COMPUTATIONAL THINKING SELF-EFFICACY IN HIGH SCHOOL LATIN LANGUAGE LEARNING

by

Dennis Clark Dickerson, Jr.

A Dissertation Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Education

Major: Instruction and Curriculum Leadership

The University of Memphis

May 2019

Dedication

This manuscript is dedicated to my father, Dennis Clark Dickerson, Sr. Thank you for everything, Dad. You are my role model and the number one influence in my life.

This manuscript is also dedicated to my wife, Dianna Watkins-Dickerson, whose love and support were integral throughout this entire process. I love you. You and D3 are my world!

Acknowledgement

I would like to add a special thank you to my advisor and dissertation chair, Dr. Andrew Tawfik. He was always there when I needed him. His guidance was instrumental in the completion of this dissertation.

Abstract

Research suggests that computational thinking is a necessary skill exercised in STEM courses, non-STEM fields, and in everyday life. However, very little research has investigated the potential transfer of computational thinking self-efficacy available through classical Latin courses. This causal comparative study contrasted the computational thinking self-efficacy of computer science students with no exposure to Latin to computer science students with exposure to Latin at a Memphis all-boy high school. The independent variables were Latin language learning experience, i.e., up to 6 years total of Latin language learning (n = 33), versus 0 years of Latin language learning experience (n = 20). Additional data on the number of years enrolled in other foreign languages was collected. The dependent variable was mean scores of items found on a computational thinking and problem solving self-efficacy scale. This instrument uses a Likert scale to measure students' self-efficacy in nine computational thinking components including algorithmic thinking; abstraction; problem decomposition; data collection, representation, and analysis; parallelization; control flow; incremental and iterative; testing and debugging; and questioning. Conducting this research addressed the question of whether the computational thinking skills present in Latin can transfer to a student's computational thinking self-efficacy which may affect STEM/computer science course achievement.

To test the null hypothesis that having a Latin language learning yields no significant influence on computer science students' self-efficacy in computational thinking and problem solving, a multivariate analysis of variance (MANOVA) test was utilized for this causal-comparative study. To test the null hypotheses that having a Latin language learning yields no significant influence on computer science students' abstraction, problem decomposition, data, parallelization, control flow, incremental and iterative, testing and debugging, and questioning skills self-efficacy, a separate ANOVA test were run for each computational thinking skill component.

The data did not meet of the necessary assumptions for a MANOVA test. The sample size for the non-Latin group was a concern at *n* = 20. The means from the descriptive statistics show that the non-Latin group outscored the Latin group in most of the computational thinking skills. Pillai's trace statistic from the MANOVA test showed no statistical significance in the computational thinking and problem solving scale. The individual results from the ANOVA tests showed no statistical significance for any of the nine subscales.

Table of Contents

List of Tables

CHAPTER ONE: INTRODUCTION

Introduction

Under the Obama Administration, the United States Department of Education predicted that between 2010 and 2020 there would be a 16% increase in math jobs, 22% increase in computer systems analyst jobs, 22% increase in systems software developer jobs, 36% increase in medical scientist jobs, and a 62% increase in biomedical engineer jobs ("Science, technology, engineering and math: Education for global leadership," n.d.). Due to this increased demand, education institutions have prioritized offering science, technology, engineering, and math (STEM) courses (Kennedy & Odell, 2014). Additionally, STEM courses, exercises, and programs are being introduced earlier in education programs – such as in the middle school level (Repenning, Webb, & Ioannidou, 2010) or even as early as the elementary school level (Bers, Flannery, Kazakoff, & Sullivan, 2014).

Computational thinking is a skill vital to preparing for a STEM field such as computer science. This parent skill encompasses such child skills as problem decomposition, algorithmic thinking, abstraction. Computational thinking is important to STEM because it fuses elements from mathematical, engineering, and scientific thinking (Wing, 2008). Deemed to include fundamental skills for not only computer scientists but all STEM professionals, computational thinking is a way of processing through which an individual can more efficiently design systems and understand concepts and human behavior (Wing, 2006). Key components of computational thinking involve decomposition, abstraction, and problem solving (Wing, 2006). These actions are not only performed by computer scientists, but by people in everyday situations. For example, when translating a complex Latin sentence, one might use decomposition to break down the sentence into smaller parts such as prepositional phrases, subordinate clauses, and main clauses. From there, utilizing abstraction eliminates extraneous information temporarily to find out what the authors is trying to say. By abstracting prepositional phrases, adjectives, and adverbs, the translator can more easily identify the main verb, its subject, and hopefully, the main idea of the sentence. Finally, problem-solving skills help the translator arrange the various clauses in a way that makes sense while conveying as accurately as possible what the author was trying to express. This might involve finding the correct usage of a noun case, applying the correct usage of the subjunctive mood, or any other decision that solves the puzzle that is a Latin sentence. For the purposes of this study, the Latin language discussed refers to classical Latin, the language of Cicero and Caesar.

While some students excel in the intricacies of computational thinking tasks, learning outcomes remain a challenge (Roscoe, Fearn, & Posey, 2014). Students who struggle with the computational thinking aspects of STEM courses may attribute their difficulties to low self-efficacy. In other fields, research suggests that self-efficacy is a non-academic antecedent to academic performance (Talsma, Schüz, Schwarzer, & Norris, 2018). Bandura (1997) defined self-efficacy as a set of "beliefs in one's capabilities to organize and execute the courses of action required to manage prospective situations" (p. 2). Self-efficacy is vital when approaching academic tasks that involve critical thinking, problem solving, and synthesis of information, because a self-efficacious student will more readily undertake challenging tasks than an inefficacious student (Zimmerman, 2000). Furthermore, Schunk and Pajares (2009) explained that self-efficacy affects students’ choices, effort, persistence, interests, and achievements. As it relates to computational thinking, an individual’s STEM self-efficacy can predict academic performance beyond his or her ability or previous achievement because such a confident person is motivated to succeed (Rittmayer & Beier, 2008).

Bandura (1997) presented four sources of self-efficacy: mastery experiences, vicarious experiences, verbal persuasion, and emotional and physiological states. Through mastery experiences, individuals gain self-efficacy in a particular area or task based on prior mastery in that same setting. For example, earning a high grade in a computer science course would boost the self-efficacy of a student trying to create a computer program. Through vicarious experiences, individuals gain self-efficacy in observing the plight and subsequent successes of people similar to themselves in similar settings. An example of this source is an increased self-self-efficacy working on a robotics project which a friend successfully completed at an earlier time. Verbal persuasion infuses self-efficacy into an individual because a person of influence possesses the ability to strengthen beliefs and encourage success. Students gain self-efficacy if a teacher or tutor expresses belief in a student when he or she is struggling. Finally, a person's emotional and physiological state directly affects the amount of self-efficacy he or she possesses. If a student is stressed because of a domestic situation, stress of an extracurricular activity, or is in any state of depression, this could affect his or her ability to be self-efficacious and thus hinder maximum performance in a given subject.

In researching how high school students can bolster their self-efficacy and achievement in STEM and computational thinking skills, one answer may be interdisciplinary selection. Because low self-efficacy in any subject is a concern pertaining to student belief and performance (Vekiri & Chronaki, 2008), it is imperative to find ways in which schools can boost the self-efficacy and achievement of students. It remains to be seen whether success in one course can equip a student with the confidence necessary to succeed in a perceived more difficult course. This study will examine whether strong self-efficacy in one subject can influence self-efficacy in another. Language learning may be one way to better support computational thinking because it has the ability to boost mental capabilities (Cooper, Wisenbaker, Jahner, Webb, & Wilbur, 2008). Classical Latin in particular requires learners to utilize the computational thinking skill of abstraction. Abstracting breaks down a complex sentence into clauses, phrases, and agreed words to be put back together and formed into a logical order of ideas. Using this skill allows a Latin translator to identify the grammatical elements of a sentence while relating it to other parts of the sentence (Settle et al., 2012). Furthermore, Latin may be especially suited to promote computational thinking because Latin word order does not map on to English word order, presenting an additional challenge to translators. This process involves both evaluation and abstraction (Settle et al., 2012) and is not as prominent in other Romance languages. For example, AP Latin requires students to translate the works of Virgil and Caesar. In addition, they must be familiar enough with the works of such Latin authors as Cicero, Livy, Catullus, Horace, and others to successfully sight translate. The syntax of these authors, their rhetorical style, and diction make translating their texts particularly challenging. Whereas in Spanish and French the linguistic goal is to speak fluently, the linguistic goal for Latin students is reading and translating accurately. Because Latin is not spoken, though many in the field do encourage spoken Latin (Avitus, 2018), teachers and learners of the language must pay attention to the language's various intricacies such as word order. Specifically, reading in Latin word order necessitates that a translator store ideas for grammatical interpretations of ambiguous words later to be clarified and held vis-à-vis its predictions (Russell, 2008). In fact, some Latin students are encouraged to use such techniques as annotating in place of translation to learn from errors and eliminate assumptions upon which students often rely (McFadden, 2008).

Problem of Practice Statement

While math and science are part of a core curriculum for any school, Latin is not emphasized in school curricula especially relative to Spanish. According to the American Councils for International Education (2017), in 2014-15 the total number of Spanish programs outnumbered the total number of Latin programs by approximately 8:1 (8,177 for Spanish, 1,512 for Latin). Furthermore, while high schools have shifted their focus toward implementing STEM programs, students are encouraged to take more STEM courses in lieu of other classes. For example, students at the Memphis high school used for this study frequently abstain inevitably from the fourth year of a foreign language to pursue engineering, robotics, and computer science in their senior year. This is by no means a unique circumstance. There is a precedent for schools making room for STEM courses by eliminating world languages. Further, this falls in line with the growing trend of college students foregoing humanities and humanities-related majors to focus on STEM courses in search of a field with more promising job prospects (Schmidt, 2018).

Due to the amount of computational thinking involved in Latin programs, their absence from most American schools means students lose an opportunity to acquire transferable computational thinking skills that may improve their self-efficacy. Latin is vital because translating a complex sentence in a step-by-step fashion exercises one’s algorithmic thinking, incremental skills, and iterative skills. Being able to accurately identify an ablative absolute phrase, for example takes practice. This construction, unique to Latin, enables a translator to more easily identify the main clause and groups together words of the same agreement.

Identification of this type comes more and more natural to a translator, thus making the process that much smoother and quicker. Testing and debugging skills are honed through the process of applying various translations of phrases and vocabulary words until the translator can develop a coherent reading of a passage. Problem decomposition is developed through a common but effective way of approaching any complex Latin section - breaking down a passage into smaller parts.

To date, research has begun to investigate the interdisciplinary nature of language learning and computational thinking. While there is some literature on this topic, more research is required to expand on this important topic. In one example, Settle et al., (2012) demonstrated that computational thinking can easily be practiced in Latin language learning. In their study, the authors introduced and enhanced computational thinking activities in middle school and high school courses. Within Latin courses, they encouraged grammar notation, sentence diagramming, and metaphrasing - all of which involve computational thinking. Within grammar notation, a translator understands the morphological information of each word and applies the necessary components to the translation, similar to the application of the computational thinking skill abstraction. In sentence diagramming, a translator breaks down the various clauses, phrases, and agreed words, akin to the skill of problem decomposition. Metaphrasing relates the morphology of a word to English word order while also using its form to predict the function of other words in the sentence (Knudsvig, Seligson, & Craig, 1986), comparable to parallelization and control flow. Not only does Latin appear to be underutilized in school curricula, but its absence forfeits an avenue where computational thinking can be practiced. As a result, additional opportunities to boost one’s self-efficacy and subsequent achievement in STEM are lost. If students learn and have success in computational thinking, they may develop self-efficacy in computational thinking, a potential positive influence in STEM courses.

Purpose Statement

The purpose of this causal comparative study is to contrast the computational thinking self-efficacy of computer science students with no exposure to Latin to computer science students with exposure to Latin at a Memphis all-boy high school. Sex is one control variable, because only boys will be included in the study. Grade level will also be controlled in this study as it may be a confounding variable. Causal comparative is a group comparison design which Gall, Gall, and Borg (2010) say is “useful for exploring causal relationships, even though they cannot confirm results to the degree that experimental research can” (p. 242). Based on the description, the causal comparative design is most appropriate for this research as the independent variable will be the type of Latin language learning experience, i.e., up to 6 years total of Latin language learning, versus 0 years of Latin language learning. Additional data on the number of years enrolled in other foreign languages will also be collected. The dependent variable will be mean scores of items found on a computational thinking and problem solving self-efficacy scale. This instrument uses a Likert scale to measure students’ self-efficacy in nine computational thinking components including algorithmic thinking; abstraction; problem decomposition; data collection, representation, and analysis; parallelization; control flow; incremental and iterative; testing and debugging; and questioning (Weese & Feldhausen, 2017). Questions on this scale are uniquely tailored for those in computer science classes. Conducting this research will address the question of whether the computational thinking skills present in

Latin can transfer to a student's computational thinking self-efficacy which may affect STEM/computer science course achievement.

Question(s)

Data for this study will be gathered from computer science students who have practice exercising algorithmic thinking, abstractive thinking, problem decomposition, and other computational thinking skills through their experience in computer science classes. In addition, this study will compare the computational self-efficacy of computer science students who have taken Latin and those who have not taken Latin. If the research reveals the Latin does impact computational thinking self-efficacy, schools could find ways of mandating more than two years of a foreign language while also encouraging more students to take Latin. There is one main research question for this study along with nine sub-questions which align with the computational thinking scale to be used in the data collection process.

The following will serve as research questions for this study:

Research Question 1: To what extent does exposure to Latin education influence computer science students' self-efficacy in computational thinking and problem solving?

a) To what extent does exposure to Latin education influence computer science students' *algorithmic thinking* self-efficacy?
b) To what extent does exposure to Latin education influence computer science students' *abstractive thinking* self-efficacy?
c) To what extent does exposure to Latin education influence computer science students' *problem decomposition* self-efficacy?

d) To what extent does exposure to Latin education influence computer science students' *data collection, representation, and analysis* self-efficacy?

e) To what extent does exposure to Latin education influence computer science students' *parallelization* self-efficacy?

f) To what extent does exposure to Latin education influence computer science students' *control flow* self-efficacy?

g) To what extent does exposure to Latin education influence computer science students' *incremental and iterative* self-efficacy?

h) To what extent does exposure to Latin education influence computer science students' *testing and debugging* self-efficacy?

i) To what extent does exposure to Latin education influence computer science students' *questioning* self-efficacy?

Null Hypothesis

Null Hypothesis 1. Having a Latin education yields no significant influence on computer science students' self-efficacy in computational thinking and problem solving.

a) Latin education exposure yields no significant influence on computer science students' *algorithmic thinking* self-efficacy.

b) Latin education exposure yields no significant influence on computer science students' *abstractive thinking* self-efficacy.

c) Latin education exposure yields no significant influence on computer science students' *problem decomposition* self-efficacy.

d) Latin education exposure yields no significant influence on computer science students' *data collection, representation, and analysis* self-efficacy.

e) Latin education exposure yields no significant influence on computer science students' *parallelization* self-efficacy.

f) Latin education exposure yields no significant influence on computer science students' *control flow* self-efficacy.

g) Latin education exposure yields no significant influence on computer science students' *incremental and iterative* self-efficacy.

h) Latin education exposure yields no significant influence on computer science students' *testing and debugging* self-efficacy.

i) Latin education exposure yields no significant influence on computer science students' *questioning* self-efficacy.

Definitions

The following are definitions of words and phrases pertinent to this research:

Algorithmic Thinking. An algorithm is a method to solve a problem that involves clearly defined steps or instructions. Algorithmic thinking is the ability to analyze and specify a problem precisely and subsequently construct a correct algorithm to resolve the said problem (Futschek, 2006).

Abstraction. Abstraction strips down a problem and captures common characteristics of one set to be used to represent all other instances (Lee, Martin, Denner, Coulter, Allan, Erickson, Malyn-Smith, & Werner, 2011).

Computational Thinking. Computational thinking involves formulating problems in a way that computers and other tools can be used to solve them. It entails implementing, analyzing, and identifying solutions. From that point, one can generalize and transfer the problem solving process to other scenarios (Barr, Harrison, & Conery, 2011).

Control Flow. Control flow directs an algorithm's steps when complete (Weese & Feldhausen, 2017). It is the order in which steps are executed.

Data. Data refers to the ability to collect, represent, and analyze information (Weese & Feldhausen, 2017).

Incremental and Iterative. Incremental and iterative methodology forms constraints with each attempt of problem solving until a solution is reached that satisfies the greatest number of constraints (Jonassen, 2008). It directs programmers to build a program step by step as opposed to all at once (Weese & Feldhausen, 2017).

