A MIXED-METHOD STUDY:

PRE-SERVICE TEACHERS' LEARNING OF

BLOCK-BASED PROGRAMMING AND COMPUTATIONAL THINKING

İLKE TOPAL

BOĞAZİÇİ UNIVERSITY

2023

A MIXED-METHOD STUDY:

PRE-SERVICE TEACHERS' LEARNING OF

BLOCK-BASED PROGRAMMING AND COMPUTATIONAL THINKING

Thesis submitted to

the Institute for Graduate Studies in Social Sciences

in partial fulfillment of the requirements for the degree of

Master of Arts

in

Educational Technology

by

İlke Topal

Boğaziçi University

2023

A Mixed-Method Study:

Pre-Service Teachers' Learning of

Block-Based Programming and Computational Thinking

The thesis of İlke Topal

has been approved by:

Assist. Prof. Duygu Umutlu _____
(Thesis Advisor)

Assist. Prof. Mutlu Şen-Akbulut _____

Prof. Süleyman Sadi Seferoğlu _____
(External Member)

May 2023

DECLARATION OF ORIGINALITY

I, İlke Topal, certify that

- I am the sole author of this thesis and that I have fully acknowledged and documented in my thesis all source of ideas and words, including digital resources, which have been produced or published by another person or institution;

- this thesis contains no material that has been submitted or accepted for a degree or diploma in any other educational institution;

- this is a true copy of the thesis approved by my advisor and thesis committee at

  Boğaziçi University, including final revisions required by them.

Signature……..

Date……..

ABSTRACT

A Mixed-Method Study: Pre-Service Teachers' Learning of Block-Based Programming and Computational Thinking

The purpose of this explanatory sequential mixed-method study is to examine the effect of pair programming on pre-service teachers' computational thinking skills and coding achievement and to explore pre-service teachers' learning experiences of programming. The participants were 36 pre-service teachers who took the Instructional Technologies and Material Development course at Boğaziçi University. Online tasks and activities were designed on a web-based block coding platform (Scratch) and given to the two sections of the course. In the sections, the experimental group completed the Scratch tasks in pairs and, the control group completed the same Scratch tasks individually. The analysis showed that there was no statistically significant difference between the computational thinking skills scores and coding achievement test scores of the experimental group and the control group. However, when the pre-test and post-test results were compared within each group, the experimental and the control group significantly increased their computational thinking and coding achievement. The themes that emerged in the qualitative analysis, on the other hand, showed the importance of pair programming. This study underscored the benefits of pair programming in early error detection, increasing coding quality, facilitating knowledge transfer, enhancing problem-solving, and developing creativity. This study contributes significantly to the literature by emphasizing the importance of programming instruction given to pre-service teachers by using pair programming.

ÖZET

Karma Yöntemli Bir Çalışma: Öğretmen Adaylarının Blok-Tabanlı Programlama Öğrenimi ve Bilgi İşlemsel Düşünme

Bu sıralı açıklayıcı karma yöntemli çalışmanın amacı, eşli programlamanın öğretmen adaylarının bilgi-işlemsel düşünme becerileri ve kodlama başarısı üzerindeki etkisini ve öğretmen adaylarının programlama öğrenme deneyimlerini incelemektir. Çalışmaya Boğaziçi Üniversitesi'nde Öğretim Teknolojileri ve Materyal Geliştirme dersini alan 36 öğretmen adayı katılmıştır. Bu bağlamda bir çevrimiçi blok tabanlı kodlama platformu olan Scratch üzerinde kodlama görevleri ve etkinlikleri tasarlanmış ve dersin iki şubesine uygulanmıştır. Deney grubu olarak belirlenen ilk şubede öğrenciler Scratch etkinliklerini eşleri ile yaparak ve kontrol grubu belirlenen diğer şubede ise öğrenciler aynı Scratch görevlerini bireysel olarak tamamlamıştır. Yapılan veri analizleri, deney grubu ve kontrol grubunun bilgi işlemsel düşünme becerileri puanları ile kodlama başarı testi puanları arasında istatistiksel açıdan anlamlı bir fark olmadığını göstermiştir. Fakat, her bir grup içinde ön test ve son test sonuçları karşılaştırıldığında, çalışmadaki iki grubun da bilgi işlemsel düşünme becerisinin ve kodlama başarısının anlamlı düzeyde arttığı görülmüştür. Diğer taraftan, nitel analiz sonucunda ortaya çıkan temalar eşli programlamanın önemini göstermiştir. Bu çalışma, erken hata tespitinde, kod kalitesini artırmada, bilgi transferini kolaylaştırmada, problem çözme yeteneğini geliştirmede ve yaratıcılığı geliştirmede eşli programlamanın faydalarını vurgulamıştır. Bu çalışma eşli programlamayı da kullanarak öğretmen adaylarına verilen kodlama eğitiminin önemini vurgulayarak literatüre önemli bir katkıda bulunmuştur.

ACKNOWLEDGEMENTS

I would like to acknowledge the individuals who provided invaluable assistance throughout my thesis journey. Foremost, I extend my appreciation to my thesis advisor, Assist. Prof. Duygu Umutlu, for her unwavering dedication, encouragement, and careful direction during my research. Also, I would like to again express my gratitude to Assist. Prof. Duygu Umutlu for facilitating my access to her students and allowing me to conduct the study during her lectures. This thesis was accomplished with her exceptional support, expert guidance, and constructive feedback.

I would like to extend my greatest gratitude to my thesis committee members, namely Assist. Prof. Mutlu Şen-Akbulut and Prof. Süleyman Sadi Seferoğlu, for their valuable comments, feedback, and suggestions that have proved to be instrumental in successfully completing my thesis.

I am thankful for my friends who supported me during this process. First of all, I would like to thank my housemates Gözde Nur Doğuer and Elvan Güler for their unwavering support. And I am thankful for my "Beyaz Yakasızlar" and “Altınoluk” friend groups. I also want to thank to my friends, Gülenay Şahin, Onur Kavalcı and İlayda Üzel for their help and support.

My incredible family, including my mother, İlknur Topal, my father, Osman Topal, and my brother, Emre Topal, has my sincere appreciation for their unwavering love and support throughout this challenging period. Without my family’s encouragement, I would not have been able to complete this writing process.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION 1

1.1 Significance of the study 5

1.2 Purpose of the study 6

1.3 Research questions 6

1.4 Research hypotheses 6

1.5 Study organization 7

CHAPTER 2: LITERATURE REVIEW 8

2.1 Computational thinking 8

2.2 Programming 16

2.3 Using scratch 20

CHAPTER 3: METHOD 22

3.1 Research design 22

3.2 Context and participants 23

3.3 Instruments for data collection 25

3.4 Data collection procedures 27

3.5 Data analysis 28

3.6 Trustworthiness 29

CHAPTER 4: RESULTS 31

4.1 CT skills of pre-service teachers who are involved in pair programming and individual programming 31

4.2 Coding achievement of pre-service teachers who are involved in pair programming and individual programming 33

4.3 Within-group comparisons of pre-service teachers' CT skills and coding achievement 36
4.4 Exploring the pre-service teachers' learning experiences of programming 39
4.5 Integration of findings 46
CHAPTER 5: DISCUSSION AND CONCLUSION 49
5.1 Recommendations and implications for future research 58
5.2 Limitations of the study 60
APPENDIX A: PARTICIPANT INFORMATION AND CONSENT FORM 61
APPENDIX B: CODING ACHIEVEMENT TEST 63
APPENDIX C: COMPUTATIONAL THINKING SKILLS SCALE 78
APPENDIX D: INTERVIEW QUESTIONS 82
APPENDIX E: THE PERMISSION FROM ETHICS COMMITTEE OF BOGAZICI UNIVERSITY 84
APPENDIX F: DESCRIPTION OF SCRATCH TASKS 85
APPENDIX G: TRANSLATED QUOTES 90
REFERENCES 94

LIST OF TABLES

Table 1. Demographic Information of Participants..... 24

Table 2. Scratch Tasks Completed During the Study 28

Table 3. Descriptive Statistics of the Experimental and the Control Groups for CT skills for Pre-test and Post-test..... 31

Table 4. Shapiro-Wilk Result of the Pre-test for the Experimental Group and the Control Group for CT Skills 32

Table 5. Mann-Whitney U Test for Pre-test Scores of CT Skills of the Experimental Group and the Control Group 32

Table 6. Shapiro-Wilk Result of the Post-test for the Experimental Group and the Control Group for CT Skills 33

Table 7. Independent Samples T-Test for CT Skills Scores of the Experimental Group and the Control Group as Post-tet..... 33

Table 8. Descriptive Statistics of the Experimental and the Control Groups for Coding Achievement for pre-test and post-test..... 34

Table 9. Shapiro-Wilk Result of the Pre-test for the Experimental and Control Groups for Coding Achievement 34

Table 10. Independent Samples T-Test for Coding Achievement Scores of the Experimental Group and the Control Group as Pre-test 35

Table 11. Shapiro-Wilk Result of the Post-test for the Experimental and Control Groups' Coding Achievement..... 35

Table 12. Independent Samples T-Test for Coding Achievement Test Scores of the Experimental Group and the Control Group as Post-test..... 36

Table 13. Shapiro-Wilk Result of the Pre-test and Post-test of CT Skills Scores of Pre-service Teachers 36
Table 14. Paired Samples Test for Post-test and Pre-test of CT Skills Scores of the Experimental Group 37
Table 15. Wilcoxon Signed Rank Test for Post-test and Pre-test of CT Skills Scores of the Control Group 37
Table 16. Shapiro-Wilk Result of the Pre-test and Post-test Coding Achievement Scores of Pre-service Teachers 38
Table 17. Paired Samples Test for Post-test and Pre-test of Coding Achievement Scores of Pre-service Teachers 39
Table 18. Emerged Themes 39

CHAPTER 1

INTRODUCTION

In the 21st century, individuals with skills such as algorithmic thinking, problem-solving, critical thinking, creativity, and collaboration are in more demand, and the need for computational thinking (CT) skills is rising. According to the International Society for Technology in Education (ISTE) standards, CT is an essential skill that students need to gain in 21st century classrooms; therefore, teachers need to become familiar with CT to meet these students' needs (ISTE, 2016). In addition, students can learn about CT by observing their teachers displaying similar thinking skills and guiding them to utilize these skills alone (National Research Council, 2010). Yet, the majority of current CT teacher education initiatives focus on only computer science educators. (Yadav, Mayfield, Zhou, Hambrusch & Korb, 2014). Therfore, it could be suggested that teacher education curricula need to be revised to equip teachers in all subject areas with CT skills.

