ernest C. Young Hall, Room 1032

IRB No.______________ Page 2

Purdue IRB Protocol #: 1712019990 - Expires on: 03-JAN-2019

155 S. Grant St., West Lafayette, IN 47907-2114

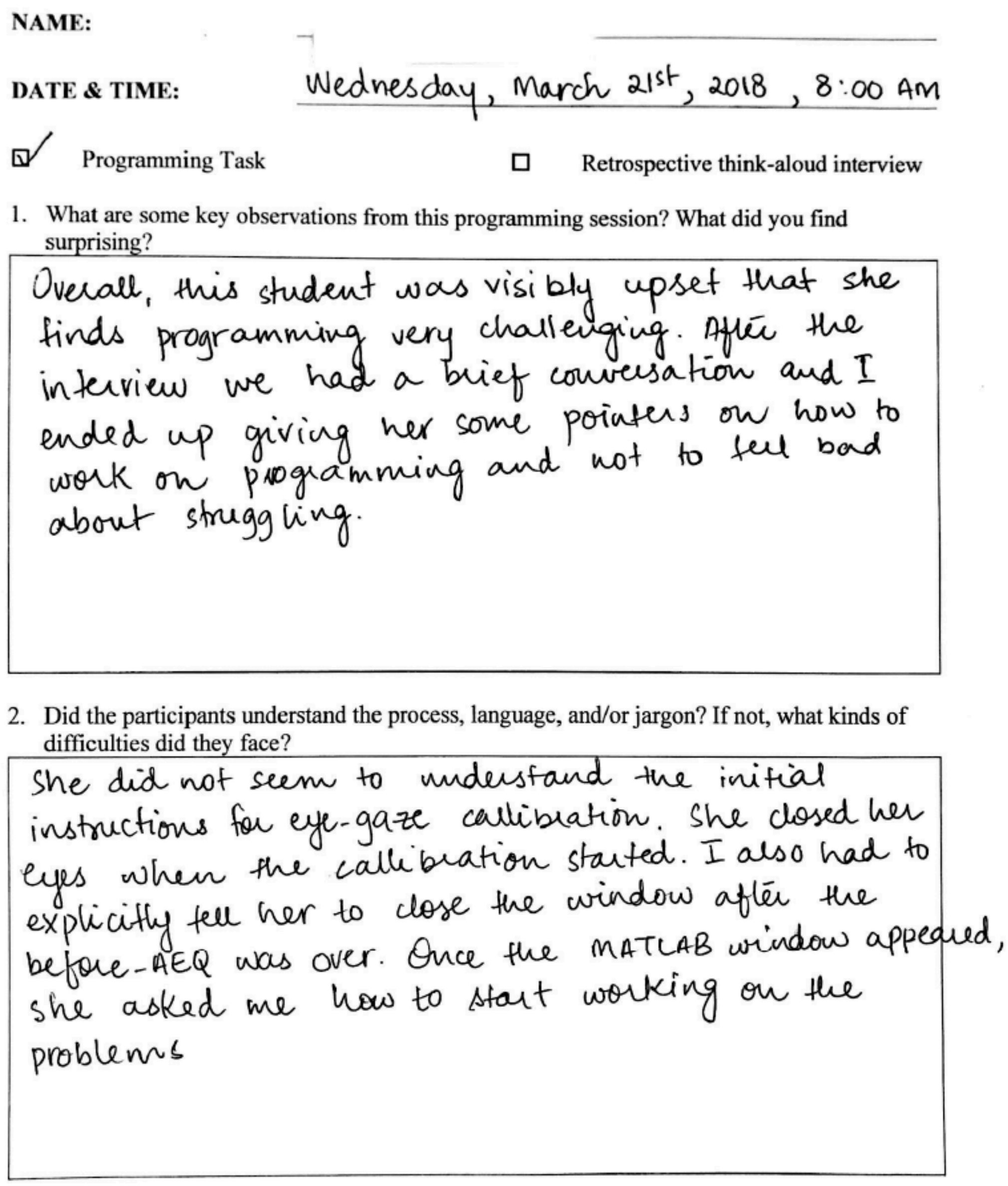
Documentation of Informed Consent

I have had the opportunity to read this consent form and have the research study explained. I have had the opportunity to ask questions about the research study, and my questions have been answered. I am prepared to participate in the research study described above. I will be offered a copy of this consent form after I sign it.