Classical Latin Language. Latin was the language of ancient Latium (central Italy) and the Roman Empire. The earliest Latin inscriptions have been dated as far back as the seventh century B.C. After the sixth century A.D. (and after the fall of the Roman Empire), Latin branched off into an ecclesiastical form and a colloquial form spoken among Roman provinces. These languages would evolve into the Romance languages of Italian, Spanish, Portuguese, French, and Romanian (Bennett, 1908). The Latin to which this research refers is the language of Cicero and Caesar.

Parallelization. Parallelization is the simultaneous processing or execution of a task (Weese & Feldhausen, 2017).

Problem Decomposition. Problem decomposition breaks down problems into smaller parts that can be more easily solved (Barr & Stephenson, 2011).

Questioning. Questioning is using each part of a code as opposed to using code not easily understood (Weese & Feldhausen, 2017). This process involves arriving at answers and solutions sought through inquiry.

Self-Efficacy. Self-efficacy is a person’s contribution to their own functioning through personal agency. This agency is centered around a belief that their capabilities can control their level of functioning and other events affecting their lives (Bandura, 1993).

STEM. STEM is an acronym for science, technology, engineering, and mathematics specifically in the context of education. This distinction is important lest it seem to be a reference to the fields which scientists, engineers, and mathematicians occupy (Sanders, 2009).

Testing and Debugging. Testing and debugging entails analyzing and fixing problems immediately during development (Weese & Feldhausen, 2017).

Transfer. Transfer occurs when learning in one context enhances a related performance in another (Perkins & Salomon, 1992).

CHAPTER TWO: REVIEW OF THE LITERATURE

Introduction

Computational thinking is a thought process that fosters learning competencies such as algorithmic thinking; abstraction; problem decomposition; data collection, representation, and analysis; parallelization; control flow; incremental and iterative thinking; testing and debugging; and questioning (Weese & Feldhausen, 2017) for students at multiple grade levels. This study seeks to investigate if, while language learning entails interdisciplinary benefits, Latin in particular promotes computational thinking self-efficacy. This study will take place at a Memphis high school where students must complete four years of math, three years of science, at least two years of a foreign language, and various other requirements. The science department offers students up to four years of computer science. Within the foreign language department, the school offers up to four years of French, Spanish, or Latin, the latter two affording an AP credit in the fourth year.

Latin translation is distinctive because it seldom requires the same wording sequence as English and other related languages. An accurate translation of Latin demands a strict adherence to grammatical rules that are, at times, more flexible in other languages. When students translate Latin passages into English, they are exercising computational thinking. In fact, it is beneficial to approach grammar and translation as one would approach a science or math problem. Chater and Manning (2006) suggested that probability and plausibility should factor into what they call "computational linguistics." The authors explained that:

> In parsing, for example, probability helps resolve the massive local syntactic ambiguity of natural language, by focusing on the relatively small number of potential parses with

> significant probability (given what is known about the frequency of different structures across a corpus). Similarly, probabilistic methods in learning dramatically narrow the infinite set of grammatical rules that could generate a given set of sentences or structures. (Chater & Manning, 2006, p. 337)

By parsing a word, Latin translators are applying the computational thinking skill of abstraction. For example, the word *amamur* in Latin parses as a 1st person plural, present tense, active voice word. The fact that the word is six letters, three syllables, and contains a long vowel is not necessary information for the purpose of translating.

Because of the overlap between computational thinking and language learning, additional study is needed to define the link between computational thinking and Latin learning. To date, research has not adequately recognized the positive role that these skills can have on students' self-efficacy. This study aims to address this research gap and contribute to the field's understanding of computational thinking and interdisciplinary instruction through language learning. What follows here is a review of the scholarly literature on meetings of STEM courses, computational thinking, self-efficacy, and language learning. This review will begin by laying out the theoretical context. Next, it will explain how research defines computational thinking, where it occurs based on age group, and the skill's cross-curricular benefits. The review will then shift to how scholarship defines self-efficacy theory, identifies predictors for self-efficacy, and elaborates on self-efficacy's potential benefits. Finally, this review will detail the benefits of language learning with particular emphasis on Latin language learning.

Theoretical Context

Self-efficacy is a person's contribution to his or her own functioning through personal agency. This agency centers around a belief that personal capabilities can control one's level of functioning and other events affecting one's life (Bandura, 1993). Bandura, who has written on self-efficacy theory since the 1970s, indicated that a person's efficacy requires cognitive, social, and behavioral capabilities to act in concert to serve innumerable purposes (Bandura, 1982). Perceived self-efficacy not only influences choice of activities, but also determines how much effort a person will expend in the face of adversity (Bandura, 1977). As the progenitor of self-efficacy theory, Bandura (1995) explained how self-efficacy can be developed in four main ways: mastery experiences, vicarious experiences, social persuasion, and physiological and emotional states.

Bandura (1995) believed the most effective way of generating strong self-efficacy is through mastery experiences. Mastery experiences are achieved through an individual experiencing personal success. Researchers have demonstrated that mastery experience is the most influential in boys' and men's STEM career self-efficacy (Zeldin & Pajares, 2000). In vicarious experiences, on the other hand, individuals gain self-efficacy by observing their peers achieve success. These experiences encourage individuals to believe more deeply in their own abilities. Social persuasion develops individuals' self-belief by means of verbal support from family members, role models, or anyone else of influence. This also entails placing people in situations intended for success and steering them away from spaces where failure seems inevitable. The final way to develop self-efficacy is by creating the appropriate physiological and emotional environment. Possessing the right state of mind is, therefore, crucial to the fostering of self-efficacy.

Bandura (1986) explained self-efficacy as a person’s confidence and beliefs concerning their capabilities to perform tasks or activities. Confidence and belief are crucial when approaching academic tasks that involve critical thinking, problem solving, and synthesis of information. Without self-efficacy, students of any age will choose to avoid courses and fields that entail such practices. Along the same lines, Schunk and Pajares (2009) describe how self-efficacy affects students’ choices, effort, persistence, interests, and achievements. Bandura (1993) argued the existence of a causal relationship between self-efficacy that directly affects memory performance and indirectly affects memory performance through raising cognitive effort. Hatlevik, Throndsen, and Gudmundsdottir (2018) viewed Bandura’s notion of self-efficacy as the idea that personal belief is a direct determinant of an individual's behavior and actions.

In contrast to a student’s internal beliefs, a student’s potential self-efficacy can be affected by external factors such as family relationships. Gonzalez-DeHass, Willems, and Holbein (2005) argued that involved parents who express praise, encouragement, and expectations produce students motivated to learn and possess positive self-efficacy. Bleeker and Jacobs (2004) found that parents possess their own set of beliefs and stereotypes about their child’s abilities.

In recent years, Schunk and Maddux have built on the work of Bandura’s initial definition of self-efficacy. Schunk argued that self-efficacy is not the only determinant of academic achievement; requisite skills must be present along with outcome expectations and the perceived value of these expectations (Schunk, 1991). Maddux linked self-efficacy with expectancy-value theories. These theories “maintain that the tendency to perform a behavior is the product of the reinforcement value of the expected outcome and the expectation that a specified behavior or behaviors will produce that outcome" (Maddux, Norton, & Stoltenberg, 1986, p. 783).

Review of the Literature

The scholarship of Bandura, Maddux, and others has thoroughly explained self-efficacy and its impact in academic and non-academic settings. Requiring mandatory language learning credits for multiple years within a larger curriculum including interdisciplinary STEM courses can produce the mastery experiences that maximize a student's self-efficacy. All the while, both fields could potentially be viewed as equally valued in their own ways. Latin courses in particular are well-suited to maximize self-efficacy because of the computational skills involved with studying and translating the language. Latin uniquely develops computational thinking skills in addition to its other benefits of vocabulary retention, grammar retention and standardized test prep.

This literature review will begin with what has been written about the definition of computational thinking.

What is computational thinking?

The term computational thinking has been popularized in part by Wing (2006) who explained that it "involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science" (p. 33). In addition, computational thinking entails problem solving, critical thinking, and "conceptualizing at multiple levels of abstraction, defining and clarifying a problem by breaking it down into relational components, and testing and retesting plausible solutions" (Miller, Soh, Chiriacescu, Ingraham, Ramsay, & Hazley, 2013, p.1). The advent of this term coincides with the recent emphasis on STEM courses. Computational thinking has evolved to include algorithmic thinking, data collection skills, parallelization, control flow, questioning skills and incremental and iterative skills. Computational thinking is a way of thinking generated and facilitated by computation, whether by a mechanical device or the human brain. It is a link to cognitive competencies found not only in science and math but in everyday life (Yaşar, 2018). As educational systems increasingly emphasize 21st century learning skills (Lombardi, 2007) that reflect recent advances in media, science, and technology, Wing (2006) argued that computational thinking should be a fundamental skill for everyone regardless of discipline.

Computational thinking benefits

In terms of computational thinking and its effect on STEM and non-STEM courses, elements such as abstraction, algorithmic notions of control flow, problem decomposition, parallel thinking, and conditional logic, comprise computational thinking and form the basis for the curricula supporting it (Grover & Pea, 2013). As early as 1962, computer scientist Alan Perils promoted learning programming as part of a liberal education. He added that "programming was an exploration of process, a topic that concerned everyone, and that the automated execution of process by machine was going to change everything" (Guzdial, 2008). Wing (2006) also argued the importance of computational thinking across multiple disciplines. It enables students to acquire new problem-solving strategies and to find new solutions for the real and virtual world. Wing (2006) advocated for computational thinking because it leads to abstract and algorithmic approaches to problems which are ubiquitous in real-world situations.

In addition to the applications of algorithmic thinking and abstraction, computational thinking has been explored in terms of its relation to creative thinking. Creativity in schools reaches beyond art, music, and writing classes. Instead, creative thinking, distinct from creativity, is a cognitive behavior that can integrate into math, science, and social studies in that it is comprised of critical thinking components (DeSchryver & Yadav, 2015). Additional research by Miller et al. (2013) found that creative competency was significantly correlated with computational thinking. When paired with existing knowledge, critical thinking can solve complex technological problems. Settle and Perkovic (2010) discussed similar benefits saying that computational thinking not only enhances creativity and innovation, but also offers new ways to view social and physical phenomena. Miller, Soh, Chiriacescu, Ingraham, Shell, and Hazley (2014) found that incorporating creative thinking exercises can improve computational thinking. Such exercises can boost student achievement in computer science courses.

In addition to interdisciplinary benefits, research also shows that computational thinking through computer-assisted problem solving supports an individual's real-world problem-solving abilities. Voskoglou and Buckley (2012) investigated whether using computers to solve problems would enhance a student's ability to solve real world problems. In their studies, the authors found that computational thinking includes analysis, logic, data modelling, testing, and algorithmic thinking. In their experiment, 90 prospective engineers at the School of Technological Applications of the Graduate Technological Educational Institute (TEI) of Patras, Greece attended the course "Higher Mathematics I" during their first term of studies. Despite no programming experience, the experiment showed strong evidence that the use of computers as a tool for problem solving augments a person's ability to solve real-world mathematical problems. In addition, "critical thinking plays a central role in knowledge acquisition and creation, in computational thinking and thus in real complex technological problems" (Voskoglou &

Buckley, 2012, p. 41). If critical thinking and problem-solving skills are inextricably tied with computational thinking, these skills may relate to other fields where such thought processes are essential.

Another interdisciplinary benefit of computational thinking fosters skills for those within the broader field of science. Computational models have been used for advancements in physics, engineering, and other sciences, leading to various fields to develop a computational branch (Denning, 2017). In their study, Sengupta, Kinnebrew, Basu, Biswas, and Clark (2013) concluded that it is beneficial to synthesize computational thinking with science curricula. In doing this, the learning curve for mathematical and scientific principles is lowered because these concepts center around intuitive computational mechanisms. Students also gain practice in scientific methodology through computational modeling. Given these connections to science and science-related fields, computational thinking provides students with tools essential to execute tasks in these areas.

Computational thinking in STEM K-16

Trends show that computational thinking has increasingly been implemented across various levels of education including grade school (Tran, 2018), middle school (Wolz, Stone, Pearson, Pulimood, & Switzer, 2011), high school (Ahamed, Brylow, Ge, Madiraju, Merrill, Struble, & Early, 2010), and university settings (Hambrusch, Hoffmann, Korb, Haugan, & Hosking, 2009). Qualls and Sherrell (2010) contended that computational thinking techniques should be present in any discipline requiring problem solving thereby making it a primary skill. For this to happen, computational thinking skills should be taught as early as elementary school and continue throughout a child’s education. Yadav, Mayfield, Zhou, Hambrusch, and Korb

(2014) further argued that, in order to maximize the benefits of computational thinking and spark student interest, it must be integrated into K-12 curricula. Lu and Fletcher (2009) asserted that children as young as grade three have already encountered computational thinking. When encountering multiplication, reading comprehension, and charting, students are exercising computational processes. For this reason, Lu and Fletcher (2009) advocated identifying computational thinking vocabulary during primary education. Bers, Flannery, Kazakoff, and Sullivan (2014) recognized that children as young as four possessed the capability of building simple robotic projects and demonstrated the tangible benefits gained from such exercises. For example, the authors discussed the TangibleK Robotics Program curriculum that tests the limits of children’s ability to learn robotics and programming. As a result of the curriculum, researchers found that students excelled in activities that involved robotics, engineering design, and programming. Because of the demonstrable benefits, the practice of integrating computational thinking instruction within curricula will grow over time.

Due to the ubiquitous application of computing processes, researchers encourage computational thinking to be taught in pre-college curricula outlets (Lu and Fletcher, 2009). Werner, Denner, Campe, and Kawamoto (2012) targeted computational thinking at the middle school level in computer science. But while other studies focus on traditional K-12 contexts, the researchers conducted their study after school or during elective periods in gaming and programming classes. The authors found that elective technology and computer game courses are promising strategies to introduce computational thinking and to observe learner growth in terms of algorithmic thinking and abstraction. In a similar study, Lee et al. (2011) presented the initiative Growing Up Thinking Scientifically (GUTS) to middle school students. This program required students to utilize abstraction and other computational thinking skills to design models to be later used to run simulations. Lee et al. (2010) also studied a robotics project given to middle and high school students where they embed code for robots and other physical devices.

There is also a push for implementing computational thinking skills at the college level; however, courses most germane to computational thinking typically lack a standardized method of teaching computational thinking skills within their curriculum (Manson & Olsen, 2010). The benefit of a standardized curriculum is that it provides core competency requirements with appropriate coursework. The authors argued that computational science programs embrace providing courses to established interdisciplinary majors, enabling students to exercise computational thinking skills. Computational science faculty at The Richard Stockton College of New Jersey expanded computational content in cognate courses in the physics programs. Manson and Olsen (2010) reported that including computational projects in mechanics courses, a computational science cognate, helps students academically. Computational thinking is taught at multiple levels of education and produces positive outcomes for those involved.

Computational thinking self-efficacy

The above research suggests the implementation of computational thinking across curricula is broadening. In response, researchers are exploring strategies and methods to engender learning and performance outcomes in computational thinking. Prior studies show that self-efficacy is important for learning because of its potential effect on student performance (Vekiri & Chronaki, 2008). In the case of computational thinking, which can be predicted by self-efficacy (Ramalingham, LaBelle, & Wiedenbeck, 2004), the learner’s perception of his or her abilities will shed light on why frustration may set in and result in the rejection of a course

(Tsai, Wang, & Hsu, 2018). Because self-efficacy correlates with learning performance, Tsai, Wang, and Hsu (2018) sought to study how self-efficacy can affect computational thinking competencies. The authors found that computer programming experience plays a role in computational thinking self-efficacies. They concluded that “increasing the programming experience may greatly enhance students’ self-perceptions of logical thinking, algorithm development, debugging, control, and cooperation abilities for computer programming” (Tsai, Wang, & Hsu, 2018, p. 8).

Although studies are limited about the role of self-efficacy and computational thinking, prior studies have explored the link between self-efficacy and technology in general. Hatlevik et al. (2018) discussed the relationship between self-efficacy in information and communication technology (ICT) and computer information literacy (CIL). They found that variables in a person’s background (such as having computers at home) are positively correlated with ICT self-efficacy and that ICT self-efficacy is a predictor for CIL achievement. In their study, Hatlevik et al. (2018) asked how students' personal characteristics and background contextual variables affected their ICT self-efficacy and CIL and explored what the relationship between the two. They explained that it is important to know what role self-confidence plays in the achievement of students and how ICT self-efficacy is correlated to computer and information literacy. In addition to predicting CIL, knowing the background of a person can offer insight on why some people are more motivated, self-driven, and confident than others. Hatlevik et al. (2018) concluded based on the analyses that autonomous learning is positively correlated to ICT self-efficacy. As in the case of prior studies, this study underscores the role an individual’s background and experience in a field plays in their self-efficacy with executing tasks in that same field.