As a problem-solving process, CT, which is also seen as the center of learning by Passey (2017), can be developed through programming. Programming has also started to be considered one of the basic fields in many developed countries globally. Tikva and Tambouris (2021) stated that it has begun to be included in the curricula as a compulsory or elective course, especially in K-12 settings. With the increasing importance of programming in education, teachers are expected to offer innovative coding lessons (Fisher, Byrne & Tangney, 2016). Thus, it could be claimed that new approaches in teacher education should be implemented to prepare teacher candidates for teaching programming (Instefjord & Munthe, 2017).

In coding, collaborative learning takes the form of pair programming, which is one of the potential methods (Demir & Seferoğlu, 2020; Faja, 2014). Pair programming enables two students or programmers to work together when writing codes on a computer. There are several techniques for pair programming. For instance, one of the programmers operate the keyboard and focuses on typing the code and the other one (may be called as navigator or observer) observes the other's working and gives feedback, makes suggestions. The roles can be interchanged on a regular basis if needed (Braught, Eby & Wahls, 2008).

Traditionally, most pair programming techniques are executed through physical interaction between two programmers working on the same computer, talking to each other, looking at the same screen, and pointing at each other (Vanhanen & Korpi, 2007). However, with the outbreak of COVID-19 pandemic, there has existed an immediate transition toward online and virtual learning methods, particularly in higher education (Li et al., 2021). For example, in pair programming in an online environment, programmers switch roles by sharing screens and using an appropriate remote communication system instead of sharing the same computer. In online pair programming, there should be screen-sharing as a communication channel that supports role-switching (Da Silva Estácio & Prikladnicki, 2015). Under these circumstances, pair programming method can be easily adapted to the online environment (Beasley & Johnson, 2022).

Denner and Werner (2007) suggested that pair programming technique is among the most successful collaborative designs, and also helps and motivates beginner programmers, and is also widely used at K-12 levels. Also, Fisher et al. (2016) indicated that learning is more self-directed when students collaborate with others. Pair programming offers scaffolding for group members as they pass

information on to one another, enabling each of them to accomplish more than they could individually. Individuals who program in pairs always have someone to discuss their problems with and seek advice from (Chao & Atli, 2006; Demir & Seferoğlu, 2020). Besides, pair programming has many advantages in terms of learning. For example, it improves and increases the program's quality and learners' coding skills through sharing and discussing information (Dongo, Reed & O'Hara, 2016). Pair programming also reduces anxiety of learners who are new to programming, brings a positive perspective towards programming, and creates a sense of achievement (Dongo et al., 2016). Later, Ghorashi and Jensen (2017) stated that it develops the efficiency in programming related to the time spent with coding and the quality of the work. According to Denner, Werner, Campe and Ortiz (2014), pair programming also enhances novice programmers' CT skills.

Various studies have been done to analyze the impacts of pair programming on learners' CT skills and their coding performance. For example, some previous studies (e.g. Braught et al., 2008; McDowell, Werner, Bullock & Fernald, 2002) indicated that student programmers who work collaboratively outperform students who program individually (Beck & Chizhik, 2013). McDowell et al. (2002) investigated impacts of pair programming on undergraduate students' performance in an introductory programming class. The results of the study demonstrated that participants with partners produced better-coded programs. Dyba, Dingsoyr and Hanssen (2007) conducted a meta-analysis on whether pair programming is advantageous over solo programming in terms of time, effort, and product quality. According to this meta-analysis, despite the additional time investment, pair programming was found to be more beneficial than solo programming in terms of task completion time and overall quality of programming. According to Denner et

al.’s (2014) study, pair programming was considerably more effective than programming independently in fostering CT and programming knowledge among middle school students, especially for students with less experience. In another study, Iskrenovic-Momcilovic (2019) examined an elementary Scratch programming session in a primary school and students worked with pairs programmed better than when working alone as a result of this study.

Prior studies have primarily examined the impact of pair programming on students' perspectives, motivation, and engagement, neglecting the investigation of how it influences the programming and CT skills of pre-service teachers (Emurian, Holden, & Abarbanel, 2008). These studies highlight the lack of consensus regarding how group dynamics can effectively coordinate multiple roles in collaborative problem solving, foster shared understanding through conflict resolution, overcome challenges in complex programming tasks, and facilitate the development of computational thinking (CT) skills (Teague & Lister, 2014). There has also been a significant increase in the literature on CT, but research about CT have concentrated more on K-12 education (Haseski, Ilic & Tugtekin, 2018). However, effective practice and activities should be designed by teachers to develop students' CT skills. To effectively develop students' CT skills, teachers should design practice exercises and activities that are targeted towards this goal. In order to address this need, programming and CT should be incorporated into teacher education curricula to ensure that prospective teachers are adequately prepared. Moreover, it is crucial to investigate the effectiveness of programming instruction within teacher education programs, particularly examining how pair programming is implemented and its efficacy as a method for pre-service teachers in higher education (Yadav et al.,

2014). Such studies would inform design decisions of instructors and instructional designers when offering a programming course.

1.1 Significance of the study

The use of pair programming as an instructional method can enhance the effectiveness of programming instruction for pre-service instructors by incorporating cooperative learning principles (Mentz, Van der Walt, & Goosen, 2008). This approach enables teacher candidates to cultivate their high-level skills in programming and CT, while also equipping them to teach these skills to their students from early grades onwards. However, there is a lack of research about programming of pre-service teachers and how they engage in CT when they complete programming tasks individually or collaboratively. In addition, Demir and Seferoğlu (2020) put forward that using pair programming more frequently in educational settings is very important, and experimental studies are needed to see the effects of pair programming. Limited number of studies (e.g., Grover & Pea, 2013; Lyon & Magana, 2020) have examined whether pair programming is effective in educational settings, especially in online environments. Also, as these studies were conducted at K-12 level, the scarcity of research with pre-service teachers creates a gap in the literature. Hence, exploring practical ways to apply CT skills in all classrooms including higher education across all majors and fields is needed (Lyon & Magana, 2020). By implying these gaps, this study investigated how pre-service teachers learned programming and how their CT skills and coding achievement effected when they worked individually and in pairs.

1.2 Purpose of the study

The purpose of this study is to examine the effect of pair programming on pre-service teachers' CT skills and coding achievement and to explore pre-service teachers' learning experiences of programming.

1.3 Research questions

Research questions of the study are given below:

Research Question 1: Is there a significant difference between the CT skills of pre-service teachers who are involved in pair programming and individual programming?

Research Question 2: Is there a significant difference between the coding achievement of pre-service teachers who are involved in pair programming and individual programming?

Research Question 3: What are learning experiences of pre-service teachers during individual and pair programming?

1.4 Research hypotheses

The research hypotheses of the research questions are given below:

Null Hypotheses:

H0: There is no significant difference between the CT skills of pre-service teachers who are involved in pair programming and individual programming.

H0: There is no significant difference between the coding achievement of pre-service teachers who are involved in pair programming and individual programming.

Alternative Hypotheses:

H1: There is a significant difference between the CT skills of pre-service teachers who are involved in pair programming and individual programming.

H1: There is a significant difference between the coding achievement of pre-service teachers who are involved in pair programming and individual programming.

1.5 Study organization

In the Chapter 2, a literature review of CT, CT components, programming, and pair programming was presented. Chapter 3 covers the research methodology; research design, the context and participants, instruments for data collection, data collection procedures, and data analysis. In the Chapter 4, the results of data analysis were shown. Lastly, in the Chapter 5, discussion of the findings, recommendations for the future research, and limitations of the study were presented.

CHAPTER 2

LITERATURE REVIEW

2.1 Computational thinking

2.1.1 Definitions of computational thinking

Papert (1996) is the first researcher who introduced the concept of CT as computational ideas, stating that information technologies and CT would provide new opportunities for learners in terms of learning and thinking skills and affective and cognitive development. Arguing that everyone should have CT skills, not just computer scientists, Wing (2006) defined CT as "solving problems, designing systems and understanding human behavior by drawing on the concepts of computer science." (p. 33). CT is considered as relevant to computer science as a field of application; however, it has started to be effective in all fields. In this respect, CT separates itself from other disciplines, creating a higher framework and presenting a new perspective (Wing, 2008).