______________________________	________________
Participant's Signature	Date
______________________________	
Participant's Name	
______________________________	________________
Researcher's Signature	Date

APPENDIX J: SAMPLE MEMO AND JOURNAL ENTRIES

Sample Memo from Data Collection



NAME:

DATE & TIME: Wednesday, March 21st, 2018 , 8:00 AM

[✓] Programming Task ☐ Retrospective think-aloud interview

1. What are some key observations from this programming session? What did you find surprising?

Overall, this student was visibly upset that she finds programming very challenging. After the interview we had a brief conversation and I ended up giving her some pointers on how to work on programming and not to feel bad about struggling.

2. Did the participants understand the process, language, and/or jargon? If not, what kinds of difficulties did they face?

She did not seem to understand the initial instructions for eye-gaze callibration. She closed her eyes when the callibration started. I also had to explicitly tell her to close the window after the before-AEQ was over. Once the MATLAB window appeared, she asked me how to start working on the problems

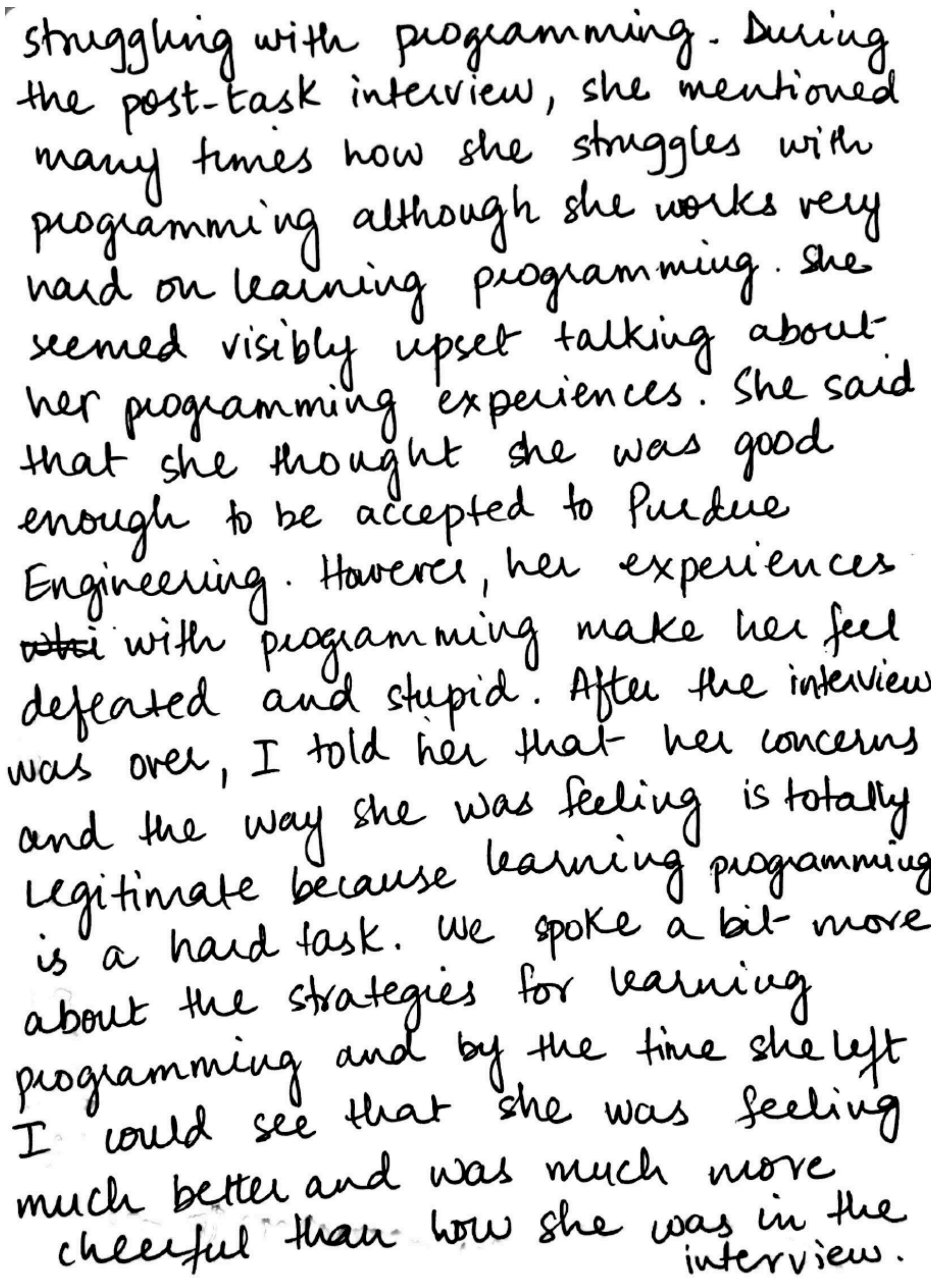
3. What problems were encountered during the execution of the programming session/ short interview/retrospective think-aloud interview? (e.g., logistical, behaviors of individuals, questions that were confusing, etc.)