Additional studies have analyzed how intervention affects a student’s self-efficacy. Webb and Rosson (2013) described an outreach program for middle-schoolers involving computer activities that fostered computational thinking skills such as problem solving, debugging, and abstraction using scaffolded examples. Students found the experience positive and reported an increase in their self-efficacy in the three computational thinking skills. In an attempt to promote computational thinking in collegiate engineering students, Shell, Hazley, Soh, Miller, Chiriacescu, and Ingraham (2014) deployed the Computational Creative Exercises initiative in an introductory computer science course. The authors compared the computer science knowledge and concept test scores of those who received the initiative with those who did not. They found that students who received the initiative not only scored higher on the test but also possessed a significantly higher self-efficacy in applying computer science in their field. These studies underscore the fact that students who receive an intervention featuring computational thinking skills generally have positive experiences and enjoy a boost in their computational thinking self-efficacy.

Another beneficial aspect of self-efficacy is its potential link with motivation. Schunk (1991) stated that a “heightened self-efficacy sustains motivation and improves skill development” (p. 213). Repenning, Webb, and Ioannidou (2010) suggested to teachers that student game design produces motivational benefits. Such activities not only appeal to a broader range of students but also satisfy such learning outcomes as problem solving and accessing, compiling, and integrating information (Repenning et al., 2010). The increased motivation also extends to students who typically avoid math and science courses. Those who feel they are devoid of the "science/math gene" often reconsider their aversion to the subjects after creating a robot, programming games, or succeeding in engineering classes. The authors suggested including computational thinking skills in existing required courses. If computational thinking is to be expanded, additional understanding is needed about how to design curricula in ways that simultaneously balance challenges and learning outcomes.

Computational thinking and language learning

As noted earlier, computational thinking is a multifaceted phenomenon that includes algorithmic thinking, abstraction, problem decomposition, etc. Research is trying to identify how these competencies can be enhanced through other disciplines such as language learning. Similar to computational thinking, language learning requires the learner to encounter structures, forms, and step-by-step processes. It is a train of thought that enables an individual to efficiently and systematically process information (Lu & Fletcher, 2009). Students exercise computational thinking when diagramming and parsing sentences by dividing phrases into sub-phrases and other linguistic categories (Lu & Fletcher, 2009). Barr and Stephenson (2011) submitted that computational thinking concepts and capabilities enable a student to perform such language arts tasks as linguistic analysis of a sentence, pattern identification of different sentence types, rhetorical figure usage, etc. Years ago, Clements (1999) found that computational thinking yields a positive effect on language mainly reading comprehension, reading skills, and foreign language acquisition. More recently, Weng and Wong (2017) tried to improve computational thinking learning outcomes through diagrammatic tools. They found that introducing computational thinking to English dialogue learners motivated primary school students to study English dialogue. Students acquired graphic programming language skills that could be transferred to English language assignments.

Given the emphasis on iterative problem solving, linguistics and computer science have been linked for years with mechanical translation and computational models of language learning and parsing (Niyogi, 2006). Language acquisition necessitates both the development of a system of rules and a language learning algorithm. Indeed, language's inherently mathematical qualities have led researchers to empirically explore the link between the similar skillsets in computational thinking and language learning. Niyogi (2006) hypothesized a link between linguistics and computer science claiming that language plays an integral part in in the investigations of logic and computer science. A byproduct of the relationship between computational processes and linguistics is the field of computational linguistics where computer systems produce, understand, and interpret language. By analyzing language processes, computers have the ability to generate and translate natural language (Grishman, 1986). In summary, the use of programming languages, computer-assisted language learning, computational linguistics, and any computer science intervention that yields positive linguistic effects support connections between computational thinking skills and language learning.

Benefits of language learning and challenges to self-efficacy

Self-efficacy plays a crucial role in an individual's ability to attain the skillsets needed for language learning. Specifically, research has indicated that foreign language acquisition is subject to the effects of a student's well-being. Anxiety falls under one of Bandura's four sources of self-efficacy: physiological state. In this state "people judge their level of anxiety and vulnerability to stress" (Bandura & Adams, 1977, p. 288). Scida and Jones (2017) examined how emotions impact language learning, retention, performance, skill development, and experience. In addition to the physiological manifestations such as sweating, tenseness, and trembling, foreign language anxiety leads to the psychological changes of low self-confidence and low self-efficacy (Scida & Jones, 2017). A low self-efficacy can predict academic achievement and performance in foreign classes and beyond (Graham & Weiner, 1996).

Research has explored the detrimental effect of anxiety in foreign language courses, one of many subjects that can cause such harm. MacIntyre, Noels, and Clément (1997) pointed out that vocabulary retention, grammar, and test performance all correlate with foreign language anxiety. Bamber and Schneider (2016) described how levels of stress and anxiety affect one's memory, concentration, attention, and problem-solving ability. Horwitz, Horwitz, and Cope (1986) explained that students who suffer from foreign language anxiety may procrastinate homework, skip classes, and are reluctant to engage in classroom activities. However, this anxiety will vary depending on the course, level, and language (Scida & Jones, 2017). Low self-efficacy, which sees its effects in foreign language learning and in other fields, produces detrimental effects on an individual's well-being and classroom performance.

Cooper et al. (2008) said that language study has a positive effect on academic performance and correlates to increased scores on standardized tests, which include learning outcomes related to language arts, science, and math skills. Foreign language learning could have additional benefits beyond vocabulary acquisition, fluency, conversational skills, etc. Bilingualism is one of these benefits, especially in the changing demographics of a country such as the United States. In addition, Cooper et al. (2008) argued that students who learn a foreign language utilize the critical-thinking skills described in Bloom's taxonomy of thinking processes.

Such processes as recall, comprehension, application, analysis, synthesis, and evaluation are at the forefront of the language learning process. Cooper et al. (2008) analyzed the SAT and PSAT scores from 16 high schools in Gwinnett County, Georgia. Within the range of possible PSAT verbal scores, students who took a foreign language outperformed students who did not. Students who took level three of their foreign language by the end of their junior year outperformed those who reached only levels one or two.

While language learning offers students a number of positive learning outcomes, Latin offers a unique benefit to students in that it builds on the reading skills and English vocabulary of students varying in background and skill levels (Masciantonio, 1977; Holliday, 2012). In fact, decline in Latin enrollment across the American education system has been linked to national awareness in illiteracy (Sparks, Ganschow, Fluharty, & Little, 1995). Anderson (1975) alluded to a report by the Department of Foreign Languages in D.C. that said the positive verbal effects of one year of Latin "far surpassed" the benefits of two or three years of Spanish or French instruction (p. 45). There is also data that supports an increase in spelling test scores for Latin pupils. As in prior studies, these studies illustrate how Latin learning produces interdisciplinary benefits for students.

The benefits of Latin and language learning resonate beyond vocabulary and reading skills and may be beneficial for improving self-efficacy in computational thinking. Decoding the writings of Cicero, for example, forces a translator to break down a passage as if it were a math equation. Probabilistic models can explain how humans process and acquire language. Probabilistic language processing presupposes that these models can infer how sentences should be parsed and how ambiguous words should be interpreted (Chater & Manning, 2006). Chater and Vitanyi (2003) said that language acquisition "involves finding patterns in linguistic input, in order to determine the structure of the language" (p. 19). Horning (1971) argued that if sentences are treated as independent, identically distributed data, grammars are learnable within a statistical tolerance. In the interpretation of Ciceronian phraseology, a translator is utilizing probability in that he or she is determining the likelihood of a particular translation over another based on previous translation data. Viewing word morphology as data triggers a translator to dive into a deep analysis of the data, syntax of the sentence, and the word pictures formed by the verbal arrangement (Markus & Ross, (2004). The computational thinking skills of data analysis and parallelization are often utilized in Latin prose and poetry. In the sentence *cum illi aut ex arido aut paulum in aquam progressi* from Caesar's *Commentarii De Bello Gallico,* knowing that *illi* and *progressi* are both nominative case, plural, and masculine alerts the reader to first translate the two words together. Then, both *illi* and *progressi* reinforce to the reader that each word in between belongs in a clause to be translated inside the word brackets. Simultaneously, the reader must recognize the correlatives *aut…aut…*within the word brackets distinguishing the two prepositional phrases. In addition, an analysis of *progressi* identifies it as a participle, thus subordinated, and not to be viewed as the main clause that comes later in the sentence. With the expectation of a nominative plural verb (main or subordinate) and the expectation of a main verb following a subordinate clause allows a reader to exercise the concept of metaphrasing. As earlier discussed, this skill has been linked to computational thinking.

Summary/ Solution

Based on the research cited above, computational thinking enables advances in science and engineering (Landau, et al., 2014). Computational thinking is also associated with success in math and technology where students need to use algorithmic reasoning and step by step scientific methodology. Computational thinking involves utilizing higher level skills such as critical thinking, problem solving, and conceptualization (Miller et al., 2013). According to Weese and Feldhausen (2017), computational thinking also centers around algorithmic thinking; abstraction; problem decomposition; data collection, representation, and analysis; parallelization; control flow; incremental and iterative; testing and debugging; and questioning. These qualities have led to the implementation and training of computational thinking skills within diverse curricula.

An individual's self-efficacy can affect his or her personal agency which in turn controls various life functions (Bandura, 1993). By practicing the interdisciplinary components of a subject in multiple arenas, students can boost their self-efficacy. Foreign language learning, for example, develops skills on Bloom's taxonomy such as problem solving, critical thinking, and analysis (Cooper et al., 2008). Cooper et al. (2008) also assert that language learning helps academic performance in other areas beyond the target language. Due to its uniqueness, Latin language learning benefits students with an increased self-efficacy in computational thinking skills.

Having reviewed the research on the benefits of computational thinking, the role of self-efficacy, and self-efficacy improvement, this study will investigate what potential effect taking Latin has on computational thinking self-efficacy. The benefits of Latin have been researched for years. From GPA increases, to improved Romance language acquisition, and English skill enhancement (Holliday, 2012), Latin's contributions to cognitive skills are well documented. However, no research has identified what exactly makes Latin so valuable in its ability to promote problem solving, critical thinking, and scientific methodology. These skills, analogous to those in computational thinking, reinforce higher level thinking present in STEM classes. Succeeding at Latin translation could bestow self-efficacy necessary to master computational thinking. The above research describes the literature about computational thinking, the theory of self-efficacy and concludes with a review of language learning scholarship.

CHAPTER THREE: METHODOLOGY

Introduction

The previous chapters have laid out the benefits of language learning with a particular emphasis on Latin. Chapter 2 summarized literature describing the importance of computational thinking in a K-12 setting. Chapter 2 also explained the theory of self-efficacy and how it influences a person's ability to succeed in various academic endeavors.

Having discussed the literature, this chapter will address the methodology needed to execute the research. The purpose of this causal-comparative study was to use a computational thinking and problem solving self-efficacy scale to investigate whether the computational thinking skills present in Latin can increase a high school student's general computational thinking self-efficacy. This chapter will explain the research inquiry. It will lay out the plan, participants, setting, instrumentation, data collection process, and data analysis.

The Investigation Plan

The research design to investigate this problem of practice was a quantitative, causal-comparative design. Causal-comparative design is best suited for this research's goal to find whether a relationship exists between Latin language learning and computational skill self-efficacy. A causal-comparative study is a group comparison design which Gall, Gall, and Borg (2010) say is "useful for exploring causal relationships, even though they cannot confirm results to the degree that experimental research can" (p. 242). Moreover, the causal-comparative design aims to examine the relationship between variables after they have been manipulated or as they naturally occur in an environment (Gall, Gall, & Borg, 2010).

Similar studies have used the causal-comparative design. For example, Noormohamadi (2009) used an ex post facto design to study learning strategies for students with language anxiety. The independent variable was language anxiety (high or low) and the dependent variable was strategy use. The study found that low anxiety students made more use of learning strategies than high anxiety students. This study comprised of computer science students with varying levels of exposure to Latin. The results of this study shed light on whether Latin language learning has any effect on computational thinking self-efficacy.

Participants

A nonprobability sampling method was used for this study where "the researcher selects individuals because they are available, convenient, and represent one characteristic the investigator seeks to study" (Creswell, 2008, p. 155). More specifically, this research utilized a convenience sampling due to the accessibility of the sample to the researcher. The sample of students were enrolled at a Memphis school where I teach Latin I, Honors Latin III, and AP Latin.

Students in computer science classes in their freshman, sophomore, junior, or senior year were asked to participate in the study. There were 59 total students enrolled in computer science in the 2018-2019 school year. Seven additional students in the robotics club comprised the sample. Each student at this high school is required to take four years of math and four years of science; however, students may elect to join the STEM program where they take additional classes in engineering and computer programming. This track begins during a student's freshman year and will continue until graduation. Students take such classes as Principles of Engineering, Aerospace Engineering, Biomedical Science, Computer Science Essentials, and Computer Science Principles. Some classes include students of multiple grades.

Future studies might want to investigate more diverse populations due to the fact that most of the participants in this study were white males - reflecting both the demographics of the school and the computer science class population. Before students completed the instrument, they were asked to fill in information about their grade, number of years taking a foreign language, and numbers of taking Latin. This Memphis high school requires each student to take a minimum of two years of a foreign language. Students can choose from Spanish, Latin, and French. Those who take Latin may elect to end their studies at the completion of their second year or opt into a third year of either standard Latin III or Honors Latin III. The Honors Latin III course is the prerequisite to AP Latin in the fourth year of their foreign language usually taking place during a student's senior year. Of the 66 students in the sample, about half were enrolled or had been enrolled in Latin.

Setting

Located in Memphis, TN, this independent high school maintains an annual enrollment of around 900 students. This setting was chosen because of its accessibility to me. Each student is issued a laptop equipped with a personalized Google account and programs such as Microsoft Word. Therefore, students have accessibility to a computer with internet throughout the school day. Data collection took place in several computer science classes throughout the day. The computer science program includes such courses as Introduction to Computer Science, Honors Computer Science Applications, Computer Science Principles, and IT Support Fundamentals. In these courses, students use Python, a programming language, as an essential tool. Development of computational thinking skills are fostered by exercises in programming, app creation, and simulation. The computer science classes take place in a computer lab with desktop computers and standard classroom accessories such as white boards, tables, and a teacher's desk. In addition to the computer science classes, data was collected from the afterschool robotics club which also meets in a school classroom.

Instrumentation

This study focused on comparing students who have any Latin background with students who have no Latin background. For the main research question and the nine sub-questions, I asked students to complete the 23-item computational thinking and problem solving self-efficacy scale developed by Weese and Feldhausen (2017) in their study on a 5th-9th grade STEM outreach program. The authors acknowledged that a threat to the internal validity of their experiment was the small sample size. The pre-survey and post-survey yielded a Cronbach's alpha of .872, showing it to be reliable.

The computational thinking and problem solving self-efficacy scale includes 23 items. The 23 items measure algorithmic thinking; abstraction; problem decomposition; data collection, representation, and analysis; parallelization; control flow; incremental and iterative; testing and debugging; and questioning (Weese & Feldhausen, 2017). Each item on the instrument measures computational thinking self-efficacy on a five-value Likert scale (i.e., 1=strongly disagree, 2=somewhat disagree, 3=not sure, 4=somewhat agree, and 5=strongly agree). The items are divided into four categories of statements. The items in the first nine statements all begin with the phrase "When solving a problem I…". The next nine begin with "I can write a computer program which…". Then the next four all begin "When creating a computer program I…". The final item is the statement “I understand how computer programming can be used in my daily life”. All statements pertain to a computational thinking skill. The items and their relation to the research questions and computational thinking skills are summarized in Table 1.