Wing (2008) improved the definition of CT as a thinking process that includes formulations in order to find solutions to problems that can be performed efficiently by an information-processing unit. According to Aho (2012), CT is also a thinking process that determines computational procedures to solve problems and develop algorithms. CT, as an important part of high-level skills in 21st century classrooms, was also expressed as a total package of skill set which includes abstraction, algorithmic thinking, problem decomposition, and debugging (Turchi, Fogli & Malizia, 2019). ISTE and the Computer Science Teachers Association (CSTA) (2011) described CT as a technique for problem-solving that contains, but is not restricted to, several aspects, such as organizing and analyzing data in a logical

and sequential manner, using resources effectively, and identifying, assessing, implementing the most appropriate solutions and alternatives to the problem.

Although there is a common consensus on critical facts in the definition of CT, there is no precise definition, but it is well-defined across disciplines (Grover & Pea, 2013; Kalelioglu, Gülbahar & Kukul, 2016). In addition, there are varying views concerning the essential components of CT. For example, according to Csizmadia et al. (2015), CT involves cognitive thinking and logical reasoning for solving problems and better understanding products, systems, and processes. According to Kalelioglu et al. (2016), the three most utilized components in establishing CT are abstraction, algorithmic thinking and problem-solving. Many different concepts are discussed within CT, and the literature shows that these concepts differ considerably. Addressing this issue, Brennan and Resnick (2012) defined CT in three dimensions: 1) computational concepts, 2) computational practices, and 3) computational perspectives. Arguing that CT should be introduced to preservice teachers from different subject areas, Yadav et al. (2014) classified five CT principles: 1) problem identification and decomposition, 2) abstraction, 3) logical thinking, 4) algorithms, and 5) debugging. On the other hand, according to Wing's (2006) widely known description of CT and ISTE Standards (2015), problem-solving, algorithmic thinking, critical thinking, cooperative learning, and creative thinking are the main components of CT. These components of CT are described below.

2.1.2 CT skills

2.1.2.1 Problem solving

CT is a method for solving problems (Wing, 2006). Problem solving is the power to overcome the obstacles encountered in the process of achieving a goal. It involves the processes of reaching a goal, developing tools to reach that goal, and overcoming

obstacles (Koray & Azar, 2008). Problem-solving skills have a multidimensional structure since they can also include thinking and learning skills (Kotluk & Kocakaya, 2015). Problem solving is also a course during which a learner sees an inconsistency between the present position and the one that is purposed (Hesse, Care, Buder, Sassenberg & Griffin, 2015). At the time the learner finds out that there is no direct solution to this disparity, he/she takes actions to achieve his/her goal, considering the given situation (Hesse et al., 2015).

CT can support problem solving with the help of several methods and strategies. Methods and strategies for problem solving may involve logically organizing data, breaking complex problems into smaller components, defining abstract ideas, and inventing and employing algorithms, patterns, and models (Selby, 2014). According to Yadav, Stephenson, and Hong (2017), students need to have the opportunity to create and implement effective problem-solving strategies using their CT skills.

CT enhances development of reasoning abilities for problem solving through the use of computer science principles, algorithms, and programming techniques (Psycharis & Kallia, 2017). Programming is a process that encompasses various methods for devising and building solutions (Saeli, Perrenet, Wim, Joochems & Zwaneveld, 2011). It involves the capacity to generate solutions to problems through effective problem analysis (Saeli et al., 2011). For example, Shin and Park (2014) how the programming language tool Scratch impacted students' problem-solving skills. 46 middle school students were participated in this study. The results showed that using Scratch programming increased problem-solving ability of participants. Shin and Park (2014) suggested that Scratch programming is a useful tool in today's

classrooms since it allows for easy articulation of ideas, which is not possible with other programming languages.

In this study, pair programming method was applied by using Scratch. How participants' problem-solving, which is one of the CT skills, developed was examined.

2.1.2.2 Algorithmic thinking

CT also includes algorithmic thinking (Denning, 2009). Algorithmic thinking is defined as the process of putting individuals' actions in order (Ross, 1998). The concept of algorithm involves defining the phases to proceed while solving a problem and the solving process itself. These two concepts are included in algorithmic thinking. (Kiss & Arki, 2017).

Computing follows an algorithmic framework in which an input is accepted and systematically processed for giving an outcome. Therefore, one of the key skills in CT is algorithmic thinking (Doleck, Bazelais, Lemay, Saxena & Basnet, 2017). Indeed, Yadav et al. (2017) asserted that "algorithms are central to both computer science and computational thinking. Algorithms underlie the most basic tasks everyone engages in, from following a simple cooking recipe to providing complicated driving directions" (p. 57). According to Wing (2008), an algorithm can be described as a process which includes forming a basic way to the purpose at the end via a step-by-step operation.

During programming, learners also need to think logically and be involved in algorithmic thinking (Kiss & Arki, 2017). Kiss and Arki (2017) found that algorithmic thinking is essential for undergraduate and graduate students, those who do not have a it beforehand are at a disadvantage, and they suggested that traditional

teaching processes were ineffective in establishing the conceptual framework necessary for coding and problem solving. Thus, they used a game-based programming method. As a result, they emphasized the importance of programming on algorithmic thinking and CT in education. In another study, Fanchamps, Slangen, Hennissen and Specht (2019) examined how employing programmable robots in various instructional approaches helps 62 students improve algorithmic thinking and self-efficacy. The participants were primary school students. They used Lego NXT robots and Mindstorm software, and the participants of this study solved programming problems as pairs. Results of the study demonstrated that the participants enhanced their algorithmic thinking abilities when using programming robots in pairs.

In this study, the impact of pair programming on pre-service teachers' algorithmic thinking skills as a component of CT was examined.

2.1.2.3 Critical thinking

Doleck et al. (2017) asserted that critical thinking is another CT skill. For a long time, critical thinking has been taken into consideration among significant life competences for achieving success, and as a result, it has received a lot of attention in educational settings. Since critical thinking involves more complicated thinking and frequently results in innovative solutions, it may enable learners to gain new knowledge during problem solving (Doleck et al., 2017). Critical thinking is important because it involves active engagement with and evaluation of interpretations, concepts, and conclusions in order to form new knowledge (Doleck et al., 2017).

Bailey and Mentz (2017) stated that critical thinking and computer programming go hand in hand. Learners who can think critically are able to distinguish between correct and incorrect information, and well-written and buggy codes. In the study of Bailey and Mentz (2017), Information Technology students and teachers were given the task of implementing pair programming in the study and reporting back on their experiences. Lecturers highlighted that students' working in pairs in programming allows them to help each other and share their knowledge. In this study, students dealt with the difficulties associated with programming tasks. So, they appreciated working with a pair. As a result of the study, Bailey and Mentz (2017) found that pair programming enhanced their critical thinking.

In this study, pre-service teachers' critical thinking skills during pair and solo programming were examined.

2.1.2.4 Creativity

Another component of CT is creativity (DeSchryver & Yadav, 2015) because when using CT strategies to solve problems, a certain amount of creativity is required to come up with solutions (Snalune, 2015). Creativity can be defined as being willing to share and convey an idea from one form to another one which is not known before by using imagination, intuition, intelligence, feeling, and emotion (Dellas & Gaier, 1970). Finding, evaluating, and solving issues from alternative viewpoints that others do not recognize are examples of creative thinking skills (Hensley, 2020).

Creativity has been also considered as an essential part of CT linked to programming abilities (Korkmaz, Çakir & Özden, 2017). Creativity is important in programming because it encourages learners to think of new ways to solve issues (Clements, 1995; Grover & Pea, 2013). Also, learning programming impacts

creativity positively (Kim, Chung & Yu, 2013; Scherer, Siddiq & Sánchez Viveros, 2019). For example, Çakır, Korkmaz, İdil and Erdoğmuş (2021) contucted a study about the effects of robotics and programming instruction on 40 preschool students' creativity. The experimental group used the WeDo 2.0 Educational Robotics Kit during the intervention, whereas the control group used pen-and-paper activities. Çakır et al. (2021) found that being involved in robotics and coding activities had a significantly positive effect on the participants' creative thinking abilities. In another study, Seo and Kim (2016) investigated the impact of coding education on CT skills and creativity using pair programming. Seo and Kim (2016) used pair programming with the experimental group, and individual programming with the control group. 44 elementary students were participated in the study, and geometry lesson was the subject of coding education in the study. They found that CT skills of the two groups did not differ significantly at the end of the study. However, when the findings were compared within each group separately, they found that the experimental group significantly increased their creativity, and the control group did not increase their creativity.

In this study, how pre-service teachers' creativity changed when they were involved in pair or solo programming was examined.

2.1.2.5 Cooperative learning

In CT, cooperation emerges as an effective strategy (Csizmadia, Standl & Waite, 2019). Working cooperatively becomes more important as the complexity of a problem increases. A teaching strategy known as cooperative learning involves having students work in small groups and rewarding them according to how well

they accomplish (Slavin, 1980). Cooperative learning is also regarded as an indispensable component of CT (Yağcı, 2019).