- Eyes closed during initial callibration. However, the callibration turned out good.
- She had trouble using the white mouse, so I had to go and switch the mouse (starting of before-AEQ).

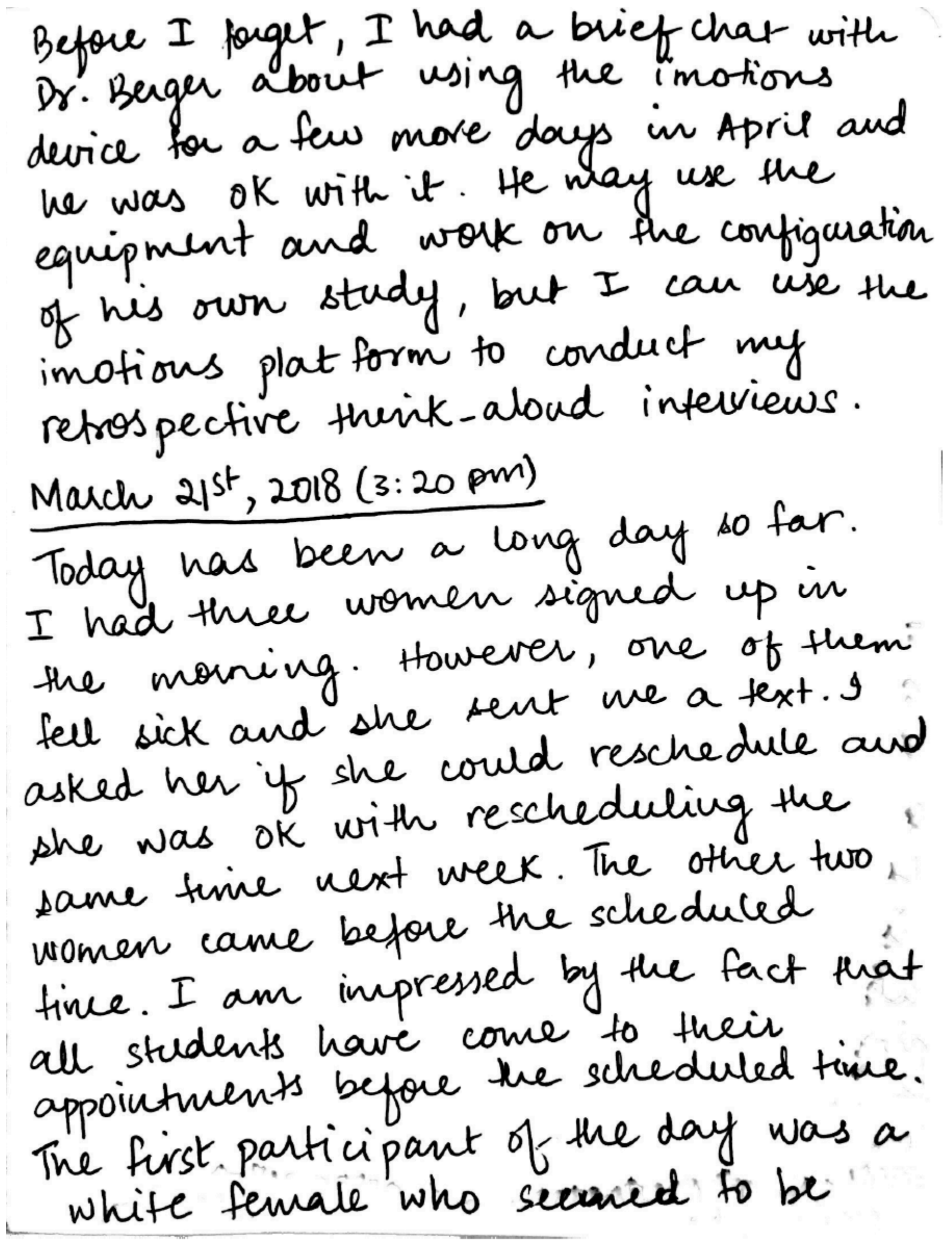
4. What are some changes that can be made for future sessions?