Table 1: *Computational Thinking and Problem Solving Self-Efficacy Scale and Research Questions*

Research Question	Theory	IV	DV
Main research Question: To what extent does exposure to Latin education or lack of Latin education influence computer science students’ self-efficacy in computational thinking and problem solving?	Self-efficacy	Type of Latin language learning experience (up to 5 years of Latin language learning vs. 0 years of Latin Language Learning)	Mean score of subscales on the computational thinking and problem solving self-efficacy scale (Weese & Feldhausen, 2017)
Sub-questions: (each scale)			
To what extent does exposure to Latin education influence computer science students’ algorithmic thinking self-efficacy?	Self-efficacy	Type of Latin language learning experience (up to 5 years of Latin language learning vs. 0 years of Latin Language Learning)	Mean score of algorithmic thinking self-efficacy items on the computational thinking and problem solving self-efficacy scale (Weese & Feldhausen, 2017)
To what extent does exposure to Latin education influence computer science students’ abstractive thinking self-efficacy?	Self-efficacy	Type of Latin language learning experience (up to 5 years of Latin language learning vs. 0 years of Latin Language Learning)	Mean score of abstraction self-efficacy items on the computational thinking and problem solving self-efficacy scale (Weese & Feldhausen, 2017)

Table 1 (continued)

To what extent does exposure to Latin education influence computer science students' problem decomposition self-efficacy?	Self-efficacy	Type of Latin language learning experience (up to 5 years of Latin language learning vs. 0 years of Latin Language Learning)	Mean score of problem decomposition self-efficacy items on the computational thinking and problem solving self-efficacy scale (Weese & Feldhausen, 2017)
To what extent does exposure to Latin education influence computer science students' data collection, representation, and analysis self-efficacy?	Self-efficacy	Type of Latin language learning experience (up to 5 years of Latin language learning vs. 0 years of Latin Language Learning)	Mean score of data collection, representation, and analysis self-efficacy items on the computational thinking and problem solving self-efficacy scale (Weese & Feldhausen, 2017)
To what extent does exposure to Latin education influence computer science students' parallelization self-efficacy?	Self-efficacy	Type of Latin language learning experience (up to 5 years of Latin language learning vs. 0 years of Latin Language Learning)	Mean score of parallelization self-efficacy items on the computational thinking and problem solving self-efficacy scale (Weese & Feldhausen, 2017)
To what extent does exposure to Latin education influence computer science students' control flow self-efficacy?	Self-efficacy	Type of Latin language learning experience (up to 5 years of Latin language learning vs. 0 years of Latin Language Learning)	Mean score of control flow self-efficacy items on the computational thinking and problem solving self-efficacy scale (Weese & Feldhausen, 2017)

Table 1 (continued)

To what extent does exposure to Latin education influence computer science students' incremental and iterative self-efficacy?	Self-efficacy	Type of Latin language learning experience (up to 5 years of Latin language learning vs. 0 years of Latin Language Learning)	Mean score of incremental and iterative self-efficacy items on the computational thinking and problem solving self-efficacy scale (Weese & Feldhausen, 2017)
To what extent does exposure to Latin education influence computer science students' testing debugging self-efficacy?	Self-efficacy	Type of Latin language learning experience (up to 5 years of Latin language learning vs. 0 years of Latin Language Learning)	Mean score of testing and debugging self-efficacy items on the computational thinking and problem solving self-efficacy scale (Weese & Feldhausen, 2017)
To what extent does exposure to Latin education influence computer science students' questioning self-efficacy?	Self-efficacy	Type of Latin language learning experience (up to 5 years of Latin language learning vs. 0 years of Latin Language Learning)	Mean score of questioning self-efficacy items on the computational thinking and problem solving self-efficacy scale (Weese & Feldhausen, 2017)

The higher the score on a subscale (1-5), the more self-efficacy the subject feels he or she possesses about the statement's computational thinking skill – 1 being weak and 5 representing strong self-efficacy. The authors reported on the descriptive statistics of mean score and standard deviation.

Data Collection/Procedures

Data collection commenced upon receiving IRB approval, approval from the university with which I am affiliated, and approval from the Memphis school to conduct research. In the Spring of 2019, I visited each computer science class and the afterschool robotics club to clearly explain the research. I explained who I am, why I am conducting this research, and what I plan to do with the data. I showed them the instrument and told them it would take a few minutes to complete. Having explained the research, I gave them a letter to be given to their parents for informed consent as the participants will be minors. The letter was signed by the parents for consent and signed by the students for their assent. Included in the letter was a space for the students to provide their email address for me to send the instrument. The research allocated two weeks for the completion of this.

After the letters were received, I directed the students to check their email for a message from me. In this message I provided a link to a Google Form of the instrument. When they open the link, they will be free to complete the form. Prior to completing the actual instrument items, there was a section of the survey where students could identify their grade level and experience with Latin. Upon completion, I had access to all responses. The students also had access to their own responses and whatever results and conclusions that followed.

Students who were absent, did not receive the email sent by me, or did not feel comfortable had no obligation to participate. I sent reminder emails to students who had not responded to the initial email. The timeline of events appears in Table 2.

Table 2: *Experiment Conditions and Activities Timeline*

Week	Students with Latin	Students without Latin
1	Spoke to students about research and procedures	Spoke to students about research and procedures
	Gave students assent and consent forms	Gave students assent and consent forms
	Compiled an email list of student participants	Compiled an email list of student participants
	Sent reminder email to fill out and return forms	Sent reminder email to fill out and return forms
	Reminded in person to return forms	Reminded in person to return forms
	Collected forms	Collected forms
2	Collected forms	Collected forms
	Sent students link to scale	Sent students link to scale
	Returned to classes to restate research and point out link was sent	Returned to classes to restate research and point out link was sent
	Sent reminder email to students who have not filled out scale	Sent reminder email to students who have not filled out scale
	Reminded in person to fill out scale	Reminded in person to fill out scale

Data Analysis

To test the null hypothesis that having a Latin language learning yields no significant influence on computer science students' self-efficacy in computational thinking and problem solving, a multivariate analysis of variance (MANOVA) test was utilized for this causal-comparative study. A MANOVA test is an analysis of variance (ANOVA) test used when there are several correlated variables and the researcher can more effectively detect an effect (Field, 2009). Using MANOVA as opposed to ANOVA is necessary because conducting multiple ANOVAs increases the familywise error likelihood (Field, 2009). In addition, based on the fact that the overarching skill sought to measure is computational thinking, there is value in seeking conceptual multivariate gain from the instrument (Grice & Iwasaki, 2007). Separate post hoc ANOVA tests were also conducted. To test the null hypotheses that having a Latin language learning yields no significant influence on computer science students' abstraction; problem decomposition; data collection, representation, and analysis; parallelization; control flow; incremental and iterative; testing and debugging; and questioning self-efficacy, a separate ANOVA test was run for each computational thinking skill component.

Once the data was collected, I transferred all the information into a Microsoft Excel spreadsheet. Then, I transferred the data into IBM's Statistical Package for the Social Sciences (SPSS) software. This software yielded the statistics needed to properly analyze the data. As noted earlier, I controlled for both sex and grade level in this study.

ANOVA tests yield an *F* value that can be converted into a *p* value (Creswell, 2008). According to Gall et al. (2010), a *p* value of .05 is considered sufficient to reject a null hypothesis in educational research. Instead of *F*, MANOVA tests use one of four statistics: Pillai's trace, Hotelling's-Lawley trace, Wilk's lambda, or Roy's largest root. This study utilized the Pillai's trace statistic because it is the best choice when dealing with a small sample size (Glen, 2016). The effect size was measured using partial eta squared where a small effect size is .01, a medium is .06, and large is .138 (Rockinson-Szapkiw, 2013). The descriptive statistics of mean and standard deviation were included in the analysis. I also reported the reliability for my study.

Prior to conducting the analysis assumption testing was completed. For an ANOVA test, normality is assumed. This normality can be tested through histograms marked by a symmetrical bell curve. The Shapiro-Wilk normality test was run to analyze tenability in a normality assumption with a significance level above .05. The Shapiro-Wilk test works well with a low sample size. ANOVA tests also assume equality of variances, also called homogeneity of variances. Levene's Test for Equality of Variance and Bartlett's test at a significance level larger than .05 indicate equality of variance.

In addition to the assumptions of an ANOVA test, MANOVA tests assume multivariate normality examined by using Mahalanobis' distance value. This value is compared with the critical value on a chi-squared distribution table. This assumption is not tenable if the data's highest value exceeds the value on the chart. Normality can be affected by outliers detected by box plots. Assuming the data are correct, I dealt with outliers by changing the score (Field, 2009). Multicollinearity and singularity are also assumed in a MANOVA test. This can be checked by scanning a correlation matrix of all predictor variables to determine if a high correlation of .80 or .90 exists. SPSS also generates the variation inflation factor and the tolerance statistic for this assumption. For the assumption of linearity, a curvilinear line on a scatterplot matrix indicates the assumption is not tenable. Finally, Box's M tests the assumption of homogeneity of variance-covariance.

Limitations

The study was conducted thoroughly but has some limitations. One limitation is that the sample is primarily comprised of white males; approximately 75% of the student body from which the sample will be drawn is white. A future study may focus on the role race plays into computational thinking self-efficacy and involve a more diverse population. Sample size is also a limitation. Freshman computer science course already have limited numbers, spanning between 11 and 20 students, which might decrease with each subsequent grade. The number of Latin students will be drawn from an equally minimal sample. Additionally, the narrow selection criteria to choose test subjects may have threatened the study's internal validity. Only investigating computer science students might limit the sample to students of a higher intellect. The fact that students with higher computational thinking skills are drawn to Latin is a selection threat. Finally, this study is limited by the confound that other factors that influence computational thinking, i.e., world languages, mathematics, STEM, science, music, etc., were not accounted for.

One potential limitation of the instrument is the audience of its original design. The computational thinking scale was originally created to investigate 5^{th}-9^{th} grade students and has not been applied to sophomores, juniors or seniors. Another potential limitation is that the scale's authors admit they created the instrument without anyone else reviewing it or offering additional input.

CHAPTER FOUR: RESULTS

Introduction

A causal comparative research study was used to investigate whether Latin exposure affected computational thinking self-efficacy. Given the multiple dependent variables, the test called for a MANOVA (Dattalo, 2013) followed by ANOVA tests for each of the nine subscales from the instrument. A partial eta squared statistic will be used to evaluate the effect size on the scale of small effect (.01), moderate effect (.06), and large effect (.14) (Cohen, 1977).

Participants

A total of 75 students were asked to participate. These students came from the following computer science classes: Introduction to Computer Science (*n* = 13), Honors Computer Science (*n* = 15), two sections of Computer Science Principles (*n* = 27), and IT Support Fundamentals (*n* = 10). In addition, students from an after-school robotics club participated (*n* = 3). Finally, a few participants came from a Principles of Engineering (*n* = 7) course where coding is an essential component of the class.

With a volunteer rate of 71%, a total of 53 students participated in this study. Having heard about the research and study purpose, students were asked to participate in person. After all consent and assent forms were received, students were asked to participate via email. Representatives of each grade participated: freshmen (*n* = 6, 11.3%), sophomore (*n* = 15, 28.3%), junior (*n* = 16, 30.2%), and senior (*n* = 16, 30.2%). Students with no exposure to Latin at any level of education (*n* = 20) represented 37.7% of the sample, while students with at least one year of Latin (*n* = 33) represented 62.2% of the sample. From the total sample, 1.9% (*n* = 1) recorded taking Latin for 7 total years, 1.9% (*n* = 1) for 6 years, 9.4% (*n* = 5) for 5 years, 17% (*n*

= 9) for 4 years, 11.3% (*n* = 6) for 3 years, 11.3% (*n* = 6) for 2 years, and 9.4% (*n* = 5) for 1 year. Concerning the number of years enrolled in computer science, 39.6% (*n* = 21) responded 1 year, 41.5% (*n* = 22) responded 2 years, 13.2% (*n* = 7) responded 3 years, and 5.7% (*n* = 3) responded 4 years. Information on the total number of foreign language years taken is listed in Appendix D.

Instrumentation and Cronbach's Alpha

This study utilized the computational thinking and problem solving self-efficacy scale (Weese & Feldhausen, 2017) to measure the main research question and the nine subscale questions. Each item measures a certain computation thinking skill. Many of the computational thinking skills require multiple questions to cover their effect. The instrument is a 23-item Likert-type scale and consists of the following constructs: algorithmic thinking (questions 1, 2, 10, and 11), abstraction (questions 3, 4, and 18), problem decomposition (questions 5 and 22), parallelization (questions 6, 15, 16), data collection, representation, and analysis (questions 7 and 17), control flow (questions 8, 9, 12, and 13), being incremental and iterative (question 19), testing and debugging (question 20), and questioning (question 23). Question 21 was not included in this study.

Each item on the instrument measures computational thinking self-efficacy on a five-value Likert scale (i.e., 1 = strongly disagree, 2 = somewhat disagree, 3 = not sure, 4 = somewhat agree, and 5 = strongly agree). The higher the score, the higher the self-efficacy reflected based on the particular statement. Example questions include, "When solving a problem I look how information can be collected, stored, and analyzed to help solve the problem," and "I can write a computer program which responds to an event like pressing a key on a keyboard". Cronbach's alpha for the subscales did not surpass .47. The entire scale, however, was deemed reliable with a Cronbach's alpha of .82.

Results

This study sought to find whether taking any level of Latin has an effect on computational thinking and problem solving self-efficacy. Additionally, this study sought to investigate to what extent several components of self-efficacy are affected by taking Latin. There was one main research question for this study, along with nine sub-questions. Specifically, the research questions were as follows:

Research Question 1: To what extent does exposure to Latin education influence computer science students' self-efficacy in computational thinking and problem solving?

a) To what extent does exposure to Latin education influence computer science students' *algorithmic thinking* self-efficacy?
b) To what extent does exposure to Latin education influence computer science students' *abstractive thinking* self-efficacy?
c) To what extent does exposure to Latin education influence computer science students' *problem decomposition* self-efficacy?
d) To what extent does exposure to Latin education influence computer science students' *data collection, representation, and analysis* self-efficacy?
e) To what extent does exposure to Latin education influence computer science students' *parallelization* self-efficacy?
f) To what extent does exposure to Latin education influence computer science students' *control flow* self-efficacy?

g) To what extent does exposure to Latin education influence computer science students' *incremental and iterative* self-efficacy?

h) To what extent does exposure to Latin education influence computer science students' *testing and debugging* self-efficacy?

i) To what extent does exposure to Latin education influence computer science students' *questioning* self-efficacy?

Table 3 provides the descriptive statistics for each of the 22 items. The two groups are broken down into those with no classroom exposure to Latin and those with at least one year of Latin. Table 4 shows the descriptive statistics for the responses of the 53 participants broken down by the mean of each subscale.

Table 3

Descriptive Statistics (*N* = 53) *for Individual Items*

	Non- Latin		*Latin*	
Variable	(*n* = 20)		(*n* = 33)	
	M	*SD*	*M*	*SD*
Algorithms Q1 When solving a problem I create a list of steps to solve it	4.35	.56	3.73	1.13
Algorithms Q2 When solving a problem I use math	4.10	1.17	3.82	1.04
Algorithms Q3	4.10	1.07	4.24	1.00

Table 3 (continued)

I can write a computer program which runs a step-by-step sequence of commands				
Algorithms Q4 I can write a computer program which does math operations like addition and subtraction	4.55	.61	4.45	1.00
Abstraction Q1 When solving a problem I try to simplify the problem by ignoring details that are not needed	4.25	1.12	4.18	1.13
Abstraction Q2 When solving a problem I look for patterns in the problem	4.85	.37	4.67	.60
Abstraction Q3 I can write a computer program which uses custom blocks	3.55	1.28	3.21	1.14
Problem Decomposition Q1 When solving a problem I break the problem into smaller parts	4.60	.82	4.55	.71
Problem Decomposition Q2 When creating a computer program I break my program into multiple parts to carry out different actions	4.30	.73	4.27	.80
Parallelization Q1 When solving a problem I work with others to solve different parts of the problem	4.00	1.26	3.82	1.19
Parallelization Q2 I can write a computer program which does more than one thing at the same time	4.40	.75	4.00	1.15
Parallelization Q3 I can write a computer program which uses messages to talk with different parts of the program	3.45	1.32	3.45	1.23

Table 3 (continued)

Data Q1 When solving a problem I look how information can be collected, stored, and analyzed to help solve the problem	4.15	.99	4.24	.75
Data Q2 I can write a computer program which can store, update, and retrieve values	3.95	1.36	4.06	1.06
Control Flow Q1 When solving a problem I create a solution where steps can be repeated	4.05	1.05	4.39	.75
Control Flow Q2 When solving a problem I create a solution where some steps are done only in certain situations	4.05	.95	3.91	1.04
Control Flow Q3 I can write a computer program which uses loops to repeat commands	4.20	.83	4.33	.96
Control Flow Q4 I can write a computer program which responds to events like pressing a key on the keyboard	4.00	.97	4.03	1.08
Control Flow Q5 I can write a computer program which only runs commands when a specific condition is met	4.55	.76	4.42	.71
Incremental and Iterative Q1 When creating a computer program I make improvements one step at a time and work new ideas in as I have them	4.60	.60	4.33	.78
Testing and Debugging Q1 When creating a computer program I run my program frequently to make sure it does what I want and fix any problems I find	4.75	.55	4.58	.61
Questioning Q1	4.70	.47	4.70	.47

Table 3 (continued)

I understand how computer programming can be used in my daily life

Table 4

Descriptive Statistics (*N* = 53): *Scale Means*

Variable	*Non- Latin* (*n* = 20)		*Latin* (*n* = 33)	
	M	*SD*	*M*	*SD*
Algorithms Mean	4.28	.44	4.06	.70
Abstraction Mean	4.22	.61	4.02	.61
Parallelization Mean	3.95	.73	3.76	.82
Problem Decomposition Mean	4.45	.48	4.41	.55
Data Mean	4.05	.90	4.15	.69
Control Flow Mean	4.17	.56	4.22	.50
Incremental and Iterative	4.60	.60	4.33	.78
Testing and Debugging	4.75	.55	4.58	.61
Questioning	4.70	.47	4.70	.47

According to the descriptive statistics above, the non-Latin group scored higher on all the individual questions except Algorithms Q3, Data Q1 and Q2, and Control Flow Q1, Q3, and Q4.