Cooperative learning is related to learning programming. Learners, especially beginners and those with little or no programming experience, usually find it difficult to learn programming (Hwang, Shadiev, Wang & Huang, 2012). Computer labs were often used for traditional programming lessons with a teacher speaking and discussing syntax, logics, ideas, and code analysis (Slavin, 1980). This instructional approach may decrease the efficacy of learning, as it offers limited chances for students to enhance their programming abilities, and such learning environments might not be appropriate for each student (Hwang et al., 2012). It was proposed by McDowell et al. (2002) that collaborative learning activities are engaged in instructional design for programming education. According to Chiu (2008), cooperation has numerous potential benefits in learning programming because learners take advantage of one another's abilities by asking for information, examining opinions that others have, and supervising one another's work. In the study of Hwang et al. (2012), they examined the students' cooperative learning of programming and how it relates to their academic success. They used a web-based programming assisted system for cooperation (WPASC) and implemented a cooperative learning exercise for students to learn programming. They found that most of the participants found the activity and use of WPASC helpful for their learning of programming.

In this study, the experimental group was involved in pair programming. And, how pair programming affects their coding achievement and CT skills was examined.

2.2 Programming

The students can use CT through programming which includes problem solving by applying computer science principles (Lye & Koh, 2014). Programming can be used effectively to enhance CT skills of learners (Kazimoglu, Kiernan, Bacon & MacKinnon, 2012; Qualls & Sherrell, 2010). With its most general definition, programming is the sum of commands sent to a computer (Lye & Koh, 2014). In other words, programming is a process of building a solution for a problem in virtual setting with the help of a programming language (Lye & Koh, 2014). Coding in a programming language properly consists of understanding and analyzing problems, converting solutions into algorithms, and establishing the right algorithm (Kazimoğlu et al., 2012). Programming is also an important part of education for students to gain 21st century skills (Çoklar & Akçay, 2018).

Efforts are being made across the world to integrate CT into the classrooms for 21st century (Buitrago Flórez et al., 2017). Tikva and Tambouris (2021) stated that CT has started to appear on syllabi as a required or elective course in K-12 education. In addition, computer science courses are also included in education programs of teachers to help prospective teachers examine how CT connects to their fields of study (Yadav et al., 2014). However, the curriculum in teacher education is insufficient in teaching CT through programming (Yadav et al., 2014). Also, previous studies (e.g., Denning, 2017; Grover & Pea, 2013) show that teachers possess insufficient knowledge on effectively incorporating CT into K-12 learning environments. The reason why teachers experience this deficiency is due to inadequacy of programming courses in teacher education curricula (Burden, Aubusson, Brindley & Schuck, 2016). In addition to revising teacher education curricula, new approaches in programming education are needed to overcome this

deficiency (Instefjord & Munthe, 2017). Programming instruction in schools is becoming prevalent, and accordingly, new strategies in programming are being developed (Demir & Seferoğlu, 2020). One of these strategies is pair programming.

2.2.1 Pair programming

It has been proposed that one of the beneficial techniques for helping the students in developing their CT skills is collaborative learning (Hanks, 2008). According to Zhong, Wang, and Chen (2016), individuals who learn programming in a collaborative setting perform better in terms of making successful projects and developing CT than those who learn programming on their own. A form of cooperative learning called pair programming includes two people working together. Previous research (e.g., Wei et al., 2008) shows that one of the most effective pedagogical methods for teaching programming is through pair programming. Two programmers work together and share their knowledge in compliance with each other's needs on the same project simultaneously in pair programming: One writes code while the other observes, assists, and helps navigate complex challenges and errors (Gómez et al., 2017). According to Gómez et al. (2017), programmers who require assistance to improve their program can benefit from pair programming.

Pair programming also enhances learners' coding skills and program quality since it usually encourages effective communication between peers (Dongo et al., 2016). For example, Demir and Seferoğlu (2020) investigated effects of pair programming on coding quality and coding achievement. 42 undergraduate students involved the study. The experimental group in the study worked in pairs, whereas the control group in the study worked alone. After three weeks, coding quality of the participants who worked in pairs was higher than the coding quality of the

participants who worked individually. No differences were identified in the other weeks in terms of the code quality. Although both groups improved their coding achievement during the study, no differences were identified in terms of the coding achievement between the groups. Demir and Seferoğlu (2020) suggested that pair programming is an effective technique and should be used more often in educational contexts. In another study, McDowell et al. (2002) investigated how pair programming affected students' performance in an introductory programming class. The participants were 600 undergraduate students. The participants of the study worked on programming tasks with a pair or on their own. In the results of the study of McDowell et al. (2002), better programs were created by the the students who coded with partners.

Moreover, pair programming increases CT skills. Denner et al. (2014) investigated the impacts of pair programming. 320 middle school students were participated in the study. The participants worked solo or in pairs on a game programming task were participated in the study. It was found that pair programming was more helpful particularly for less-experienced learners to enhance their CT and programming skills. Wei et al. (2021) investigated the impacts of pair programming on students' CT skills and self-efficacy. 171 middle school students were participated in this study by learning from a course named "Computational Thinking with Scratch." For the programming tasks, the experimental group of the study were involved in pair programming, whereas the control group of the study were involved in individual programming. In the results of the study of Wei et al. (2021), the participants who worked in pairs developed their CT skills and their self-efficacy for programming more than participants who worked alone.

In addition, Khudhur (2022) examined the effectiveness of pair programming on the comprehensibility of flowcharts and the learning outcomes of first-year computer science students in a C programming course. The study employed an experimental design with two groups to examine the impacts of pair programming on defect detection performance when working on flowcharts. The study analyzed and compared the progress of both groups in the course based on pre-test and post-test experimental design. The results of the study showed that pair group detected defects on the flowchart more accurately than the solo group.

Pair programming can also be applied in online environments (Beasley & Johnson, 2022). Pair programming occurs online when two programmers work on the same project using software for screen sharing and communication (Li et al., 2021). There are also studies (e.g., Ayub, Karnalim, Risal & Wijanto, 2020; Ghorashi & Jensen, 2017) about pair programming that was implemented in online environments. For example, Beasley and Johnson (2022) investigated pair programming's impact on a higher-level computer science course. They used separate sections, which included the control group (individual programmers) and the experimental group (pair programmers). Pair programmers chose whether to work remotely or in person. They examined the participants' assignments and exams and compared the modality and gender. At the end of the study, they found pair programmers performed better and got higher scores on the tasks and exams. Paired groups who worked remotely had high scores as well as paired groups who worked in person. Also, female students felt more comfortable involving the class discussions and accomplished better in the course when using pair programming.

In the light of the above studies, it is obvious that pair programming is an effective method for programming instruction. As mentioned earlier, these studies

were limited to K-12 settings, and there is insufficient research on how pair programming affects teacher candidates' CT skills and coding. This study's purpose is to examine the effects of pair programming on pre-service teachers' CT skills and coding achievement and to explore pre-service teachers' learning of programming.

2.3 Using scratch

Numerous programming tools have been created especially for novice programmers. Scratch is an instructional programming setting created by the Massachusetts Institute of Technology (MIT) Media Lab's project (MIT Media Lab, 2008). Scratch is a visual programming language that makes programming accessible to children, teens, and other inexperienced programmers. Inexperienced programmers can learn programming easily through visual programming languages as they make programming less challenging through visuals (Maloney et al., 2004). According to Fagerlund, Häkkinen, Vesisenaho and Viiri (2021), visual programming also enables learners with little coding experience to develop complex programs.

It is mentioned by Maloney et al. (2004) that Scratch enables programmers to form multimedia products which combines graphics, animation, music, and sound. Besides, creating video games, animation and interactive tales are possible via Scratch (Ilic, 2021). In addition, various programming activities that learners use and are involved in can be developed on the Scratch platform. This programming tool also helps learners to develop their CT skills (Fagerlund et al., 2021).

According to the literature, the studies emphasized the importance of programming as a tool to develop CT skills, although it points out the insufficient implementation of programming in teacher education curricula and the need for new approaches such as pair programming. The latter technique, characterized by two

programmers simultaneously working on a single project, has been shown to foster successful projects and improve CT and programming skills, particularly when used by novice or less-experienced learners. Several studies, which have investigated the efficacy of pair programming in both physical and online environments, suggest that learners who participate in this collaborative environment tend to achieve better results than their solo-programming peers. However, previous studies mostly focus on K-12 settings, with limited research available on pre-service teachers, according to the literature. This study aims to investigate how pair programming influences pre-service teachers' CT skills and coding achievement, using Scratch, a visual programming language specifically designed for novice programmers, as the programming tool.

CHAPTER 3

METHOD

The aim of this study is to examine the effect of pair programming on pre-service teachers' CT skills and coding achievement and to explore the pre-service teachers' learning experiences of programming. This chapter involves the followings: (1) research design, (2) the context and participants, (3) instruments for data collection, (4) data collection procedures, and (5) data analysis.