5. Additional comments

Sample Journal Entry from Data Collection



struggling with programming. During the post-task interview, she mentioned many times how she struggles with programming although she works very hard on learning programming. She seemed visibly upset talking about her programming experiences. She said that she thought she was good enough to be accepted to Purdue Engineering. However, her experiences ~~other~~ with programming make her feel defeated and stupid. After the interview was over, I told her that her concerns and the way she was feeling is totally legitimate because learning programming is a hard task. We spoke a bit more about the strategies for learning programming and by the time she left I could see that she was feeling much better and was much more cheerful than how she was in the interview.

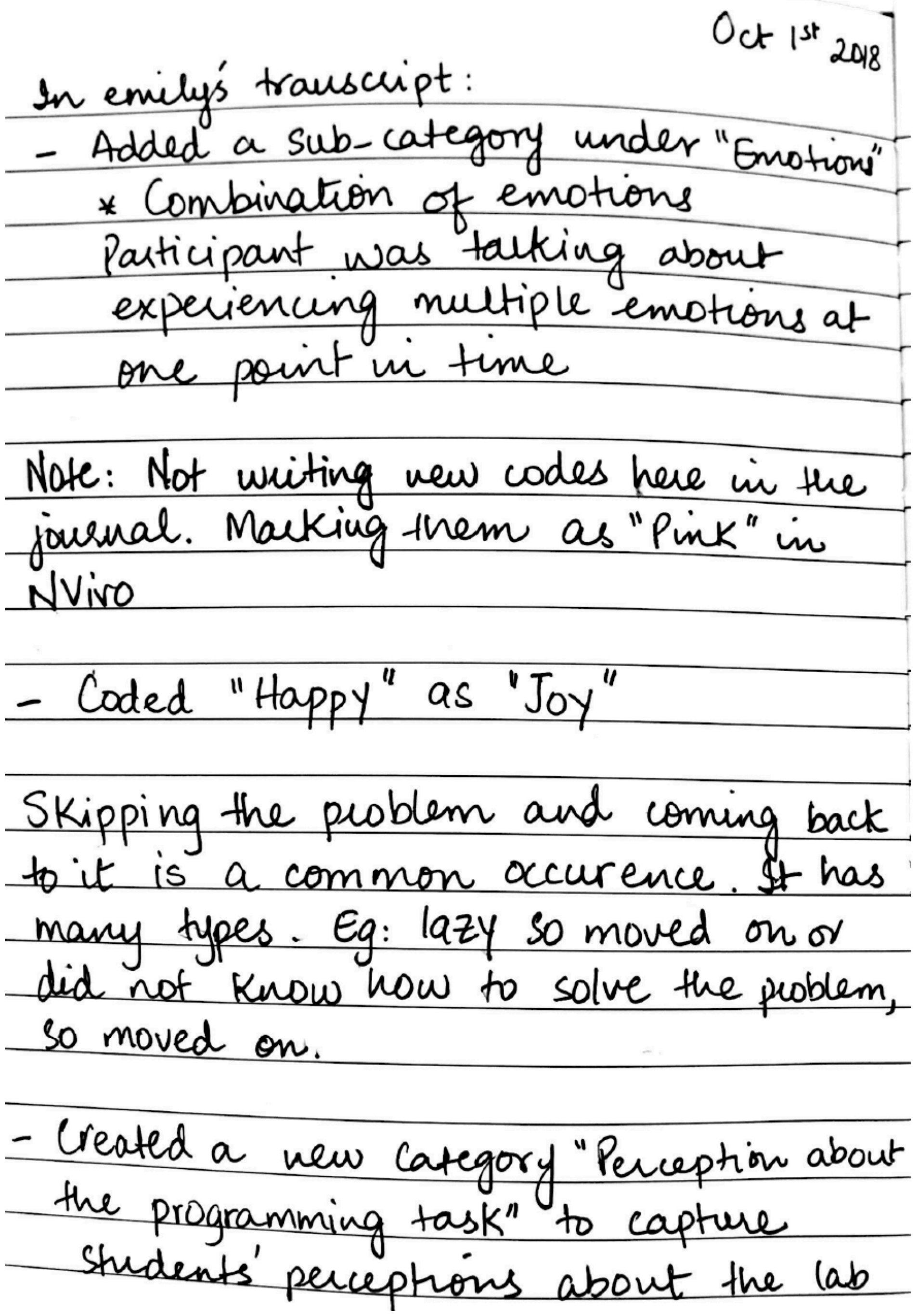


Before I forget, I had a brief chat with
Dr. Berger about using the imotions
device for a few more days in April and
he was OK with it. He may use the
equipment and work on the configuration
of his own study, but I can use the
imotions platform to conduct my
retrospective think-aloud interviews.