The highest mean score for a question in both the non-Latin group (4.85) and the Latin group (4.67) was Algorithms Q2: “When solving a problem I look for patterns in the problem”. The lowest mean for a question in the non-Latin group (3.45) was Parallelization Q3: “I can write a computer program which uses messages to talk with different parts of the program”. For the Latin group (3.21), the lowest mean was found in Abstraction Q3: “I can write a computer program which uses custom blocks”. The highest mean for a subscale in the non-Latin group (4.75) was found in Testing and Debugging. The highest mean subscale for the Latin group (4.61) was found in Questioning. In the Parallelization subscale, both the non-Latin group (3.95) and the Latin group (3.76) produced the lowest means.

Assumption Testing

Prior to conducting the MANOVA tests, assumption testing was conducted. MANOVA tests assume multivariate normality examined by using Mahalanobis’ distance value (Filzmoser, 2004). One outlier was found using this statistical measure. For nine dependent variables, the maximum critical value is 27.88. One respondent recorded a critical value 33.56. This score was changed via mean substitution, a commonly used practice for handling an outlier (Somasundaram & Nedunchezhian, 2012). Multicollinearity and singularity are also assumed in a MANOVA test which was checked by scanning a correlation matrix of all predictor variables to determine a high correlation of .80 or .90 (Rockinson-Szapkiw, 2013). A Pearson correlation test was run to measure the correlation between the variable of Latin exposure and each of the nine computational thinking skills. None of the correlations were found to be significant at the 0.01 level. The correlations and Pearson values are displayed in Table 5.

Table 5

Correlation Matrix

		Algorithms	Abstraction	Problem Decomp.	Parallelizat.	Data
Latin Exposure	Pearson	-0.169	-0.156	-0.038	-0.12	0.065
	Sig. (2-tailed)	0.225	0.263	0.785	0.393	0.646

		Control Flow	Incremen./ Iterative	Testing and Debugging	Question.
Latin Exposure	Pearson	0.046	-0.181	-0.144	-0.067
	Sig. (2-tailed)	0.746	0.195	0.303	0.632

For the assumption of linearity, a curvilinear line on a scatterplot matrix indicates the assumption is not tenable (Rockinson-Szapkiw, 2013). The matrix scatterplots show a weak, negative, linear relationships amongst the variables. To test normality, this study ran a Shapiro-Wilk test (Rockinson-Szapkiw, 2013) at a significance of .05. The significance for most of the subscales ranged from .000-.026. For Control Flow, the significance was .07. Finally, Box's M test was run to test homogeneity of variance-covariance (Field, 2009). This test resulted in a significance of .01, above the threshold of .001.

MANOVA/ANOVA

A MANOVA test was conducted to determine the effect of Latin course exposure on computational thinking self-efficacy. Due to the violations of assumptions and the low sample size, Pillai's Trace will be the statistic of interest because it is more robust with a low sample (Scheiner, 2001). The test found Pillai's Trace = .17, *F* = .95, p = .492, and an observed power = .40. The effect size was large, partial *η2* = .17, under the convention of .138 being a large effect. Despite not finding significance in the MANOVA test, separate ANOVA tests were conducted for each subscale scale to analyze and interpret each dependent variable separately (Grice & Iwasaki, 2007). Table 6 summarizes the results of the tests.

Table 6

Separate ANOVA Results

Variable	*Results*
Algorithms Mean	*F* = 1.51, p = .225, partial *η2* = .03
Abstraction Mean	*F* = 1.28, p = .263, partial *η2* = .02
Problem Decomposition Mean	*F* = .08, p = .393, partial *η2* = .00
Parallelization Mean	*F* = .74, p = .785, partial *η2* = .01
Data Mean	*F* = .21, p = .646, partial *η2* = .00
Control Flow Mean	*F* = .106, p = .746, partial *η2* = .00
Incremental and Iterative	*F* = 1.73, p = .195, partial *η2* = .03
Testing and Debugging	*F* = 1.08, p = .303, partial *η2* = .02
Questioning	*F* = .23, p = .632, partial *η2* = .00

Summary

Reliability of the entire instrument was confirmed using a reliability test yielding a Cronbach's alpha of .82. The data, however, did not meet of the necessary assumptions for a MANOVA test. The sample size for the Latin group was of no concern at *n* = 33, yet the paucity of responses for the non-Latin exposure group caused problems (*n* = 20). The means from the descriptive statistics show that the non-Latin group outscored the Latin group in most of the computational thinking skills. Pillai's trace statistic from the MANOVA test shows no statistical significance in the computational thinking and problem solving scale. The individual results from the ANOVA tests showed no statistical significance for any of the nine subscales.

CHAPTER FIVE: DISCUSSION AND CONCLUSIONS

Discussion

Due to the recent increase of STEM-related jobs (Breiner, Harkness, Johnson, & Koehler, 2012), educational institutions have prioritized offering science, technology, engineering, and math (STEM) courses (Kennedy & Odell, 2014). STEM courses and programs are being introduced earlier in education programs – such as in the middle school level (Repenning, Webb, & Ioannidou, 2010) or even as early as the elementary school level (Bers, Flannery, Kazakoff, & Sullivan, 2014). And as the range of STEM course availability increases, so has the precise understanding of the skills needed to succeed in those courses. Many of these skills are present in computational thinking, fundamental skills for not only computer scientists but all STEM professionals (Wing, 2006). Computational thinking encompasses the analytical skills utilized in all STEM courses (Wing, 2008). Weese and Feldhausen (2017) argue that computational thinking consists of the following constructs: algorithmic thinking; abstraction; problem decomposition; data collection, representation, and analysis; parallelization; control flow; incremental and iterative; testing and debugging; and questioning.

Research shows STEM learning outcomes are significantly influenced by affective components such as self-efficacy. Self-efficacy is important for learning in a variety of domains and has a positive effect on student performance (Vekiri & Chronaki, 2008). A student’s self-efficacy is a predictor of their success at computational thinking (Ramalingham, LaBelle, & Wiedenbeck, 2004). The iterative problem-solving nature of computational thinking makes understanding self-efficacy vital. Studying a learner’s perception of his or her abilities will offer insight on why frustration may set in and result in the rejection of a concept (Tsai, Wang, & Hsu, 2018). The intervention of STEM activities such as creative thinking projects can enhance self-efficacy by teaching computational skills (Shell et al., 2014).

Considering the lack of ideal growth in STEM learning outcomes, Miller et al. (2013) argue that interdisciplinary approaches are needed to better improve competencies such as computational thinking. In fact, computational thinking has increasingly been implemented in STEM and non-STEM-related fields. To date, psychologists, sociologists, and anthropologists have all applied computational thinking to the field of cognitive science (Settle & Perkovic, 2010). Various studies show that interdisciplinary approaches such as writing (Wolz, Stone, Pulimood, & Pearson, 2010) and humanities coursework (Dierbach et al., 2011) play an important role in how students improve computational thinking in STEM. Interdisciplinary disciplines allow learners to apply creative thinking, infuse technology into the curriculum, and complete computer-assisted tasks. Language learning may also produce these benefits given its similar emphasis on problem solving, critical thinking, and analysis (Cooper et al., 2008). Classical Latin language learning may uniquely benefit students with skills related to computational thinking (Olabe, Olabe, Basogain, & Castaño, 2011) because its makeup is analogous to that of the language of mathematics. Therefore, research indicates a high probability that success in Latin courses could produce the self-efficacy necessary to master computational thinking and then transfer that benefit to other courses.

To explore this possibility this study analyzed the interdisciplinary relationship between Latin language learning and computational thinking self-efficacy. Students' exposure to Latin was correlated with proficiency in the nine computational skills of algorithmic thinking; abstraction; problem decomposition; data collection, representation, and analysis; parallelization;

control flow; being incremental and iterative; testing and debugging; and questioning (Weese & Feldhausen, 2017). The data resulting from this study does not show Latin courses have an overall effect on computational thinking self-efficacy. However, the descriptive statistics hold potential for further research opportunities and showed some instances where the Latin group yielded a higher self-efficacy level. Algorithms Q3, Data Q1, Data Q2, Control Flow Q1, Control Flow Q3, and Control Flow Q4 produced a higher mean score for the Latin group and could serve as catalysts for future study. Below I will summarize and interpret the data previously presented in Chapter 4, describe the potential limitations of the study, discuss the theoretical and practical implications of the study, and, finally, present the conclusions that can be drawn from the study.

Data Collection Procedures

Data collection took place at a Memphis boys' high school. A total of 53 boys enrolled in computer science classes and a robotics club voluntarily participated in this study. Prior to collecting data, I received forms expressing consent and assent. Data collection was acquired via a Google Form link sent to the students upon collecting the forms.

Findings

Research question 1 findings

For the main research question (RQ 1), the findings failed to yield that taking Latin increases computational thinking and problem solving self-efficacy; that is, the hypothesis that Latin exposure influences computational thinking self-efficacy was not supported. These results fail to align with results of Settle et al. (2012), Cooper et al. (2008), and Chater and Vitanyi (2003) which found interdisciplinary links between computational thinking and language learning. Further analysis of the methodology may yield insight into the results and differences across these studies. Settle et al. (2012) found that computational thinking can easily be practiced in Latin language learning through grammar notation, sentence diagramming, and metaphrasing. However, it was different from this study in that its subjects were middle and high school students whereas this study only surveyed high school students. Similarly, Cooper et al. (2008) found that language learning enhances computational thinking because of its link to improving critical thinking skills. However, the authors looked at multiple foreign languages instead of singling out Latin. In addition, Cooper et al. (2008) investigated foreign language students versus students without foreign language in the context of the SAT test; this study made no comparison of groups in relation to a test. Finally, Chater and Vitanyi (2003) argued that language acquisition entails finding patterns in linguistic input with the mathematical and computational theories guiding it. Collectively, the research of Settle et al. (2012), Cooper et al. (2008), and Chater and Vitanyi (2003) suggest interdisciplinary benefits of foreign languages in general. The authors did not test any subjects, but instead analyzed language learning with these theories, while this study especially focused on the benefits and potential influence of Latin learning.

There has been little research on a potential influence of Latin on computational thinking self-efficacy. Due to the novel way this study approaches STEM it may be appropriate to explore the descriptive statistics since no significant differences were found in the MANOVA and ANOVA tests on this construct. The descriptive statistics, despite no significance, showed that the non-Latin exposure group often scored a higher mean and that both groups often averaged a score higher than “4” or *somewhat agree*.

Research question 1a findings

The algorithmic thinking skill (RQ 1a) is defined as the ability to analyze a problem precisely and subsequently construct a step-by-step process to resolve the presented problem (Futschek, 2006). Wing (2006) argued that algorithmic thinking is applicable and ubiquitous when one encounters real-world situations. Despite a lack of statistical significance, this study shows both groups scored highly in self-efficacy regarding algorithmic thinking; both Latin and non-Latin groups produced a mean score above 4. However, the non-Latin group yielded a mean slightly higher. Of the four items pertaining to algorithmic thinking, Q3 (“I can write a computer program which runs a step-by-step-sequence of commands”) was one item where the Latin group did have a higher mean. The gap between the mean of the non-Latin group and the Latin group for algorithmic thinking Q1 was the single highest out of all the items (4.35 – non-Latin, 3.713 – Latin). Once again, these results are interesting in view of prior studies which found differential results. For example, Voskoglou and Buckley (2012) connected algorithmic thinking to computational thinking as a means to solve real-world problems, indicating that analyzing behaviors and reactions can be done within an algorithmic framework. The authors, however, ran their study on students from Graduate Technological Educational Institute (TEI) in Greece while this study used high school students as its subjects. Considering the pervasiveness of algorithmic thinking in everyday life argued in previous studies (Voskoglou and Buckley, 2012; Wing, 2006), it is difficult to distinguish whether it is the students’ Latin language exposure or the inevitable daily encounters with step-by-step processes contributing to self-efficacy in these particular skills. Given the pervasiveness of this activity, this construct may thus be the easiest to achieve in terms of mastery experience and self-efficacy. The scores may thus be a result of the ceiling effect which can affect response distribution (Moret et al., 2007). In addition, Voskoglou and Buckley (2012) and Wing (2006) centered their research on cognitive abilities, whereas this study focused on the transfer of self-efficacy. For self-efficacy to increase, there must be a source such as mastery experiences, vicarious experiences, verbal persuasion, or emotional and physiological states through which it transfers.

Research question 1b findings

Abstraction (RQ 1b) is defined by stripping down a problem and capturing common characteristics of one set to be used to represent all other instances (Lee et al., 2011). This study found no statistical significance with this construct. Further, the descriptive statistics show that the Latin group did not outperform the non-Latin group in any of the three questions pertaining to abstraction. For abstraction Q3, both groups produced mean scores just under 4 (3.55 – non-Latin, 3.21 – Latin) and the lowest means of the 22 items. A score of 3 was the coded for the response *not sure*. Given these results, it would seem this study fails to align with previous research on abstraction. Settle et al. (2012) asserted that Latin involves abstraction when a translator exercises grammar notation. These authors, however, were writing with middle school students in mind as they were included in the sample. Alternatively, this study only drew from high school students for the sample. In a similar study, Webb and Rosson (2013) described how an outreach program for middle-schoolers increased self-efficacy in computational thinking skills through computer activities that fostered abstraction using scaffolded examples. The authors described an outreach problem given to middle school girls. Once again, the sample for this study only included high school boys. While Settle et al. (2012) and Webb and Rosson (2013) are comparable in some ways to this study, a middle school sample versus a high school sample is a worthy distinction to make. Research suggest that the transition from middle school to high school is drastic because high school faculty is more diverse, students' grades tend to drop, the courses are more grade-oriented, and the coursework is more competitive (Mizelle & Irvin, 2000). Also, in the case of Webb and Rosson (2013), the distinction between boys' and girls' literacy rates, leisure activities, and other significant variables (Sanford, 2005) further make it difficult to compare such a study to this current study. An alternative interpretation is that while abstraction might frequent one domain of Latin learning such as grammar notation it may not play a large enough role to increase self-efficacy in computational thinking. There are additional parts of Latin learning such as understanding syntax, vocabulary acquisition, and translating that may not be relevant to the abstraction aspect of computational thinking.

Research question 1c findings

Parallelization (RQ 1c), the simultaneous processing or execution of a task (Weese & Feldhausen, 2017), is a skill that can be utilized in spaces beyond the classroom. While a student will exercise this skill in Latin courses, this study did not find Latin to be a significant influence on parallelization. In fact, descriptive statistics showed the non-Latin group surpassed the Latin group in the mean for parallelization as evidenced by the mean for both groups was under 4 (3.95 – non-Latin, 3.76 – Latin). Parallelization Q3 also yielded low means relative to other items with both groups averaging 3.45. Barr and Stephenson (2011) argued from the perspective of cognitive outcomes that parallelization appears uniquely in math, science, and computational science courses. Once again, this study is different in that the results were guided by the theory of self-efficacy and the affective domain whereas the other focused on cognitive outcomes. If parallelization is not present in courses outside of math, science, and computer science then it would be difficult for the source of mastery experiences (Bandura, 1997) to be applicable. Through mastery experiences, individuals gain self-efficacy in a particular area or task based on prior mastery in that same setting. While translating Latin does entail processes similar to parallelization, e.g., simultaneous parsing and translating, the self-efficacy that comes from parallelization may be uniquely suited for math, science, and computer science. In language learning, parallelization requires the learner to simultaneously know the morphology of each word for agreement and syntactical purposes. Although parallelization is present in computational thinking, the tasks simultaneously being executed are full processes as opposed to the individual components of a process performed through language learning. The tasks in parallelization may also consist of full calculations potentially more complex than word forms and sentence structure. This may explain the lack of statistically significant findings for this construct.

Research question 1d findings

Problem decomposition (RQ 1d) or breaking down problems into smaller parts that can be more easily solved (Barr & Stephenson, 2011), is a multi-layered operation where an individual must construct a hierarchy of procedures in an attempt to facilitate a task (Jackson & Jackson, 1996). Wing (2006) submitted that problem decomposition is used when attacking complex tasks or designing large complex systems. This study found no statistical significance for this construct and the means for problem decomposition were close for both groups (4.45 – non-Latin, 4.41 – Latin). Once again, the literature and prior studies may help explain why there were no results found in this study. For example, Yasar (2018) argued that people are neither aware when they practice problem decomposition, nor do they fully utilize this skill. The author added that, because problem decomposition is used so heavily in programming, finding outlets to improve this skill is a concern for educators. While Yasar (2018) described the various instances where an individual might encounter problem decomposition, they failed to mention demographic information. That is, not much is known about the makeup of the participants. This study, however, specifically targeted high school boys. The focus on boys is a crucial point because, of the many differences between boys and girls, boys tend to be more reluctant to read, less attentive, and inferior verbally (Hunsader, 2002). Further, problem decomposition may be too complex for Latin students at multiple levels to grasp. Students may thus not be aware of how to practice problem decomposition even if necessitated by a task.