3.1 Research design

This study employed mixed methods research design, using the explanatory sequential type (Creswell & Plano-Clark, 2018). That is, quantitative data collection was followed by qualitative data collection. This design provides a comprehensive understanding of the research problem through a two-phase approach (Greene, 2007). First, quantitative data is gathered and analyzed. This helps for objective measurements and a wide range of generalizations. After this quantitative phase, the study moves to the qualitative phase. In this phase, interviews or focus groups are used to understand better and interpret the results obtained from the quantitative phase. Examining the initial phase can uncover fundamental themes, patterns, or phenomena that offer a more profound understanding of numerical results. The Explanatory Sequential Mixed Method is a well-regarded research approach that provides a clearer and more nuanced understanding of complex issues. It is a valuable tool for producing meaningful insights.

The research questions of this study involved both quantitative and qualitative questions. While quantitative questions sought for the effect of pair

programming on pre-service teachers' coding achievement and computational thinking skills, qualitative ones explored how preservice teachers experience programming.

The study's independent variable was the mode of programming activities; that is, individual and pair programming. The study's dependent variables were pre-service teachers' coding achievement measured with Coding Achievement Test (Demir & Seferoğlu, 2019) and CT skills measured with Computational Thinking Skills (CTS) Scale (Yağcı, 2019).

3.2 Context and participants

The sample of this study were pre-service teachers taking a required technology integration course (CET 360) at Boğaziçi University. This course aims to enhance students' comprehension of the theoretical underpinnings of technology-enhanced instructional materials, while also providing hands-on experience in creating materials that can be utilized by pre-service teachers in their future teaching practices. The course had both lecture and lab sections that equipped pre-service teachers with the theoretical background and technological skills necessary for effective technology integration into their teaching. The data was collected from the lab sessions of this course. Lab sessions were conducted in online environment. Lab sessions lasted for 10 weeks.

The total of 36 pre-service teachers from different subject areas and who were not computer teachers participated in the study. Purposive sampling method was used since it was aimed to select the participants from pre-service teachers who did not take Scratch programming course before. At the beginning of the first lesson, the instructor informally asked the participants whether they had taken Scratch

lessons before, and it was determined that none of the participants knew how to use Scratch before. It was voluntary to take part in the study. Before the study started, students in the course were invited to the study, and their informed consents were taken (see Appendix A).

The course was offered in two sections. The sections were randomly assigned to control and experimental groups. Since all the participants met the criterion for purposive sampling, their data was included in the study if they agreed to be a participant. The participants' ages range was around 21-22. There were 6 male and 30 female pre-service teachers in the sample of the study. The study included a diverse range of pre-service teachers from different departments: 15 pre-service teachers studying at middle school mathematics teaching, 10 studying at science teaching, 6 studying at English teaching, 3 studying at physics teaching, 1 studying in the department of guidance and psychological counseling, and 1 studying at chemistry teaching (see Table 1).

Table 1. Demographic Information of Participants

	Number	Percentage
Gender		
Female	30	83
Male	6	17
Department		
Middle School Mathematics Teaching	15	41
Science Teaching	10	28
English Teaching	6	17
Physics Teaching	3	08
Guidance and Psychological Counseling	1	03
Chemistry Teaching	1	03

There were two groups in the study. One group was the control group who was involved in individual programming. The other group was the experimental group who was involved in pair programming. The course instructor determined the content of the course, which included block-based coding tasks on Scratch. Scratch, which is a block-based visual programming language, was taught to pre-service teachers as an educational tool for programming in the lab sessions of CET 360 course. The Scratch activities and tasks lasted about 2 hours each week. The instructor was the same in both groups during the study. During the first lab session, the experimental group's participants were informed how they needed to work as a pair. Participants working in pairs were matched according to the department they were in. Participants in the other class section were the control group. They were involved in programming activities by completing the same tasks individually.

3.3 Instruments for data collection

In this study, three instruments were used. These involved Coding Achievement Test (Demir & Seferoğlu, 2019) which was designed for testing participants' coding achievement, Computational Thinking Skills (CTS) Scale (Yağcı, 2019) which measured the CTS of participants. For collecting coding achievement results, which was quantitative data, the coding achievement test (Demir & Seferoğlu, 2019) was applied before the intervention started as a pre-test, and the same test was applied at the end of the semester as a post-test. Quantitative data was collected through CTS scale (Yağcı, 2019) as a pretest and a posttest in order to examine CT skills of the pre-service teachers. For collecting qualitative data, interviews were conducted to explore pre-service teachers' learning experiences during programming activities.

3.3.1 Coding achievement test

Quantitative data of the study was obtained by conducting the Coding Achievement Test as pre-test and post-test. The test developed by Demir and Seferoğlu (2019) aimed to determine participants' coding achievement. This coding achievement test was a multiple-choice test, consisting of 18 questions (see Appendix B). Each correct answer was scored with a positive value (1 point), and incorrect answers and no answers were scored with a value of zero. The sum of correct scores was the total test score. The test had a great internal consistency level, as determined by a Cronbach Alpha reliability coefficient of 0.801.

3.3.2 CTS scale

The other instrument to collect quantitative data in the study was CTS scale (Yağcı, 2019). The scale's goal was to determine participants' CT skills. The scale included five components: (a) problem-solving, (b) algorithmic thinking, (c) critical thinking, (d) cooperative learning, and (e) creative thinking. The CTS scale was a five-point Likert scale consisting of 42 items: (1) "Strongly disagree", (2) "Disagree", (3) "Neutral", (4) "Agree", and (5) "Strongly agree" (see Appendix C). The scale had a large internal consistency level, as determined by a Cronbach's Alpha reliability coefficient of 0.969.

3.3.3 Interviews

Qualitative data of the study was obtained through interviews. The interviews' purpose was to explore participants' learning experiences during programming within the study. Interview questions were structured by consisting of the components of the CTS scale, which were problem-solving, algorithmic thinking, critical thinking, cooperative learning, and creative thinking. Interviews were semi-

structured with 8 open-ended questions (see Appendix D), and each took about 15 minutes.

3.4 Data collection procedures

Before conducting the study, ethics approval was taken from the Ethics Committee in Social Sciences and Humanities (SOBETIK) of Boğaziçi University where the study was implemented (see Appendix E). Participants who were pre-service teachers were informed regarding the study, and the consents of the participants were taken before the study started (see Appendix A).

At the beginning of the semester, Coding Achievement Test (Demir & Seferoğlu, 2019) and CTS scale (Yağcı, 2019) were given as pretests to both control and experimental groups (see Appendices B and C). 60 minutes were given to participants to complete the test. And, participants completed the CTS scale in approximately 20 minutes. During the first-class session, the test and the scale were administered online. In the second-class session, participants in the experimental group were divided into groups, and how pair programming activities should be completed was explained to them. The experimental group completed Scratch tasks and activities in 20 class hours over 10 weeks by doing the tasks with their partners. Pairs were matched according to the department they were in. The tasks and the activities (see Table 2) were adapted from Getting Unstuck Curriculum at https://gettingunstuck.gse.harvard.edu/, a Scratch curriculum developed by the Creative Computing Lab at the Harvard Graduate School of Education (2021). The activities consisted of three parts named *Explore*, *Create*, and *Reflect*. Detailed descriptions of the tasks can be found in Appendix F.

Table 2. Scratch Tasks Completed During the Study

Week	Scratch Tasks	Duration to Complete
#1	When Clicked	2 hours
#2	Parallelism	2 hours
#3	Key Press	2 hours
#4	Loops	2 hours
#5	Broadcast	2 hours
#6	Color Sensing	2 hours
#7	Random	2 hours
#8	Ask and Answer	2 hours
#9	Variables	2 hours
#10	Lists	2 hours

The control group individually completed the same Scratch tasks and activities (see Table 2) in 20 class hours over 10 weeks. When the semester ended, the same coding achievement test and the CTS scale were applied to both groups as post-tests. After quantitative data collection and analysis, qualitative data was collected through individual semi-structured interviews. Four volunteering participants from each group were selected and interviewed. Interviews consisted of 8 open-ended questions (see Appendix D) and took about15 minutes. The researcher integrated the findings from quantitative (CTS Scale and Coding Achievement) and qualitative (semi structured interviews) datasets.

3.5 Data analysis

To answer the first and second research questions, quantitative analysis was conducted. The IBM SPSS statistical software (Statistical Package for Social Sciences - Version 20) was used. After the quantitative data were examined descriptively and normality check was completed, independent sample t-test was conducted since the data was normally distributed. However, when the normality of the CT skills of the groups was examined for the pre-test, the Mann-Whitney U test was used because the data did not show normal distribution. The goal of conducting

independent sample t-test and Mann-Whitney U test was to see the effects of pair programming on pre-service teachers' coding achievement and CT skills. A post-hoc power analysis was performed solely in cases where statistical significance was not detected (Thompson, 2008). Also, paired-samples t-test was applied to investigate if there was a difference between participants' pre-test and post-test scores of CT skills and coding achievement after they completed all the programming tasks throughout the semester. However, when the normality of the CT skills of the control group was examined for the pre-test, the Wilcoxon signed-rank was used because the data did not show normal distribution.

For qualitative data, qualitative thematic analyses were conducted through data coding on participants' responses to the interview questions. The researcher transcribed the audio recordings using a software program and then manually edited the transcription to eliminate any errors. Subsequently, the transcriptions were transferred from Microsoft Word to a qualitative analysis software. MAXQDA 2020 qualitative data analysis software was used. First, the whole dataset was examined through open-coding (Saldaña, 2021), and then focused coding (Saldaña, 2021) was implemented to identify categories, patterns, and themes. After both data analyses were completed, quantitative and qualitative findings were integrated.