March 21st, 2018 (3:20 pm)

Today has been a long day so far.
I had three women signed up in
the morning. However, one of them
fell sick and she sent me a text. I
asked her if she could reschedule and
she was OK with rescheduling the
same time next week. The other two
women came before the scheduled
time. I am impressed by the fact that
all students have come to their
appointments before the scheduled time.
The first participant of the day was a
white female who seemed to be

Sample Journal Entry from Data Analysis



Oct 1st 2018

In emily's transcript:

- Added a sub-category under "Emotions"
  * Combination of emotions
  Participant was talking about experiencing multiple emotions at one point in time

Note: Not writing new codes here in the journal. Marking them as "Pink" in NVivo

- Coded "Happy" as "Joy"

Skipping the problem and coming back to it is a common occurence. It has many types. Eg: lazy so moved on or did not know how to solve the problem, so moved on.

- Created a new category "Perception about the programming task" to capture students' perceptions about the lab setting of the programming task (It was not a test / graded task)

* Finished coding Emily. There were some excerpts where she was explaining her thought process. I found those sections particularly hard to code

Oct 2nd 2018

After having a discussion with Dr. Loui, I decided to start the categorization process for the Category "Emotion or Behavior Reason". Since this is a large category with many sub-codes, it makes sense to start the categorization process sooner rather than later.

APPENDIX K: SAMPLE CODEBOOK FOR QUALITATIVE ANALYSIS

Name	Description	Files	References
BEFORE AND AFTER AEQp	Change in emotions before and after the programming task	0	0
Anger		16	20
Anxiety		1	1
Embarrassment		2	2
Enjoyment		17	19
Pride		17	19
Relief		17	17
Shame		12	12
EMOTION	Emotions experienced by the students. These emotions are taken from the control-value theory, except for Neutral	0	0
Anger		5	11
Anxiety		15	52
Boredom		3	7
Disappointment		12	27
Enjoyment		8	15
Frustration		17	123
Hope		7	15
Hopelessness		3	12
Joy		12	35
Neutral		11	46
Pride		11	27
Relief		13	45
Sadness		3	5
Shame		2	4
EMOTIONS AND REASONS RQ1	The emotions students experienced during the programming task and the reasons students reported for	0	0

Name	Description	Files	References
	experiencing the emotions		
DEALING WITH DIFFICULTIES	In the “Dealing with Difficulties” stage, students dealt with the difficulties they faced in the “Encountering Difficulties” stage	0	0
ENGAGED IN TRIAL-AND-ERROR TO SOLVE THE ISSUE	While engaging in trial-and-error, students went back and forth over the work they had written and tried to identify errors in their code	0	0
Arrived at answer in an unconventional way		1	1
Found another way to do a task		1	1
Needs more time to do trial-and-error		3	3
The problem was easy to fix		1	1
Tried to fix the errors		4	4
Trying to troubleshoot to understand why the program isnt working		7	15
KEPT PUSHING THROUGH	Most students mentioned that they pushed through and persisted, even when they encountered challenges	0	0
Getting stuck at a problem motivates them to solve the problem		1	1
Just keep going with coding		2	4
Tries not to deal with or dwell on the emotions		2	2
LOOKED UP HELP OR ONLINE RESOURCES	Students looked at help when they encountered syntax errors, or if they did not know the syntax of a function. They were either able to resolve errors by looking up help, or they were	0	0

Name	Description	Files	References
	unable to resolve errors		
Could not find a solution online		1	1
Did not have to use outside resources		1	1
Does not understand the results of searching online of MATLAB help		1	1
Looked at MATLAB or online resources		11	37
Looking up help did not work		2	5
Not knowing how to fix the problem		1	1
Online help was not very helpful		1	1
Should have known how to solve a simple error		2	2
The errors were not too major, so they were not too bothered		1	1
ENCOUNTERING DIFFICULTIES	In the “Encountering Difficulties” stage, students encountered difficulties while working on the programming task. These difficulties hindered students’ progress	0	0
FORGOT HOW SOMETHING WORKED	Some students experienced negative emotions like frustration and anger, when they realized they had previously worked on something, but forgot how to use their previous learning during the programming task	0	0
Forgot how a function worked		2	3
Knew how to do it, but couldn't find the		1	2