Research question 1e findings

For the research questions dealing with data representation, collection, and analysis (RQ 1e), the study yielded no significance. The mean for both groups were above a 4 average (4.05 – non-Latin, 4.15 – Latin). Notably, both scores of the two items were higher for the Latin group albeit with no statistically significant difference. The two items read, "When solving a problem I look how information can be collected, stored, and analyzed to help solve the problem" for Q1 and "I can write a computer program which can store, update, and retrieve values" for Q2. For the interpretation of these results, Forehand (2010) said analysis is applicable in a variety of fields for a broad range of learners. This study, however, investigated specifically the self-efficacy of high school boys. In a similar vein, Barr and Stephenson (2011) argued the presence of data representation, collection, and analysis in language arts. These authors, however, discussed data representation, collection, and analysis in a K-12 setting and not specifically with high school boys as in this study. The specificity of high school boys is an important point. Not only do boys differ from girls in terms of socialization in learning (Hunsader, 2002; Sanford, 2005), but there is a stark difference in ability and efficacy between elementary school aged children and high school aged young adults (Pajares, Johnson, & Usher, 2007).

Research question 1f findings

Defined as the order in which steps are executed and directing an algorithm's steps (Weese & Feldhausen, 2017), control flow (RQ 1f), produced descriptive statistics where the Latin group scored higher in three of the five items. These items include the following questions: "When solving a problem I create a solution where steps can be repeated" for Q1, "I can write a computer program which uses loops to repeat commands" for Q3, and "I can write a computer program which responds to events like pressing a key on the keyboard" for Q4. Despite the higher means, no statistical significance was found for this construct. However, given the limited research on language learning and STEM self-efficacy, future discussion may be appropriate for these results. Control flow consists of controlling the progression of steps for a program through programming methods according to Bers et al. (2014). Bers et al. (2014) infused a computational thinking and robotics program into a curriculum tailor-made for early childhood students instead of the high school students analyzed by this study. Bers et al. (2014) focused on the strength of the curriculum implemented instead of this study's focus on self-efficacy. An alternative explanation for the lack of significance for this construct might be that in language learning, control flow requires the learner to develop a flow of steps to be used for translation and is especially germane to a particular age group. However, the control flow exercised in language learning is devoid of the programming present in the control flow exercised in computational thinking.

Research question 1g findings

Concerning the research question on being incremental and iterative (RQ 1g), this construct entails forming constraints with each attempt to problem-solve until a solution is reached (Jonassen, 2008) and to direct programmers to build a program step by step as opposed to all at once (Weese & Feldhausen, 2017). This subscale with only one item was not found to be statistically significant. Once again, prior studies may help elucidate the reasons for this result. Larman and Basili (2003) found that iterative and incremental thought processes present in software development has accelerated via the promotions of it in books and papers. The authors, however, did not indicate a particular age or gender in their explanation of incremental and iterative processes. Alternatively, this study looked at incremental and iterative processes in high school boys through the lens of self-efficacy, which factor in a person's characteristics and background (Hatlevik et al., 2018). Viewing incremental and iterative processes in boys versus girls requires an important distinction since research has suggested that girls outperform boys in some measures of problem-solving (Zhu, 2007). Another possible reason for the lack of significance found for this construct is that maybe language learning applies being incremental and iterative when the learner produces a logical translation, learning from mistakes with each attempt. Although a similar process is present in computational thinking, it could be different in that there is a programming element not present in language learning.

Research question 1h findings

Testing and debugging (RQ 1h), also with one item on the instrument, is defined as analyzing and fixing problems immediately during development (Weese & Feldhausen, 2017). There are multiple ways to interpret these results. Bers et al. (2014) explained that testing and debugging entails identifying a problem, developing a hypothesis on the cause of the problem, then developing a solution. These authors investigated early childhood students as opposed to self-efficacy with high school boys as in this study. In a similar manner, Webb and Rosson (2013) built off Bers et al. (2014) finding that debugging self-efficacy could be fostered by computer activities featuring scaffolded examples and minimalist workbooks. Once again, the population was different in that these authors only focused on high school girls. This may merit future research because girls generally perform better academically in school and record higher test scores (Sanford, 2005).

For further interpretation of the results, another possible reason why this construct was not significant and why the non-Latin group outperformed the Latin group in the descriptive statistics could be that testing and debugging would occur at the highest level of Latin learning. In doing so, they failed to attain the mastery experiences the theory suggests is important for STEM self-efficacy. Alternatively, it is more likely that students in the third, fourth, or fifth years of Latin could analyze an entire sentence of authentic classical Latin, identify unique and unusual characteristics of the syntax, diction, and morphology, and then develop a working translation. Along those lines, Tsai et al. (2018) argued that increasing experience in computational thinking activities may greatly enhance students' self-perceptions of debugging. It is possible that the sample did not possess enough advanced years in these skills which is why transfer did not occur.

Research question 1i findings

Finally, questioning (RQ 1i), defined as using each part of a code as opposed to using code not easily understood (Weese & Feldhausen, 2017), is a skill more similar one applicable in computer science than in Latin courses. This subscale, also showing no statistical significance, was represented by one item: “I understand how computer programming can be used in my daily life”. In the constructs being incremental and iterative, testing and debugging, and questioning, the non-Latin group scored higher means. All three means for the three groups were higher than 4. Mason (2015) described the complexity of questioning by asserting that it can invite a wide range of very different responses- answers, fact, data, etc. This author explored questioning in a digital environment, but never established a demographic on which he would focus, unlike this study which targeted self-efficacy in high school boys. The focus solely on boys is noteworthy since research suggests that there is a difference between boys and girls regarding the frequency with which each asks questions (Crowley, Callanan, Tenenbaum, & Allen, 2001). Further, in computational thinking, questioning might answer the question *why;* that is, as in *why* a certain result took place. This construct may be less germane to language learning, which tends to focus on reflection, interpretation, and carrying out of an inquiry (Wells, 1995).

Theoretical Implications

Self-efficacy is a factor in an individual’s ability to attain the learning outcomes through its effect on retention, performance, and skill development (Scida and Jones, 2017). Furthermore, low self-efficacy can predict student belief, performance (Vekiri & Chronaki, 2008), and academic achievement (Bong & Skaalvik, 2003). Due to the ever-increasing importance and ubiquity of computational thinking skills in STEM, possessing a high computational thinking self- efficacy is advantageous for students of multiple grade levels. Wing (2006) argued the importance of implementing computational thinking skills across multiple disciplines. Recently, trends show that exercising computational thinking has increasingly been visible and encouraged across multiple levels of education including grade school (Tran, 2018), middle school (Wolz, Stone, Pearson, Pulimood, & Switzer, 2011), high school (Ahamed, Brylow, Ge, Madiraju, Merrill, Struble, & Early, 2010), and university settings (Hambrusch, Hoffmann, Korb, Haugan, & Hosking, 2009). For these reasons, Tsai, Wang, and Hsu (2018) found that a high self-efficacy can influence computational thinking competencies.

Because languages in general have a probabilistic component to them in phonetic variation, corpus counts of different syntactic structures, and sociolinguistic variation (Chater & Manning, 2006), a language course may relate to computational thinking self-efficacy. One might hypothesize that using computational thinking skills to break down a Latin sentence uniquely allows a Latin translator to identify the grammatical elements of a sentence while relating it to other parts of the sentence (Settle et al., 2012). For example, metaphrasing in Latin translation, comparable to parallelization and control flow, relates the morphology of a Latin word to English word order while keeping its form to predict the function of other words in the sentence (Knudsvig, Seligson, & Craig, 1986). Considering the results of this study, there is not enough evidence to determine that Latin exposure has an impact on computational thinking self-efficacy. According to the theory, for an increase in self-efficacy to take place, several factors must be in play. For example, self-efficacy can only occur or improve through the sources of mastery experiences, vicarious experiences, verbal persuasion, and emotional and physiological states (Bandura, 1997). In the case of mastery experiences, it is not enough to simply take Latin. One must display some degree of discipline-specific mastery for self-efficacy to occur or improve. That is, the mastery may not transfer across computational thinking and Latin language learning.

Practical Implications

The results of this study have practical applications for teachers of Latin and STEM. Computational thinking skills form the basis for the curricula that support it, both STEM and non-STEM (Grover & Pea, 2013). Wing (2006) argues that computational thinking skills have real-world applications through its abstract and algorithmic approach to solving problems. Although results from this student were not statistically significant, students can still benefit their computational thinking skills from Latin learning (Settle et al., 2012; Chater and Vitanyi (2003). In Latin classes, these activities might be exercised in color-coding, sentence-diagramming, or anything where a step-by-step process is involved.

Additionally, the results show that the means of eight of the nine subscales was at least a 4 for both groups, with means ranging from 4.02-4.61. While results did not show this self-efficacy to be necessarily higher than the non-Latin group, a measure of self-efficacy was present for the various constructs based on the descriptive statistics. Given the early stages of this type of research, one might hypothesize that regardless of Latin influence or not, computer science students at this Memphis high school demonstrated relatively strong self-efficacy in computational thinking skills based solely on the descriptive statistics.

Several studies on computational thinking have utilized a methodology investigating cognitive outcomes. However, it is useful to approach a novel study such as this from multiple perspectives. Because methodologies involving cognitive outcomes and self-efficacy provide valuable results, explorations in these areas will benefit future research.

Since mastery experiences is a source of self-efficacy, it is practical for Latin teachers to understand how their students can reach this level. For students to master a subject, they must understand what they know, what they do not know, in what areas they struggle, what methods help them learn, etc. Equally as important is to know whether students are aware exactly what skills they are using in their courses. If students understand the computational thinking skills associated in Latin translating, this may aid in the transfer of self-efficacy. As a Latin teacher, we can be more intentional in making students aware of the computational thinking skills they are exercising at the various levels of Latin language learning.

Limitations

The greatest limitation for this study was the sample size. For educational research, a recommended sample size is 30 for both conditions (Maas & Hox, 2005). While the Latin condition had 33 participants, the non-Latin condition had only 20. The limitation of the sample size thus does not allow a more definitive finding. With an increased sample size, the study could have allowed more diversity in the independent variables, i.e., one year of Latin, two years of Latin, etc.

A second limitation is the pool of participants. All the participants came from the same school and took the same classes from the same teacher. In addition, most of them were white males. More diversity in the demographics of the participants, along with diversity in their educational background might have elicited different results. For example, it is possible that with a more varied background, individuals could have a larger array of interdisciplinary classes, which Kennedy and Odell (2014) argue is important for computational thinking. Also, including girls in the sample adds a new dimension to the study. The researcher would need to consider that while girls tend to possess lower self-efficacy in STEM courses (Rittmayer & Beier, 2008), this may not necessarily relate to cognitive outcomes in STEM.

A third limitation of this study is that it only looked at Latin versus non-Latin; that is, there was no factoring in years of Latin, which might be important for gaining mastery experiences. The reason for this is that the sample would have been too low to obtain the requisite number of participants. For example, there were students who have more than five, six, and seven years of Latin included in this study. A student in their second semester of Latin ever was in the same group as a student who had taken Latin for up to six or seven years. To get a sample of 30 per group, i.e., 30 for each year of Latin taken, it would require this study to include multiple schools. Considering the academic benefits of language learning (Cooper et al., 2008), excluding other languages such as French and Spanish also limited this study. In addition, self-efficacy is closely tied with the source mastery experiences (Bandura, 1997). Students with only one or two years of Latin may not be presented with an opportunity to receive this source of self-efficacy in such a short amount of time.

A fourth limitation is that the study only looked at self-efficacy using a Likert-style survey. This study did not utilize rating scale questions, rank order questions, or any other type of survey questions. This study was also limited to quantitative results. Opening the study up to qualitative research would allow other means of gauging self-efficacy such as interviews. These interviews would provide more information on self-efficacy including potential strengths and challenges. This qualitative data would also help the researcher better understand the cognitive processes of the participants.

Future Research

Given that this study looked at only Latin versus non-Latin exposure, future research would utilize this study, but with years of Latin as separate independent variables (i.e., 0, 1, 2, 3, etc). With Latin as a continuous variable, another study could run a logistic regression study to see if computational thinking self-efficacy increases as years increase. This would be worth investigating considering Settle et al. (2012) has already established the presence of computational thinking in studying the Latin language. This type of study would explore the degree to which increased exposure to Latin might impact computational thinking learning outcomes, which self-efficacy argues is important for mastery.

Another potential study could look at actual STEM scores, i.e., grades or standardized test scores, and how they may or may not be influenced by Latin exposure. The current study only investigated the influence of Latin on self-efficacy. Looking at STEM scores would be beneficial because studies have shown that students can perform at a high level despite having low self-efficacy (Mills, Pajares, & Herron, 2006; Lee, & Witta, 2001; Hodges, 2005). In a similar vein, a potential study could also explore a potential correlation between cognitive outcomes and affective outcomes, i.e., self-efficacy.

An additional study would open this research to multiple high schools other than this all-boys school with a fairly affluent student body. Inevitably, this would open the door to acquiring more and more diverse participants. Ideally, these new participants would also include high school girls. Adding girls to the student would also contribute to the conversation of how the two genders approach STEM differently, partly because many argue each gender is socialized differently (Mann, 1994; Gurian, 2010). In addition, students from different high schools would have learned computer science and Latin from different teachers and will have acquired varying sets of skills.

Future research might also choose an instrument other than the one used in this study to find greater diversity in the results. Computational thinking is a generally new idea, so instruments are beginning to emerge. It is possible that other instruments would include different aspects of computational thinking, which might provide further insights into the learning strategies that engender STEM learning outcomes. Further, if researchers continue to investigate self-efficacy in the area of Latin learning and computational thinking, the focus could lie on looking at actual classroom experiences applying a computational thinking skill such as metaphrasing and then tying it to self-efficacy scores.

Conclusions

Despite the lack of statistical significance in this study, interdisciplinary approaches are necessary to better improve STEM outcomes (Kennedy & Odell, 2014). Studies outline the various benefits of language learning because it can lead to academic achievement (Stewart, 2005; Cooper et al., 2008). There is also evidence that Latin uniquely produces benefits in the realm of computational thinking skills (Chater & Vitanyi, 2003). Even though the benefits of Latin have been established, there is not enough in this study to argue that taking Latin influences computational thinking and problem solving self-efficacy based on the lack of significance found in the MANOVA and ANOVA tests. The lack of findings may be due to the small sample size and the unreliability of the individual subscales.

Paired with the issues of sample size and the instrument, most of the descriptive statistics point to there not being an influence of Latin on computational thinking self-efficacy. In fact, the non-Latin group recorded higher means in most of the subscales. Nevertheless, it is noteworthy that control flow and data collection, representation, and analysis means higher for the Latin group. Based on the step-by-step processes of involved with Latin translating, it is possible other studies could explore how these skills would transfer to computational thinking self-efficacy. The lack of a distinguishable difference between the Latin group and non-Latin group in these categories may be of little concern because the means of both groups are fairly high, all averaging a score between *somewhat agree* and *strongly agree*.

REFERENCES

Ahamed, S. I., Brylow, D., Ge, R., Madiraju, P., Merrill, S. J., Struble, C. A., & Early, J. P. (2010). Computational thinking for the sciences: a three day workshop for high school science teachers. In *Proceedings of the 41st ACM technical symposium on Computer science education* (pp. 42-46). ACM.

American Councils for International Education. (2017). *The national K-12 foreign language enrollment survey report*. Retrieved from https://www.americancouncils.org/sites/default/files/FLE-report-June17.pdf

Anderson, J. D. (1975). Latin, English vocabulary, and declining SAT's. *Classical Journal*, 42-46.

Avitus, A. G. (2018). Spoken Latin: Learning, Teaching, Lecturing and Research. *Journal of Classics Teaching*, *19*(37), 46-52.

Bamber, M. D., & Schneider, J. K. (2016). Mindfulness-based meditation to decrease stress and anxiety in college students: A narrative synthesis of the research. *Educational Research Review*, *18*, 1-32.

Bandura, A. (1977). Self-efficacy: toward a unifying theory of behavioral change. *Psychological review*, *84*(2), 191.

Bandura, A. (1982). Self-efficacy mechanism in human agency. *American psychologist*, *37*(2), 122.

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

Bandura, A. (1993). Perceived self-efficacy in cognitive development and functioning. *Educational psychologist*, *28*(2), 117-148.

Bandura, A. (Ed.). (1995). *Self-efficacy in changing societies*. Cambridge university press.