3.6 Trustworthiness

Multiple approaches were implemented to ensure the reliability and credibility of this study. A mixed-method design was used to gather data from diverse sources, such as the Computational Thinking Skills Scale, Coding Achievement Test, and semi-structured interviews. This comprehensive methodology facilitated a deeper comprehension of the research subject and enhanced the validity and dependability

of the outcomes. By combining quantitative and qualitative data, the researcher synthesized the findings, identifying recurring themes and patterns.

For the sampling process, a purposive sampling technique was employed to carefully choose representative participants from various departments among pre-service teachers. To ensure a diverse range of perspectives, the interview participants were selected from three groups: those who demonstrated good, moderate, and bad coding success upon completing the course. This approach allowed the researcher to obtain comprehensive findings. To analyze the collected data from thesemi-structured interviews, the researcher worked with an expert who is well-versed in the field of Educational Technologies and in qualitative research designs. The expert reviewed all the codes and emerging themes and provided peer-debriefing, and the researcher revised and refined the codes and themes accordingly. This collaborative effort involved multiple stages of review and improvement, resulting in a refined a coding scheme.

Throughout the study, the researcher maintained an open and collaborative approach with the participants. To foster clear communication, explicit language was employed to prevent misunderstandings. Furthermore, the researcher ensured that complex ideas or terms were explained whenever necessary and provided participants with the required instructions.

CHAPTER 4

RESULTS

In this section, results of the study that aimed to investigate the effect of pair programming on pre-service teachers' CT skills and coding achievement were explained. Results in regard to each research questions and data analysis were presented in-depth using both descriptive and inferential statistics.

4.1 CT skills of pre-service teachers who are involved in pair programming and individual programming

CTS Scale (Yağcı, 2019) was used in order to measure and compare the experimental and the control group participants' CTS scores both before the study started in the beginning of the semester, and the same scale was used after the study ended at the end of the semester to investigate the Research Question 1 of the study. The scale was applied to a total of 36 teacher candidates, 16 from the control group and 20 from the experimental group, as pre-test and a post-test. Data from CTS Scale was inputted into Microsoft Excel and subsequently imported in SPSS. The descriptive statistics of CTS scores of pre-service teachers for pre-test and post-test results are shown in Table 3.

Table 3. Descriptive Statistics of the Experimental and the Control Groups for CT skills for Pre-test and Post-test

		N	Skewness	Kurtosis	Min	Max	Mean	Median	SD
Pre-test	Experimental	20	-.346	-.106	123	190	156.8	158.5	17.5
	Control	16	-1.476	4.029	101	183	155.62	156.5	18.88
Post-test	Experimental	20	.140	-.409	142	204	170.31	170	16.63
	Control	16	-.464	.764	130	192	166.56	166	15.59

Shapiro-Wilk test was performed to check whether the data for the CTS scale were normally distributed for the pre-test scores of the participants. The CT skills scores of the control group were not normally distributed ($p < .05$) according to the results of Shapiro-Wilk test (see Table 4).

Table 4. Shapiro-Wilk Result of the Pre-test for the Experimental Group and the Control Group for CT Skills

	Shapiro-Wilk		
	Statistic	df	Sig.
Experimental	.972	20	.787
Control	.879	16	.038

Since the data was not normally distributed for control group's pre-test scores, Mann-Whithey U test, which is one of the non-parametric tests, was conducted (see Table 5). The results indicated that there was no significant difference between CT skills of the experimental group and the control group in the beginning of the study ($p > .05$).

Table 5. Mann-Whitney U Test for Pre-test Scores of CT Skills of the Experimental Group and the Control Group

	Experimental (n=20)	Control (n=16)		
	Mean Rank	Mean Rank	Z-value	*p*
CTS	18.65	18.31	-.096	.924

Shapiro-Wilk test was also performed to check whether the data for the CTS scale were normally distributed for the post-test scores of the participants. The CT skills scores of the control group were normally distributed ($p > .05$) according to the results of Shapiro-Wilk test (see Table 6).

Table 6. Shapiro-Wilk Result of the Post-test for the Experimental Group and the Control Group for CT Skills

	Shapiro-Wilk		
	Statistic	df	Sig.
Experimental	.976	20	.866
Control	.964	16	.728

The CT skills of paired programmers (N=20) and individual programmers (N=16) in a technology integration course were compared using an independent-samples t-test because the post-test scores were normally distributed. The study's group statistics were reported in Table 7. After conducting the test, it was found that there was no significant difference between scores of preservice teachers who were involved in pair programming (*M*= 169.60, *SD*= 16.63) and individual programming (*M*= 166.56, *SD*= 15.59) in terms of CT skills, *t*(34)= .560, *p*= .579 (see Table 7).

Table 7. Independent Samples T-Test for CT Skills Scores of the Experimental Group and the Control Group as Post-test

		F	Sig.	*t*	*df*	Sig. (2-tailed)	Mean Difference
CT Skills	Equal variances assumed	.133	.718	.560	34	.579	3.03750
	Equal variances not assumed			.564	33.085	.577	3.03750

4.2 Coding achievement of pre-service teachers who are involved in pair programming and individual programming

To answer the Research Question 2 of the study, Coding Achievement Test (Demir & Seferoğlu, 2019) was conducted as a pre-test and a post-test in order to measure and compare the experimental and the control group participants's coding. The descriptive statistics of Coding Achievment Test scores of pre-service teachers for pre-test and post-test results are shown in Table 8.

Table 8. Descriptive Statistics of the Experimental and the Control Groups for Coding Achievement for pre-test and post-test

		N	Skewness	Kurtosis	Min	Max	Mean	Median	SD
Pre-test	Experimental	20	-.040	-.818	2	16	9.7	9.5	3.89
	Control	16	-.026	-.318	5	14	9.62	10	2.39
Post-test	Experimental	20	.015	-1.332	8	18	12.93	13.5	3.09
	Control	16	.217	-.561	8	18	12.25	12	2.95

Shapiro-Wilk test was performed to check whether the data for the coding achievement scores were normally distributed for the pre-test scores of the participants. The coding achievement of the control group and the experimental group was normally distributed ($p > .05$) according to the results of Shapiro-Wilk test (see Table 9).

Table 9. Shapiro-Wilk Result of the Pre-test for the Experimental and Control Groups for Coding Achievement

	Shapiro-Wilk		
	Statistic	df	Sig.
Experimental	.961	20	.556
Control	.979	16	.954

Since the data was normally distributed, independent samples t-test was conducted (see Table 10). The results indicated that there was no significant difference between coding achievement of the experimental group and the control group in the beginning of the study ($p > .05$).

Table 10. Independent Samples T-Test for Coding Achievement Scores of the Experimental Group and the Control Group as Pre-test

		F	Sig.	*t*	*df*	Sig. (2-tailed)	Mean Difference
Coding achievement	Equal variances assumed	4.486	.042	.067	34	.947	.075
	Equal variances not assumed			.071	32.096	.944	.075

Shapiro-Wilk test was also performed to check whether the data for the coding achievement were normally distributed for the post-test scores of the participants. The coding achievement test scores of the control group were normally distributed ($p > .05$) according to the results of Shapiro-Wilk test (see Table 11).

Table 11. Shapiro-Wilk Result of the Post-test for the Experimental and Control Groups' Coding Achievement

	Shapiro-Wilk		
	Statistic	df	Sig.
Experimental	.151	20	.274
Control	.154	16	.502

As the data were normally distributed, independent samples t-test was conducted to determine if there was a significant difference between the post-test scores of the experimental and control groups in terms of coding achievement. The results indicated that there was no significant difference between the scores of preservice teachers involved in pair programming (M= 13.15, SD= 3.09) and individual programming (M= 12.25, SD= 2.95) in terms of coding achievement, $t(34) = .665$, $p = .383$ (see Table 12).

Table 12. Independent Samples T-Test for Coding Achievement Test Scores of the Experimental Group and the Control Group as Post-test

		F	Sig.	*t*	*df*	Sig. (2-tailed)	Mean Difference
Coding Achievement Test Scores	Equal variances assumed	.191	.665	.884	34	.383	.90000
	Equal variances not assumes			.888	32.902	.381	.90000

A post-hoc power analysis was applied. Based on the post-hoc power analysis, it was estimated that the achieved power for scores was 21% for the Coding Achievement Test and 13% for CTS measures. These results suggest that a larger sample size is necessary to obtain statistical significance.

4.3 Within-group comparisons of pre-service teachers' CT skills and coding achievement

In order to assess the pre-test and post-test scores of pre-service teachers' CT skills within the groups, the normality of the data was examined to determine the appropriate statistical test. A Shapiro-Wilk's test showed that the pre-test and post-test scores of the experimental group were normally distributed ($p > .05$) but the pre-test scores of the control group were not (see Table 13).

Table 13. Shapiro-Wilk Result of the Pre-test and Post-test of CT Skills Scores of Pre-service Teachers

		Shapiro-Wilk		
		Statistic	df	Sig.
Pre-test	Experimental	.120	20	.787
	Control	.195	16	.038
Post-test	Experimental	.112	20	.866
	Control	.114	16	.728

A Paired Sample t-test was applied for the experimental group's pre-test and pos-test scores. The comparison between the pretest and posttest of the experimental group demonstrates the effect of the programming instruction on pre-service teachers' CT skills scores. The results of the experimental group ($t(19)= 5.109$, $p= .000$) showed that programming instruction resulted in statistically significant improvement in the post-test scores compared to the pre-test scores of the experimental group, with a strong effect size ($d= 1.12$) (see Table 14).