Name	Description	Files	References
MATLAB command for it			
THEY DID NOT KNOW HOW TO DO A PROBLEM	Students did not know how to do a certain problem on the programming task or did not know how to fix an issue they encountered while working on the programming task	0	0
Did not know how to do the problem		7	15
Did not know how to fix the issue		5	6
Did not know how to use a function		2	2
Did not know what to do		10	22
Did not understand the instructions for the problem		1	1
Felt stuck with a problem		1	1
Has not seen this error before		1	2
Having difficulty with difficult problem		1	1
Ignored the problem they were having difficulty with		2	2
In a rush to do something but not knowing what to do		1	2
Knew what was being asked but the code didnt work		2	2
Mad at themselves for not knowing what to do		2	2
Not being able to think clearly		1	1
Not sure how to interpret the problem		1	1

Name	Description	Files	References
Not sure what they were doing		2	3
Optimistic that they will get the new problem right		1	1
The task was challenging because they didnt quite know what to do		1	1
They know they are not good with loops		1	1
They were not sure how to start the problem		2	2
Thought she could do it with pencil and paper, but it was hard to translate into code		1	1
WHEN THE CODE DOES NOT WORK	This stage is when code did not execute correctly	0	0
Have issues with logical operators		1	1
They re-read the problem when they are stuck		3	4
Trying to figure out what to do		7	11
What they thought would work, did not work		1	1
When the code doesnt work		13	33
GETTING STARTED	Before students started working on the programming problems, they went through the “Getting Started” stage, which involved reading instructions, getting familiar with the interface, and saving the files and folders. The students then started understanding the problems and decided on which problem to work on	0	0

Name	Description	Files	References
GETTING FAMILIAR WITH THE INTERFACE	Students got familiar with unknown interface (windows vs. mac), or had difficulty saving files in the right place	0	0
A different computer than what they are used to		1	1
Able to do the problem		2	2
Code template were not provided		1	3
Did not want to mess up the interface		1	1
Encountering difficulties with the IDE		1	1
Faced challenges with the interface		1	3
File not present in their folder		3	3
Got everything set up before starting the problem		9	10
Setup and formatting of problem is stressful		2	2
Spent a long time figuring out where the MATLAB script was		1	1
Trying to save file		12	24
Wasted some time initially trying to figure out interface related stuff		2	2
OPTIMISTIC THEY COULD DO THE PROBLEM	Students evaluated if they would be successful at writing code for that problem, and felt confident about their ability to write code for the chosen problem	0	0
Problem did not seem hard		2	2
Problem were shorter than expected		1	1

Name	Description	Files	References
They knew how to do a certain task		5	8
They knew they could do the task		5	9
PROBLEMS ON THE TASK WERE FAMILIAR	Students had previously seen one or more problems on the programming task	0	0
Had seen the problem before		2	3
Problem was similar to hw assignments		1	1
Programming problem related to concepts taught in class		1	1
They had done the task before		2	2
READING THE INSTRUCTIONS	Reading the instructions to figure out which one to start working on	0	0
Figuring out how to get started with a problem		1	1
Getting started with the programming task		1	1
Going through the instructions		11	25
STOPPING	During the "Stopping" stage, students encountered three types of events. Students decided to move on to the next problem, when they did not understand a problem, or when they had successfully completed on problem. Students were in the "Stopping" stage when the programming task finished	0	0
COULD NOT UNDERSTAND THE PROBLEM, SO DECIDED TO MOVE ON	When students were unable to successfully complete one problem, they decided to move on to the next problem, instead of	0	0

Name	Description	Files	References
	spending time trying to figure out a solution for the problem		
Could come back and troubleshoot later		1	2
Did not feel like solving the problem, so decided to skip it and come back to it later		1	1
Did not know how to do a problem to skipped to the next one		1	1
Did not know how to do the problem and decided to skip it, hoping to come back to it later		2	3
Did not like a certain sub-task, so decided to skip it and come back to it later		1	1
Does not like a certain task, so did not do it		1	2
Lost focus and was tired, so decided to skip problem and come back to it later		1	1
Moved on to the next problem because the previous problem was a lost cause		2	3
Remembered how to do a previous problem, so went back to it		1	1
FINISHED ONE PROBLEM	When students successfully finished one problem	0	0
Finished one problem on the task		6	8
Finished the problem by themselves		1	1
REACHING THE END OF THE TASK	When the programming task finished after thirty minutes	0	0