Bandura, A. (1997). *Self-efficacy*: *The exercise of control*. NY: Freeman.

Bandura, A., & Adams, N. E. (1977). Analysis of self-efficacy theory of behavioral change. *Cognitive therapy and research*, *1*(4), 287-310.

Barr, D., Harrison, J., & Conery, L. (2011). Computational thinking: A digital age skill for everyone. *Learning & Leading with Technology*, *38*(6), 20-23.

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: what is involved and what is the role of the computer science education community?. *ACM Inroads*, *2*(1), 48-54.

Bennett, C. E. (1908). *New Latin Grammar*. Bolchazy-Carducci Publishers.

Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & Education*, *72*, 145-157.

Bleeker, M. M., & Jacobs, J. E. (2004). Achievement in math and science: Do mothers' beliefs matter 12 years later? *Journal of Educational Psychology*, *96*(1), 97–109.

Bong, M., & Skaalvik, E. M. (2003). Academic self-concept and self-efficacy: How different are they really?. *Educational psychology review*, *15*(1), 1-40.

Breiner, J. M., Harkness, S. S., Johnson, C. C., & Koehler, C. M. (2012). What is STEM? A discussion about conceptions of STEM in education and partnerships. *School Science and Mathematics*, *112*(1), 3-11.

Chater, N., & Manning, C. D. (2006). Probabilistic models of language processing and acquisition. *Trends in cognitive sciences*, *10*(7), 335-344.

Chater, N., & Vitányi, P. (2003). Simplicity: A unifying principle in cognitive science?. *Trends in cognitive sciences*, *7*(1), 19-22.

Clements, D. H. (1999). The future of educational computing research: The case of computer programming. *Information Technology in Childhood Education Annual*, *1999*(1), 147-179.

Cohen, J. (1977). *Statistical power analysis for the behavioral sciences* (rev. ed.). New York Academic Press.

Cooper, T. C., II, D. J. Y., Wisenbaker, J. M., Jahner, D., Webb, E., & Wilbur, M. L. (2008). Foreign language learning and SAT verbal scores revisited. *Foreign Language Annals*, *41*(2), 200-217.

Creswell, J. W. (2008). *Educational research: Planning, conducting, and evaluating quantitative and qualitative research*. Upper Saddle River, NJ: Prentice Hall.

Crowley, K., Callanan, M. A., Tenenbaum, H. R., & Allen, E. (2001). Parents explain more often to boys than to girls during shared scientific thinking. *Psychological Science*, *12*(3), 258-261.

Dattalo, P. (2013). *Analysis of multiple dependent variables*. Oxford University Press.

Denning, P. J. (2017). Computational thinking in science. *American Scientist*, *105*(1), 13-17.

DeSchryver, M. D., & Yadav, A. (2015). Creative and computational thinking in the context of new literacies: Working with teachers to scaffold complex technology-mediated approaches to teaching and learning. *Journal of Technology and Teacher Education*, *23*(3), 411-431.

Dierbach, C., Hochheiser, H., Collins, S., Jerome, G., Ariza, C., Kelleher, T., ... & Kaza, S. (2011). A model for piloting pathways for computational thinking in a general education curriculum. In *Proceedings of the 42nd ACM technical symposium on Computer science education* (pp. 257-262).

Field, A. (2009). *Discovering statistics using SPSS* (3rd ed.). London: Sage.

Filzmoser, P. (2004). A multivariate outlier detection method. In S. Aivazian, P. Filzmoser, & Y. Kharin (Eds.), *Proceedings of the Seventh International Conference on Computer Data Analysis and Modeling: Vol. 1* (pp. 18 –22).

Forehand, M. (2010). Bloom's taxonomy. *Emerging perspectives on learning, teaching, and technology, 41*(4), 47-56.

Futschek, G. (2006). Algorithmic thinking: the key for understanding computer science. *International conference on informatics in secondary schools-evolution and perspectives*, 159-168.

Gall, M. Gall, J. & Borg, W. (2010) *Applying educational research: How to read, do, and use research to solve problems or practice*. Boston, MA: Allyn & Bacon/Pearson.

Glen, S. (2016, October 9). Pillai's Trace. Retrieved from https://www.statisticshowto.datasciencecentral.com/pillais-trace/

Gonzalez-DeHass, A. R., Willems, P. P., & Holbein, M. F. D. (2005). Examining the relationship between parental involvement and student motivation. *Educational Psychology Review, 17*(2), 99–123.

Graham, S., & Weiner, B. (1996). Theories and principles of motivation. *Handbook of educational psychology*, *4*, 63-84.

Grice, J. W., & Iwasaki, M. (2007). A truly multivariate approach to MANOVA. *Applied Multivariate Research*, *12*(3), 199-226.

Grishman, R. (1986). *Computational linguistics: an introduction*. Cambridge University Press.

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher, 42(1),* 38-43.

Gurian, M. (2010). *Boys and girls learn differently! A guide for teachers and parents: Revised 10th anniversary edition*. John Wiley & Sons.

Guzdial, M. (2008). Education Paving the way for computational thinking. *Communications of the ACM*, *51*(8), 25-27.

Hambrusch, S., Hoffmann, C., Korb, J. T., Haugan, M., & Hosking, A. L. (2009). A multidisciplinary approach towards computational thinking for science majors. *ACM SIGCSE Bulletin*, *41*(1), 183-187.

Hatlevik, O. E., Throndsen, I., Loi, M., & Gudmundsdottir, G. B. (2018). Students' ICT self-efficacy and computer and information literacy: Determinants and relationships. *Computers & Education*, *118*, 107-119.

Hodges, C. B. (2005). *Self-efficacy, motivational email, and achievement in an asynchronous mathematics course*. Unpublished doctoral dissertation, Virginia Polytechnic Institute and State University, Blacksburg, VA.

Holliday, L. R. (2012). The Benefits of Latin?. *Educational Research Quarterly*, *36*(1), 3.

Horning, J. (1971). A procedure for grammatical inference. In *IFIP Congress 71,* pp. 519-523.

Horwitz, E. K., Horwitz, M. B., & Cope, J. (1986). Foreign language classroom anxiety. *Modern Language Journal*, *70*(2), 125-132.

Hunsader, P. D. (2002). Why Boys Fail-and what we can do about it. *Principal-Arlington*, *82*(2), 52-55.

Jackson, D., & Jackson, M. (1996). Problem decomposition for reuse. *Software Engineering Journal*, *11*(1), 19-30

Jonassen, D. H. (2008). Instructional design as design problem solving: An iterative process. *Educational Technology*, 21-26.

Kennedy, T. J., & Odell, M. R. L. (2014). Engaging students in STEM education. *Science Education International*, *25*(3), 246-258.

Knudsvig, G. M., Seligson, G. M., & Craig, R. S. (1986). Latin for reading: a beginner's textbook with exercises. University of Michigan Press.

Landau, R., Mulder, G., Holmes, R., Borinskaya, S., Kang, N., & Bordeianu, C. (2014). INSTANCES: incorporating computational scientific thinking advances into education and science courses. *Concurrency and Computation: Practice and Experience*, *26*(13), 2316-2328.

Larman, C., & Basili, V. R. (2003). Iterative and incremental developments. a brief history. *Computer*, *36*(6), 47-56.

Lee, C.-Y., & Witta, E. L. (2001). Online students' perceived self-efficacy: Does it change? Paper presented at the *National Convention of the Association for Educational Communications and Technology*, Atlanta, GA.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, & Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads*, *2*(1), 32-37.

Lin, T. J., & Tsai, C. C. (2013). A multi-dimensional instrument for evaluating Taiwanese high school students' science learning self-efficacy in relation to their approaches to learning science. *International Journal of Science and Mathematics Education*, *11*(6), 1275-1301.

Lombardi, M. M. (2007). Authentic learning for the 21st century: An overview. *Educause learning initiative*, *1*(2007), 1-12.

Lu, J. J., & Fletcher, G. H. (2009). Thinking about computational thinking. *ACM SIGCSE Bulletin*, *41*(1), 260-264.

Maas, C. J., & Hox, J. J. (2005). Sufficient sample sizes for multilevel modeling. *Methodology*, 1(3), 86-92.

MacIntyre, P. D., Noels, K. A., & Clément, R. (1997). Biases in self-ratings of second language proficiency: The role of language anxiety. *Language Learning*, *47*, 265-287.

Maddux, J. E., Norton, L. W., & Stoltenberg, C. D. (1986). Self-efficacy expectancy, outcome expectancy, and outcome value: Relative effects on behavioral intentions. *Journal of Personality and Social Psychology*, *51*(4), 783.

Mann, J. (1994). Bridging the Gender Gap: How Girls Learn. In *Streamlined Seminar* (Vol. 13, No. 2, p. n2).

Manson, J. R., & Olsen, R. J. (2010). Diagnostics and rubrics for assessing learning across the computational science curriculum. *Journal of Computational Science*, *1*(1), 55-61.

Markus, D. D., & Ross, D. P. (2004). Reading proficiency in Latin through expectations and visualization. *The classical world*, 79-93.

Masciantonio, R. (1977). Tangible benefits of the study of Latin: A review of research. *Foreign Language Annals*, *10*(4), 375-382.

Mason, J. (2015). Scoping Research with a Focus on Questioning. In *Conference Paper,* *December*.

May, K. (2009). *Mathematics self-efficacy and anxiety questionnaire*. (Unpublished doctoral dissertation). University of Georgia, Athens, GA.

Mcfadden, P. (2008). Advanced Latin Without Translations? Interactive Text-Marking as an Alternative Daily Preparation. *Committee for the Promotion of Latin* 4.1 p.1-15.

Miller, L. D., Soh, L. K., Chiriacescu, V., Ingraham, E., Shell, D. F., Ramsay, S., & Hazley, M. P. (2013). Improving learning of computational thinking using creative thinking exercises in CS-1 computer science courses. In *Frontiers in Education Conference, 2013 IEEE* (pp. 1426-1432).

Miller, L. D., Soh, L. K., Chiriacescu, V., Ingraham, E., Shell, D. F., & Hazley, M. P. (2014). Integrating computational and creative thinking to improve learning and performance in CS1. In *Proceedings of the 45th ACM technical symposium on Computer science education* (pp. 475-480). ACM.

Mills, G. E. (2011). *Action research: A guide for the teacher researcher (with MyEducationLab). (4th ed.)*. Upper Saddle River, NJ: Pearson/ Allyn & Bacon.

Mills, N., Pajares, F., & Herron, C. (2006). A reevaluation of the role of anxiety: Self-efficacy, anxiety, and their relation to reading and listening proficiency. *Foreign language annals*, *39*(2), 276-295.

Mizelle, N. B., & Irvin, J. L. (2000). Transition from middle school into high school. *Middle School Journal*, *31*(5), 57-61.

Moret, L., Nguyen, J. M., Pillet, N., Falissard, B., Lombrail, P., & Gasquet, I. (2007).

Improvement of psychometric properties of a scale measuring inpatient satisfaction with care: a better response rate and a reduction of the ceiling effect. *BMC health services research*, *7*(1), 197.

Noormohamadi, R. (2009). On the Relationship Between Language Learning Strategies and Foreign Language Anxiety. *Journal of Pan-Pacific Association of Applied Linguistics*, *13*(1), 39-52.

Niyogi, P. (2006). *The computational nature of language learning and evolution*. Cambridge, MA: MIT press.

Olabe, J. C., Olabe, M. A., Basogain, X., & Castaño, C. (2011). Programming and robotics with Scratch in primary education. *Education in a technological world: communicating current and emerging research and technological efforts*, 356-363.

Pajares, F., Johnson, M. J., & Usher, E. L. (2007). Sources of writing self-efficacy beliefs of elementary, middle, and high school students. *Research in the Teaching of English*, 104-120.

Perkins, D. N., & Salomon, G. (1992). Transfer of learning. International encyclopedia of education, 2, 6452-6457.

Qualls, J. A., & Sherrell, L. B. (2010). Why computational thinking should be integrated into the curriculum. *Journal of Computing Sciences in Colleges*, *25*(5), 66-71.

Ramalingam, V., LaBelle, D., & Wiedenbeck, S. (2004). Self-efficacy and mental models in learning to program. *Paper presented at the Ninth Annual SIGCSE Conference on Innovation and Technology in Computer Science Education (ITiCSE 2004)*, Leeds, United Kingdom.

Repenning, A., Webb, D., & Ioannidou, A. (2010). Scalable game design and the development of a checklist for getting computational thinking into public schools. In *Proceedings of the 41st ACM technical symposium on Computer science education* (pp. 265-269). ACM.

Rittmayer, A. D., & Beier, M. E. (2008). Overview: Self-efficacy in STEM. *SWE-AWE CASEE Overviews*, 1-12.

Rockinson-Szapkiw, A. J. (2013). Statistics Guide. Retrieved from www.amandaszapkiw.com

Roscoe, J. F., Fearn, S., & Posey, E. (2014). Teaching computational thinking by playing games and building robots. In *Interactive Technologies and Games (iTAG), 2014 International Conference on* (pp. 9-12).

Russell, K. (2018). Read like a Roman: teaching students to read in Latin word order. *Journal of Classics Teaching*, 19(37), 17-29.

Sanders, M. (2009). STEM, STEM education, STEM mania. *Technology Teacher, 68*(4), 20–26.

Sanford, K. (2005). Gendered literacy experiences: The effects of expectation and opportunity for boys' and girls' learning. *Journal of Adolescent & Adult Literacy, 49*(4), 302-315.

Scheiner, S. M. (2001). Multiple response variables and multi-species interactions. *Design and analysis of ecological experiments*, 99-133.

Schmidt, B. (2018). The Humanities are in crisis. Retrieved from https://www.theatlantic.com/ideas/archive/2018/08/the-humanities-face-a-crisisof-confidence/567565/

Schunk, D. H. (1991). Self-efficacy and academic motivation. *Educational psychologist*, *26*(3-4), 207-231.

Schunk, D. H., & Pajares, F. (2009). Self-efficacy theory. In K. R. Wentzel, & A. Wigfield

(Eds.). Handbook of motivation at school (pp. 35–53). New York: Routledge.

Scida, E. E., & Jones, J. E. (2017). The impact of contemplative practices on foreign language anxiety and learning. *Studies in Second Language Learning and Teaching*, 7(4), 573-599.

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies*, *18*(2), 351-380.

Settle, A., Franke, B., Hansen, R., Spaltro, F., Jurisson, C., Rennert-May, C., & Wildeman, B. (2012). Infusing computational thinking into the middle-and high-school curriculum. In *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education* (pp. 22-27). ACM.

Settle, A., & Perkovic, L. (2010). Computational thinking across the curriculum: a conceptual framework.

Shell, D. F., Hazley, M. P., Soh, L. K., Miller, L. D., Chiriacescu, V., & Ingraham, E. (2014). Improving learning of computational thinking using computational creativity exercises in a college CSI computer science course for engineers. In *Frontiers in Education Conference (FIE), 2014 IEEE* (pp. 1-7).

Somasundaram, R. S., & Nedunchezhian, R. (2012). Missing value imputation using refined mean substitution. *International Journal of Computer Science Issues (IJCSI)*, *9*(4), 306.

Sparks, R. L., Ganschow, L., Fluharty, K., & Little, S. (1995). An exploratory study on the effects of Latin on the native language skills and foreign language aptitude of students with and without learning disabilities. *The Classical Journal*, *91*(2), 165-184.

Stewart, J. H. (2005). Foreign language study in elementary schools: Benefits and implications for achievement in reading and math. *Early Childhood Education Journal*, *33*(1), 11-16.

Talsma, K., Schüz, B., Schwarzer, R., & Norris, K. (2018). I believe, therefore I achieve (and vice versa): A meta-analytic cross-lagged panel analysis of self-efficacy and academic performance. *Learning and Individual Differences*, *61*, 136-150.

Tran, Y. (2018). Computational Thinking Equity in Elementary Classrooms: What Third-Grade Students Know and Can Do. *Journal of Educational Computing Research*, doi: 0735633117743918.

Tsai, M. J., Wang, C. Y., & Hsu, P. F. (2018). Developing the Computer Programming Self-Efficacy Scale for Computer Literacy Education. *Journal of Educational Computing Research*, 0735633117746747.

Tuan, H. L., Chin, C. C., & Shieh, S. H. (2005). The development of a questionnaire to measure students' motivation towards science learning. *International Journal of Science Education*, *27*(6), 639-654.

U.S. Department of Education (n.d.). *Science, technology, engineering and math: Education for global leadership*. Retrieved from https://www.ed.gov/stem

Vekiri, I., & Chronaki, A. (2008). Gender issues in technology use: Perceived social support, computer self-efficacy and value beliefs, and computer use beyond school. *Computers & Education*, *51*(3), 1392-1404.

Voskoglou, M. G., & Buckley, S. (2012). Problem solving and computational thinking in a learning environment. *arXiv preprint arXiv:1212.0750*.