Table 14. Paired Samples Test for Post-test and Pre-test of CT Skills Scores of the Experimental Group

		Mean	Std. Deviation	*t*	*df*	Sig. (2-tailed)	Cohen's d
Experimental Group	Pretest-posttest	12.8	11.20432	5.109	19	.000	1.120

A Wilcoxon signed-rank test was applied for the control group's pre-test and pos-test scores. The comparison between the pretest and posttest scores of the control group demonstrates the effect of the programming instruction on pre-service teachers' CT skills scores. The results of the Wilcoxon signed-rank test ($z= -2.743$, $p= .006$) showed that programming instruction also resulted in statistically significant improvement in the post-test scores compared to the pre-test scores of the control group, with a moderate effect size ($d= .591$) (see Table 15).

Table 15. Wilcoxon Signed Rank Test for Post-test and Pre-test of CT Skills Scores of the Control Group

		N	Mean Rank	Sum of Ranks
Pre-test – Post-test	Negative Ranks	2[a]	7.50	15.00
	Positive Ranks	14[b]	8.64	121.00
	Ties	0[c]		
	Total	16		

a. Post-test < Pre-test; b. Post-test > Pre-test; Post-test = Pre-test

In order to assess the pre-test and post-test scores of pre-service teachers' coding achievement scores in both groups, the data's normality was checked to determine the appropriate statistical test. Shapiro-Wilk's test analysis revealed that the pre-test and post-test results for the pre-service teachers were normally distributed (p >.05) (see Table 16).

Table 16. Shapiro-Wilk Result of the Pre-test and Post-test Coding Achievement Scores of Pre-service Teachers

		Shapiro-Wilk		
		Statistic	df	Sig.
Pre-test	Experimental	.129	20	.556
	Control	.127	16	.954
Post-test	Experimental	.112	20	.314
	Control	.114	16	.502

A Paired Sample t-test was applied in each group. The comparison between the pretest and posttest in each group demonstrates the effect of the programming instruction on pre-service teachers' coding achievement scores. The results of the experimental group ($t(19)= 5.548$, $p= .000$) showed that programming instruction resulted in statistically significant improvement in the post-test scores compared to the pre-test scores, with a very strong effect size ($d= 1.216$) (see Table 17). The results of the control group ($t(15)= 3.918$, $p= .001$) showed that programming instruction also resulted in statistically significant improvement in the post-test scores compared to the pre-test scores, with a strong effect size ($d= .955$) (see Table 17). Although there was no significant difference found between the coding achievement scores of pre-service teachers, it can be seen that providing programming instruction for pre-service teachers was helpful for both groups.

Table 17. Paired Samples Test for Post-test and Pre-test of Coding Achievement Scores of Pre-service Teachers

Groups		Mean	Std. Deviation	*t*	*df*	Sig. (2-tailed)	Cohen's d
Experimental Group	Pretest-posttest	3.45	2.78104	5.548	19	.000	1.216
Control Group	Pretest-posttest	2.625	2.68017	3.918	15	.001	.955

4.4 Exploring the pre-service teachers' learning experiences of programming

To explore learning experiences of pre-service teachers and to answer the Research Question 3 of the study, 8 semi-structured interviews were conducted at the end of the semester. After analyzing these interviews, which was conducted with 4 participants from the experimental group (pair programming group) and 4 participants from the control group (individual programming group), six themes emerged (see Table 18).

Table 18. Emerged Themes

Themes	
Theme 1	The exchange of ideas during the activities enhances the quality of programming projects
Theme 2	Working with a pair during programming strengthens the communication skills of pre-service teachers
Theme 3	Preservice teachers who work in pairs detect the mistakes they make while coding earlier than preservice teachers who work individually do
Theme 4	When a problem arises during pair programming, another coder having a different perspective helps to identify the problem instantly
Theme 5	Working within a pair helps boost coding achievement
Theme 6	Story-based programming tasks boost novice coders' creativity

4.4.1 The role of exchanging ideas during programming

Pre-service teachers who worked in pairs completed all the activities in the lab sessions as pairs all the time. At the end of these sessions, they also created a project

each week. The participants who created the projects in pairs stated that they were in constant communication during the activities because they had to think and work on these projects together. As a result, there was a continuous exchange of ideas. These participants also claimed that the quality of their projects was high thanks to these exchanges of ideas. That is, the participants created the projects by taking their partner's opinions from different perspectives into consideration and constantly updating them during the session. Below are the quotes that illustrate how these projects integrated different perspectives of partners:

> The exchange of ideas in these activities can take our projects much further than I initially imagined. Um... If I have an idea, the additions by my pair have made the project even more fun, visually beautiful or more efficient. (Experimental Group\Participant3) (see Appendix G, Q1)

> I think pair programming has been very efficient because it's hard to come up with a new idea every week when I'm working alone. My pair and I had a consensus on what should or should not be a part of projects. We were evaluating each other's ideas. So, I think pair programming is more efficient. (Experimental Group\Participant2) (see Appendix G, Q2)

Moreover, the activities consisted of three parts called *Explore*, *Create*, and *Reflect*. In the explore part of these activities, students doing pair programming examined the given projects together. In the create part, they created a project together. While designing these projects, it was necessary to go through the stories and reveal a new idea. Participants who completed these projects in pairs believed that they had less difficulty creating them as they constantly exchanged ideas. They also felt that it was easier to create projects together as they shared ideas during the sessions. The following is an example that demonstrates how less challenging completing the projects in pairs was:

> I did [the activities] with my pair. It was easier for me to do projects this way. Because we were talking about how to do these projects in the lab sessions. I think it would be difficult if I did it alone. It was much easier for me and my pair to think about how to produce a project by exchanging ideas, to talk

about the project, to create a story. (Experimental Group\Participant1) (see Appendix G, Q3)

4.4.2 Enhancing pre-service teachers' communication skills through pair programming

The participants doing pair programming were divided into pairs on an online platform during the coding activities and completed the activities together during the session. During pair programming, one shared a screen while the other talked and communicated with her, telling her where to place the codes. They kept doing this until the classes finished, and each week a different partner shared their screen. During these sessions, there was interaction and communication between them. And with each passing week, it seemed that their communication got stronger, and they started to understand each other better. The participants recognized that being two people in programming strengthened their communication skills. The following quotes are examples of how they improved their communication during programming:

> I don't usually like group work, but I think I liked group work in this course because I shared the burden of the project with my pair, we exchanged ideas in every session, and the burden of work was not left to a single person. For me, it was easier to work in a group because we kept in constant communication. One of us was sharing screens, one of us was controlling coding. We were doing the tasks by dividing them up. That's why our communication was very good with my pair. (Experimental Group\Participant1) (see Appendix G, Q4)

> The perk of pair programming is that it brings a new perspective. And I also had the opportunity to improve my communication skills. (Experimental Group\Participant3) (see Appendix G, Q5)

Moreover, the pre-service teachers who communicated effectively among themselves stated that this situation was reflected in their coding. They realized that they were open to new ideas and wanted to create a good project through effective

communication. They pointed out that good communication increased their motivation to work on coding activities, as it is illustrated in the following quote from Participant 3.

> My partner and I established good communication. We have always been able to discuss projects together. Um… she was also very open to new ideas, and I always wanted to listen to her ideas. In a limited time, we tried to create something sound by contributing to each other's ideas. Overall, I think it was nice and effective. (Experimental Group\Participant3) (see Appendix G, Q6)

4.4.3 Pair programming accelerates error detection for pre-service teachers

The participants working in pairs learned programming and created a product together by sharing screens during lab lessons and activities. They rotated screen-sharing each week, and while one was coding, the other was monitoring and controlling these codes. They realized that this enabled them to detect their mistakes early during coding. This also helped them to debug the codes effectively. The participants highlighted this situation during the interviews and mentioned that working in pairs enabled them to recognize their mistakes more quickly compared to working alone. For example, Participant 2 and Participant 8 stated:

> For example, it can be difficult to notice when there is a mistake when [coding]. If something goes wrong with Scratch, you have to start over. But it's much easier to spot the error when one person is sharing a screen and the other is controlling it while [coding]. (Experimental Group\Participant2) (see Appendix G, Q7)

> We are constantly trying to produce something, trying to find the error and trying to fix that error while coding. I think this is mind-opening. Whenever I work individually, I cannot clearly see the mistakes I have made. Since I can't ask anyone else -- we also consult you -- but it is more beneficial for us to discuss and benefit from our peers more. Sometimes I cannot see the problem because I look at where there might be a problem as a single person. (Control Group\Participant8) (see Appendix G, Q8)