Name	Description	Files	References
Losing motivation by the end of the task		1	2
Reached the end of the task		3	4
They killed time at the end of the task		1	1
Unable to complete a task		1	2
Unable to finish all questions		2	2
Wasted time switching between problems		1	1
When the time ended for the problems		1	1
SUCCEEDING	Students reached the “Succeeding” stage when they had successfully finished a problem on the programming task	0	0
CODE RAN SUCCESSFULLY ON FIRST ATTEMPT	When students wrote code for a programming problem, and it executed correctly on the first attempt	0	0
Code ran successfully on the first attempt		2	2
Nothing wrong was happening		1	2
FIGURED OUT THE PROBLEM WITH THEIR CODE	After encountering and dealing with difficulties, students successfully resolved the errors	0	0
Code had many errors but it finally worked		1	1
No more errors to deal with		1	1
Save the file because the code works		6	6
Was able to fix errors and made the program work		6	8

Name	Description	Files	References
When the code works		15	34
TYPING CODE	Students typed code for the problem they selected in the "Getting Started" stage	0	0
COMPUTATION IS EASY, WRITING CODE IS HARD	Sometimes students perceived they could find a solution of a descriptive problem in their minds, but faced difficulty converting their solution into syntax	0	0
Having code written is the hardest thing for them		1	1
The computation is easy, finding code to do the computation is hard		1	1
MONITORING	Some students adopted a cautious stance while typing code. These students expected to encounter errors in their code, and hence decided to double check everything as they typed	0	0
Anticipating that the code will not work		8	13
Code had to be exact and precise		1	1
Commented test case in the script		1	1
Commenting is a secondary task		3	3
Copy pasting the code to go a bit faster		2	2
Figured out a problem in advance and change course accordingly		1	2
NEGATIVE EXPECTATIONS	In this stage, students had typed the code but had not executed it. Some students had negative expectations from the code they had written, and hence they	0	0

Name	Description	Files	References
	thought their code would be erroneous		
Does not know if the code will work		1	1
They thought they were about to mess up		1	3
Was expecting the code to not work		1	3
Wondering if the code would run correctly		1	1
POSITIVE EXPECTATIONS	Students had typed the code but had not executed it. Some students had positive expectations, hence, they felt confident their work would yield positive outcomes.	0	0
Commenting and running test cases is not hard		2	2
Successfully converted math formula into code		2	2
Was expecting the code to work		1	1
Was making good progress on the code		2	4
SIMPLE MISTAKES	While typing code, some students made simple incidental mistakes (e.g., spelling mistakes)	0	0
Messed up the commands		1	1
Tends to make careless mistakes		2	2
Was making spelling mistakes		4	6
SELF-REGULATION RQ3	The self-regulation strategies students adopted while working on the programming task. For this RQ, I used Zimmerman's self-regulated learning. Self-regulation is a process in which	0	0

Name	Description	Files	References
	students adapt and orient their thoughts, feelings, and behaviors to attain their goals		
Forethought	In the forethought stage, students prepare for the upcoming activity	0	0
Goal setting		3	3
Intrinsic interest		0	0
Learning goal orientation		0	0
Outcomes expectations		3	3
Self-confidence		6	9
Self-efficacy		3	5
Strategic Planning		7	23
Performance	In the performance stage, students work on the activity. During this stage, students engage in two types of processes that may affect their motivation and learning: self-control and self-observation	0	0
Self-control		0	0
Attention focusing		6	10
Self-instruction		3	5
Task strategies		10	18
Self-observation		2	2
Self-experimentation		6	9
Self-recording		8	18
Self-reflection		1	1
Self-judgment		0	0
Causal attribution		4	5
Self-evaluation		5	8
Self-reaction		0	0
Adaptive		6	8
Defensive		2	2
Self-satisfaction		6	6