Webb, H., & Rosson, M. B. (2013). Using scaffolded examples to teach computational thinking concepts. In *Proceeding of the 44th ACM technical symposium on Computer* *science education* (pp. 95-100). ACM.

Weese, J. L., & Feldhausen, R. (2017). *STEM Outreach: Assessing computational thinking and problem solving*. Paper presented at the 2017 ASEE Annual Conference & Exposition, Columbus, Ohio.

Wells, G. (1995). Language and the inquiry-oriented curriculum. *Curriculum Inquiry*, *25*(3), 233-269.

Weng, X., & Wong, G. K. (2017). Integrating computational thinking into English dialogue learning through graphical programming tool. In *Teaching, Assessment, and Learning for Engineering (TALE), 2017 IEEE 6th International Conference on* (pp. 320-325). IEEE.

Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012, February). The fairy performance assessment: measuring computational thinking in middle school. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (pp. 215-220). ACM.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35.

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical transactions of the royal society of London A: mathematical, physical and engineering sciences*, *366*(1881), 3717-3725.

Wolz, U., Stone, M., Pearson, K., Pulimood, S. M., & Switzer, M. (2011). Computational thinking and expository writing in the middle school. *ACM Transactions on Computing Education (TOCE), 11*(2), 9.

Wolz, U., Stone, M., Pulimood, S. M., & Pearson, K. (2010). Computational thinking via interactive journalism in middle school. In *Proceedings of the 41st ACM technical symposium on Computer science education* (pp. 239-243).

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education (TOCE), 14*(1), 5.

Yaşar, O. (2018). A new perspective on computational thinking. *Communications of the ACM*, *61*(7), 33-39.

Zeldin, A. L., & Pajares, F. (2000). Against the odds: Self-efficacy beliefs of women in mathematical, scientific, and technological careers. *American Educational Research Journal*, *37*(1), 215-246.

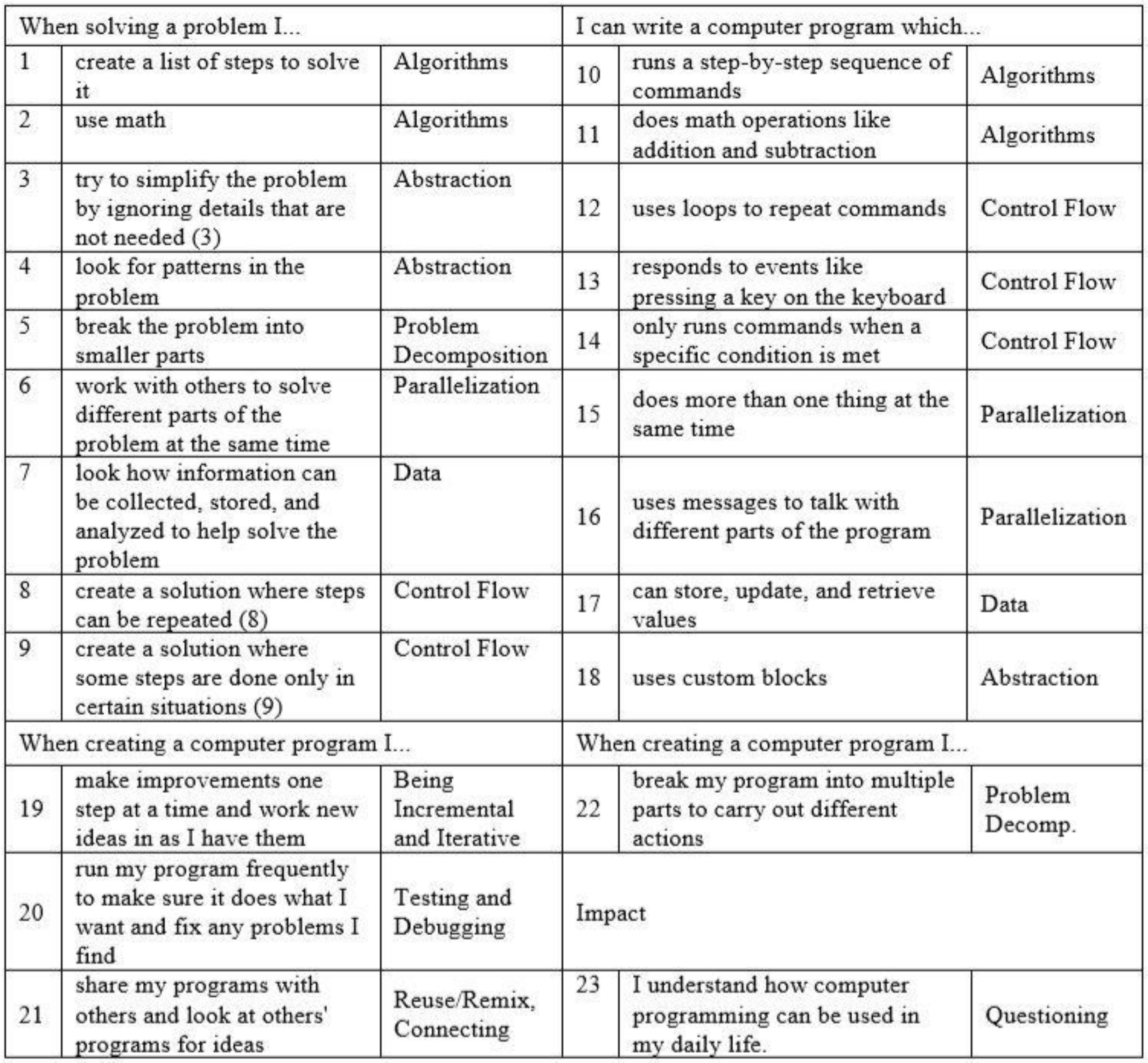
Zhu, Z. (2007). Gender differences in mathematical problem solving patterns: A review of literature. *International Education Journal*, *8*(2), 187-203.

Zimmerman, B. J. (2000). Self-efficacy: An essential motive to learn. *Contemporary educational psychology*, *25*(1), 82-91.

APPENDICES

Appendix A

Computational Thinking and Problem Solving Self-Efficacy Scale



When solving a problem I...			I can write a computer program which...		
1	create a list of steps to solve it	Algorithms	10	runs a step-by-step sequence of commands	Algorithms
2	use math	Algorithms	11	does math operations like addition and subtraction	Algorithms
3	try to simplify the problem by ignoring details that are not needed (3)	Abstraction	12	uses loops to repeat commands	Control Flow
4	look for patterns in the problem	Abstraction	13	responds to events like pressing a key on the keyboard	Control Flow
5	break the problem into smaller parts	Problem Decomposition	14	only runs commands when a specific condition is met	Control Flow
6	work with others to solve different parts of the problem at the same time	Parallelization	15	does more than one thing at the same time	Parallelization
7	look how information can be collected, stored, and analyzed to help solve the problem	Data	16	uses messages to talk with different parts of the program	Parallelization
8	create a solution where steps can be repeated (8)	Control Flow	17	can store, update, and retrieve values	Data
9	create a solution where some steps are done only in certain situations (9)	Control Flow	18	uses custom blocks	Abstraction
When creating a computer program I...			When creating a computer program I...		
19	make improvements one step at a time and work new ideas in as I have them	Being Incremental and Iterative	22	break my program into multiple parts to carry out different actions	Problem Decomp.
20	run my program frequently to make sure it does what I want and fix any problems I find	Testing and Debugging	Impact		
21	share my programs with others and look at others' programs for ideas	Reuse/Remix, Connecting	23	I understand how computer programming can be used in my daily life.	Questioning

Appendix B

Assent Form

ASSENT FORM

COMPUTATIONAL THINKING SELF-EFFICACY IN HIGH SCHOOL LATIN LANGUAGE LEARNING

You are invited to be in a research study being done by Dennis Dickerson, Jr. from the University of Memphis. You are invited because your experience in computer science courses and potentially Latin too.

If you agree to be in the study, you will be asked to fill out a 23-item survey in a Google form. The form will contain preliminary questions about your grade and levels of Latin or other foreign language taken.

You will not receive any rewards or payment for taking part in the study.

Your family will know that you are in the study. If anyone else is given information about you, they will not know your name. A number or initials will be used instead of your name.
If something makes you feel bad while you are in the study, please tell Dennis Dickerson, Jr. If you decide at any time you do not want to finish the study, you may stop whenever you want.

You can ask Dennis Dickerson, Jr. questions any time about anything in this study. You can also ask your parent any questions you might have about this study.

Signing this paper means that you have read this or had it read to you, and that you want to be in the study. If you do not want to be in the study, do not sign the paper. Being in the study is up to you, and no one will be mad if you do not sign this paper or even if you change your mind later. You agree that you have been told about this study and why it is being done and what to do.

____ ____________________

Signature of Person Agreeing to be in the Study Date Signed

Appendix C

Parental Permission Form

Parental Permission for Your Child to Participate in a Research Study

COMPUTATIONAL THINKING SELF-EFFICACY IN HIGH SCHOOL LATIN LANGUAGE LEARNING

WHY IS YOUR CHILD BEING INVITED TO TAKE PART IN THIS RESEARCH?

Your child is being invited to take part in a research study about computational thinking self-efficacy in Latin language learning. Your child is being invited to take part in this research study because of your experience in computer science courses and potentially Latin too. If he volunteers to take part in this study, he will be one of about 68 students to do so in his high school.

WHO IS DOING THE STUDY?

The person in charge of this study is Dennis Clark Dickerson, Jr. of the University of Memphis Department of Instruction and Curriculum Leadership: Instructional Design and Technology. He is being guided in this research by Dr. Andrew Tawfik. There may be other people on the research team assisting at different times during the study.

WHAT IS THE PURPOSE OF THIS STUDY?

The purpose of this study is to see whether taking Latin courses influences a student's computational thinking self-efficacy.

By doing this study, we hope to learn if computational thinking skills are present in Latin courses and thus could be transferred to other computer science courses.

ARE THERE REASONS WHY YOUR CHILD SHOULD NOT TAKE PART IN THIS STUDY?

Your child should not take part in this study if he has not taken at least a full semester of a computer science course.

WHERE IS THE STUDY GOING TO TAKE PLACE AND HOW LONG WILL IT LAST?

The research procedures will be conducted at Christian Brothers High School. He will need at least 10-15 minutes to fill out the form.

WHAT WILL YOUR CHILD BE ASKED TO DO?

Your son will be asked to fill out a 23-item survey in a Google form. The form will contain preliminary questions about his grade and levels of Latin or other foreign language taken.

WHAT ARE THE POSSIBLE RISKS AND DISCOMFORTS?

To the best of our knowledge, the things your child will be doing have no more risk of harm than your child would experience in everyday life.

WILL YOUR CHILD BENEFIT FROM TAKING PART IN THIS STUDY?

There is no guarantee that your child will get any benefit from taking part in this study. His willingness to take part, however, may, in the future, help society as a whole better understand this research topic.

DOES YOUR CHILD HAVE TO TAKE PART IN THE STUDY?

If you decide to allow your child take part in the study, it should be because your child really wants to volunteer. Your child will not lose any benefits or rights your child would normally have if your child chooses not to volunteer. Your child can stop at any time during the study and still keep the benefits and rights your child had before volunteering. (*Add the following, if applicable:* If you or your child decides not to take part in this study, your child's decision will have no effect on the quality of care, services, etc., your child receives). *Add the following for student volunteers:* As a student, if your child decides not to take part in this study, your child's choice will have no effect on your child's academic status or grade in the class.

IF YOUR CHILD DON'T WANT TO TAKE PART IN THE STUDY, ARE THERE OTHER CHOICES?

If your child does not want to be in the study, there are no other choices except not to take part in the study.

WHAT WILL IT COST YOU FOR YOUR CHILD TO PARTICIPATE?

There are no costs associated with taking part in the study.

WILL YOUR CHILD RECEIVE ANY REWARDS FOR TAKING PART IN THIS STUDY?

Your child will not receive any rewards or payment for taking part in the study.

WHO WILL SEE THE INFORMATION THAT YOUR CHILD PROVIDES?

We will make every effort to keep private all research records that identify your child to the extent allowed by law.

Your child's information will be combined with information from other children taking part in the study. When we write about the study to share it with other researchers, we will write about the combined information we have gathered. Your child will not be personally identified in these written materials. We may publish the results of this study; however, we will keep your child's name and other identifying information private.

We will make every effort to prevent anyone who is not on the research team from knowing that your child gave us information, or what that information is. Participants will be given a number so that names are not attached to the data. The data will be stored on a secure device and only uploaded to a secure software for analysis.

We will keep private all research records that identify you to the extent allowed by law. However, there are some circumstances in which we may have to show your information to other people. We may be required to show information which identifies you to people who need to be sure we have done the research correctly; these would be people from such organizations as the University of Memphis.

CAN YOUR CHILD'S TAKING PART IN THE STUDY END EARLY?

If your child decides to take part in the study your child still have the right to decide at any time that your child no longer want to continue. Your child will not be treated differently if your child decides to stop taking part in the study.

The individuals conducting the study may need to withdraw your child from the study. This may occur if your child are not able to follow the directions they give your child, if they find that your child's being in the study is more risk than benefit to your child, or if the agency funding the study decides to stop the study early for a variety of scientific reasons.

WHAT IF YOUR CHILD HAVE QUESTIONS, SUGGESTIONS, CONCERNS, OR COMPLAINTS?

Before you decide whether to accept this invitation for your child to take part in the study, please ask any questions that might come to mind now. Later, if you have questions, suggestions, concerns, or complaints about the study, you can contact the investigator, Dennis Dickerson, Jr. at 615-491-6172. If you have any questions about your child's rights as a volunteer in this research, contact the Institutional Review Board staff at the University of Memphis at 901-678-3074. We will give you a signed copy of this permission form to take with you.

WHAT ELSE DOES YOUR CHILD NEED TO KNOW?

The University of Memphis is providing financial support and/or material for this study.

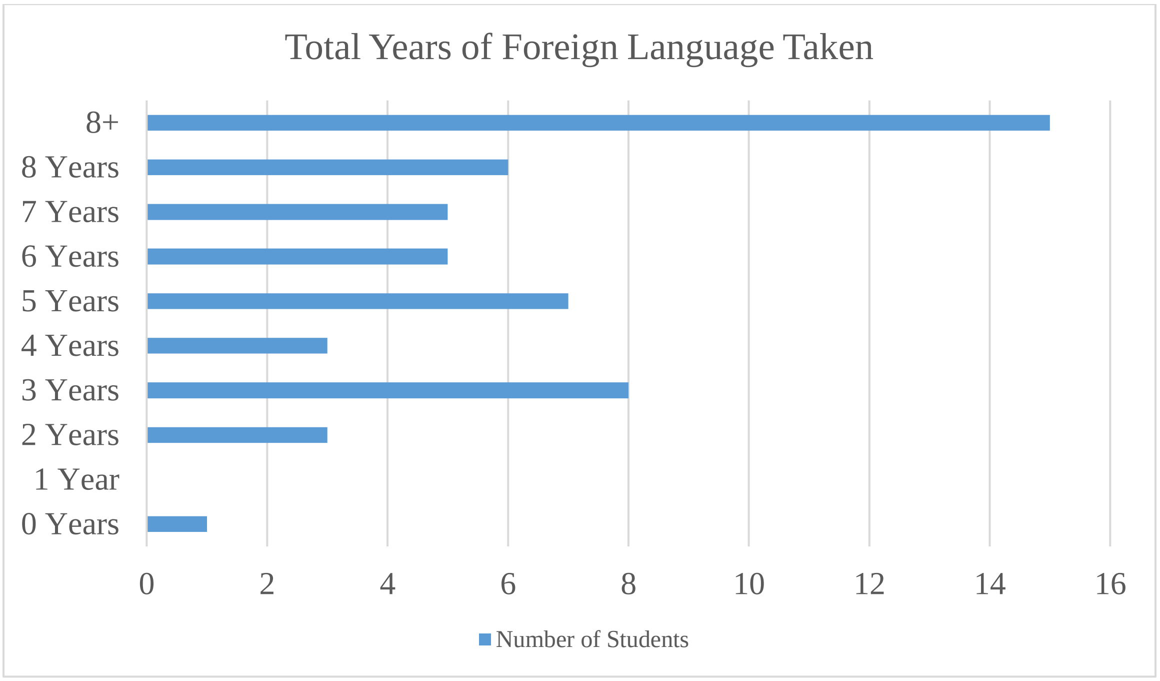
____ ____________
Signature of person agreeing to take part in the study Date

____
Printed name of person agreeing to take part in the study

____ ____________
Name of [authorized] person obtaining informed consent Date

Appendix D

Bar Graph of Total Years of Foreign Language


Total Years of Foreign Language Taken
8 Years
7 Years
6 Years
5 Years
4 Years
3 Years
2 Years
1 Year
0 Years
Number of Students