APPENDEX L: RAW DATA

Table 10: Detailed Student Grades Corresponding to Each Learning Objective

Pseudonym	Problem 1					Problem 2				Problem 3				Problem 4					Total (56)	Percentage
	Learning Objective				Total (16)	Learning Objective			Total (12)	Learning Objective			Total (12)	Learning Objective				Total (16)		
	1.1	1.2	1.3	1.4		2.1	2.2	2.3		3.1	3.2	3.3		4.1	4.2	4.3	4.4			
Andrew	0	4	4	4	12	4	0	4	8	0	0	0	0	0	0	0	0	0*	20	35.7
Anna	4	4	4	4	16	4	4	4	12	0	0	0	0*	0	0	0	0	0	28	50.0
Becky	4	4	4	4	16	3	0	4	7	0	0	0	0	4	0	0	1	5	28	50.0
Christina	4	3	4	4	15	4	0	4	8	0	0	0	0	0	3	0	0	3	26	46.6
Ella	4	4	4	4	16	4	4	4	12	1	2	4	7	4	4	4	4	16	51	91.0
Emily	4	4	4	4	16	4	2	4	10	4	3	4	11	0	4	0	4	8	45	80.3
Erica	4	4	4	4	16	4	2	4	10	3	2	4	9	0	0	0	0	0*	35	62.5
George	0	0	0	0	0+	0	0	0	0*	0	0	0	0*	0	0	0	0	0+	0	0.0
Harris	4	4	4	4	16	4	4	4	12	0	0	0	0	4	4	0	4	12	40	71.4
Jack	4	4	4	4	16	4	3	4	11	0	0	0	0	4	4	0	4	12	39	69.6
John	4	4	4	4	16	4	4	4	12	0	0	0	0	4	4	4	4	16	44	78.5
Lilly	4	4	4	4	16	4	2	4	10	0	0	0	0	4	4	4	4	16	42	75.0
Mark	1	4	4	1	10	0	0	0	0*	0	0	0	0+	0	0	0	0	0+	10	17.8
Martha	4	4	4	4	16	4	0	4	8	0	0	0	0	0	0	0	0	0*	24	42.8
Rachel	4	4	4	4	16	4	0	4	8	0	0	0	0	4	4	4	4	16	40	71.4
Randy	0	4	4	4	12	4	2	4	10	0	0	0	0	0	4	0	4	8	30	53.5
Sarah	4	4	4	4	16	4	0	4	8	0	0	0	0*	0	0	0	0	0*	24	42.8

Not Attempted (*), File not Saved (+)

Table 11: Before- and After-AEQp Responses for Each Student

Pseudonym	**Positive Activating Emotions**				**Negative Activating Emotions**			
	Enjoyment		**Pride**		**Anger**		**Shame**	
	Before	**After**	**Before**	**After**	**Before**	**After**	**Before**	**After**
Andrew	3.0	3.5	3.0	2.2	2.5	3.6	1.0	3.0
Anna	2.2	1.0	1.0	2.7	4.5	3.3	4.0	1.75
Becky	4.0	2.0	4.0	3.5	3.5	3.5	2.0	2.25
Christina	3.0	1.0	3.0	1.8	3.0	1.8	4.0	4.25
Ella	3.2	2.5	2.0	3.3	2.0	3.6	2.0	2.5
Emily	2.0	1.0	2.0	2.8	1.0	1.3	2.0	2.0
Erica	3.4	2.0	2.0	3.4	3.0	2.3	1.0	1.0
George	3.2	2.5	2.0	4.0	3.0	3.3	2.0	1.0
Harris	3.0	1.5	3.0	1.8	2.5	2.1	2.0	2.75
Jack	3.6	2.5	2.0	2.4	3.5	1.8	1.0	1.0
John	3.2	2.0	2.0	2.3	3.0	2.8	3.0	1.0
Lilly	2.6	1.0	1.0	2.1	5.0	2.5	3.0	1.75
Mark	3.2	2.5	4.0	2.7	4.0	3.6	2.0	2.75
Martha	3.0	1.0	2.0	2.5	2.0	3.6	2.0	2.5
Rachel	2.2	2.0	1.0	2.2	2.5	3.8	2.0	1.25
Randy	3.0	1.5	2.0	2.4	3.5	3.3	3.0	2.0
Sarah	2.8	2.0	2.0	2.8	2.5	2.6	2.0	1.75

Table 12: Emotions Measured Only on the Before- or After-AEQp

Pseudonym	**Before**			**After**
	Hope	**Anxiety**	**Hopelessness**	**Relief**
Andrew	1.5	3.0	1.0	2.5
Anna	2.3	3.2	2.4	2.5
Becky	4.0	2.8	1.0	2.3
Christina	2.3	3.4	2.4	4.1
Ella	3.5	2.0	1.0	1.0
Emily	3.1	1.4	1.0	1.0
Erica	3.1	2.0	1.0	1.0
George	3.8	1.8	1.0	1.0
Harris	3.5	2.8	1.2	2.3
Jack	3.3	1.4	1.0	1.3
John	2.8	3.0	1.8	2.0
Lilly	2.6	3.2	1.0	2.0
Mark	3.3	3.6	2.0	2.1
Martha	3.3	3.0	1.0	1.3
Rachel	2.5	3.2	2.4	1.0
Randy	3.1	3.6	1.4	2.3
Sarah	2.6	3.2	1.2	1.1