4.4.4 Instant problem-solving in pair programming through different perspectives

All of the participants in the study took a course on block-based programming for the first time. This is why they ran into so many problems when learning the code blocks and developing a project. For example, Participant 4 articulated: “I had a lot of problems [during coding activities]. For example, the backdrop I wanted to move didn't start the way I wanted it to, and it didn't move back to its original position. There was an absurd situation in the image, and I could not solve it in any way.” (Experimental Group\Participant4) (see Appendix G, Q9). While solving such problems, participants in both groups claimed that the problem should be looked at from a different perspective. Even if the participants who do individual programming stated that when they encounter a problem, another point of view towards that problem would be more effective while solving the problem. Because having a different point of view at that time can bring a solution to the problem from another perspective, and it might help detect where there is a problem. Below are the quotes, respectively, by participants working individually and working in pairs on how a different perspective helps identify a problem:

> When I work on these codes individually, I find it difficult. As I mentioned earlier, I think it's better to ask someone else and get a different perspective. But when we work individually, we have to change our perspective ourselves. (Control Group\Participant7) (see Appendix G, Q10)

> For finding a solution, it’s always better to have someone with you from whom you can get ideas. (Experimental Group\Participant3) (see Appendix G, Q11)

> As I said, I sometimes have difficulty determining where the problem comes from when I'm alone. Uh, because some of the codes are too complicated. We can't see exactly where the error is. Having a different perspective makes my task easier. Most of the time, we have solved [the problem] because there were two of us. (Experimental Group\Participant3) (see Appendix G, Q12)

The participants emphasized the significance of pair programming in dealing with problems as they arose. They expressed how important it was to identify issues promptly through this technique. Furthermore, the participants also argued that employing diverse perspectives provided swift problem-solving. The following quote illustrates the importance of recognizing problems quickly with the help of pair programming:

> In my opinion, the biggest advantage of pair programming is that when you encounter a problem and if you cannot find an answer, a second person can approach that problem differently. That way, you can find a quicker solution to that problem. (Experimental Group\Participant4) (see Appendix G, Q13)

4.4.5 Working within a pair helps boost coding achievement

Participants practiced how code blocks work by generating new ideas and creating projects while completing the coding activities every week. Regardless of their programming mode (solo vs. pair), all of the participants engaged in a learning process as novice learners. The participants who worked in pairs indicated that they completed this learning process by getting a lot of help from their peers. Therefore, the participants pointed out that it was an advantage for them to work with a pair to be successful in this course. It seemed that the exchange of ideas between participants, that is, working with a pair, increased their success in activities and projects. Furthermore, when there was a situation where a participant did not understand code blocks or activities, the help of their partners enabled them to increase his/her achievement. The following quotes illustrate this:

> [Pair programming] definitely increased my coding achievement even more. Because, as I said, I encounter problems. For example, I can't think of an idea and I don't know where to start, or I want to add a feature [to the project] at some point. But my partner gives me an idea of how to do it. So, it definitely increased my achievement. I feel like I could do things I couldn’t do alone. (Experimental Group\Participant3) (see Appendix G, Q14)

> When I was with my pair and we were brainstorming, I couldn't come up with ideas she had. And I was convinced that we could code this way. It has affected my coding achievement in an effective way. The most important thing for me was the exchange of ideas in group work. (Experimental Group\Participant1) (see Appendix G, Q15)

In addition, participants had a chance to learn from each other when they discussed their work. In this way, they advanced their achievement. They appreciated when their partners influenced their learning process positively. The participants also mentioned that seeing their development in their projects increased their motivation. The following are the examples that demonstrate how they mentioned their appreciation toward each other: "I think that working in pairs also increases my coding achievement. I learned a lot about coding from my pair." (Experimental Group\Participant2) (see Appendix G, Q16)

> I think that working with a pair had a positive effect on my coding success, because at the very beginning of the semester I was afraid of coding. [Coding] was something I never really knew. There was a lot of development between the last project we did in the labs and the first project we did. We observed the development of our projects every week we did. Seeing them both increased my motivation and made me happy. (Experimental Group\Participant4) (see Appendix G, Q17)

4.4.6 Boosting novice coders' creativity through story-based programming tasks

A relevant task was completed after a code block was explored in the lab sessions every week. Participants were asked to base these projects on a story or fiction rather than simply to use them to demonstrate that these codes were practiced. That is, each week meant a new story and fiction alongside a new code. There was evidence in the data showing that this allowed the participants in both groups to express their creativity and helped them develop their creativity each week. They pointed out that their creativity was boosted thanks to such programming tasks. The following are some representative quotes that illustrate this:

> I think I've used my creativity more than ever, because every week we tried to do a new project with a new concept, albeit at a simple level. When I looked at each project, my classmates and I all did different projects. It shows our creativity. (Control Group\Participant5) (see Appendix G, Q18)

> My pair and I were constantly thinking about how to transfer ideas from our minds to Scratch using this code block, and we were constantly trying to create a new story. So, it wasn't enough just to transfer the code because we had to embed a meaningful story in that code. In fact, constantly using our creativity was an adventure. (Experimental Group\Participant2) (see Appendix G, Q19)

4.5 Integration of findings

This study aimed to examine the effect of pair programming on pre-service teachers' computational thinking skills and coding achievement and to explore pre-service teachers' learning experiences during coding. For analyzing this effect, quantitative analysis was performed, and the difference in coding achievement between the two groups was examined. In the results, no significant difference was found between the two groups in terms of coding achievement. However, the post-hoc power analysis showed that this study should have a larger sample group for this difference to occur. Although there was no significant difference between the two groups, the mean of coding achievement scores of the experimental group was higher than the mean of coding achievement of the control group. According to the results of the qualitative analysis, all participants who worked in pairs stated during the interviews that working with a partner while coding boosted their coding success and motivation. They mentioned that their project discussions and ideas had a big influence on their work. This means the conversations and ideas they shared while developing their projects influenced their coding achievement scores. The participants highlighted that discussing ideas during activities helped improve project quality. They also

mentioned that working with someone else, especially in a new field like coding, motivated them and contributed to their success.

CT of participants in both groups was examined with the CTS as a quantitative analysis between the two groups. Problem-solving, algorithmic thinking, critical thinking, creativity, and cooperative learning are the CT components included in this scale, and they were examined within the scope of this study. In the results, no significant difference was found between the two groups. However, again, the post-hoc power analysis showed that this study should have a larger sample group for this difference to occur. Although there was no significant difference between the two groups, the mean of the experimental group's CT skills was higher than the mean of the control group's CT skills. According to the qualitative analysis' results, both groups in this study stated that their CT skills developed. Nonetheless, the participants who worked in pairs mentioned a greater emphasis on enhancement of their problem-solving skills. For example, the participants working in pairs said that when a problem was encountered by two people while programming, working in pairs was a great advantage to both determine and solve that problem. The involvement of a second coder with a different perspective during pair programming aids in the prompt identification of any arising issues. As a result, pre-service teachers who engaged in pair programming were able to detect their coding errors earlier than those who worked individually.

Upon analyzing the difference between the pre-test and post-test scores of the two groups, a significant improvement was observed in both coding achievement and CT skills. This highlights that the programming instruction is importance for teacher candidates. According to the findings, the use of story-based tasks led to an enhancement in the creativity skills of both groups. In the qualitative findings, the

participants in both groups highlighted that incorporating story-based programming into their projects boosted their creativity.

While the quantitative analysis did not reveal any significant difference between the two groups, the interviews conducted with the participants from both groups provided strong evidence regarding the significance of pair programming. The participants highlighted that pair programming is important in terms of improving coding achievement and developing problem-solving skills. They also emphasized that working in pairs increased their motivation and encouraged them to think more about their projects. In addition, it was observed that there is a difference between the pre-test and post-test scores underlined the importance of providing programming instruction to pre-service teachers who are novice learners.

CHAPTER 5

DISCUSSION AND CONCLUSION

The current study examined the effects of pair programming on coding achievement and CT skills of pre-service teachers. Quantitative data were collected via CTS Scale and Coding Achievement Test. Qualitative data were obtained through semi-structured interviews with eight participants. Integrating findings from quantitative and qualitative data provided a comprehensive understanding of the effects of pair programming on pre-service teachers' coding achievement and CT skills and their learning experiences of programming. This section discusses the data results in relation to the pertinent literature and presents the potential implications of the findings. Additionally, the chapter concludes by discussing recommendations for future research and acknowledging its limitations.

The first research question of the study investigated whether there was a significant difference between the CT skills of pre-service teachers involved in pair programming and individual programming. In the results, no significant difference was observed between the two groups' CT skills. However, the mean of the experimental group's CT skills scale scores was higher than the control group. This result conflicts with an earlier study, which showed that pair programming enhanced CT skills (Denner et al., 2014; Wei et al., 2021). This result in the current study might be the consequence of several factors.

One possible reason might be that learners are exposed to the fundamentals of CT while learning programming, which involves dividing difficult tasks into smaller, more doable ones, developing algorithms to solve these tasks, and applying logic and critical thinking to create effective solutions (Lye & Koh, 2014). The deep

relationship between programming and CT, emphasizing that engaging students in programming activities can significantly strengthen their CT skills (Yadav et al., 2017). Several research studies showed the benefits of programming instruction since it quite improves students' CT skills (Kazimoglu et al., 2012; Qualls & Sherrell, 2010). The finding obtained in this study is in line with the study of Seo and Kim (2016). Seo and Kim (2016) found no difference between the control (individual programming) and the experimental (pair programming) groups; however, they found the experimental group increased their CT skills when they examined the difference between the pre-test and post-test scores of participants. In this study, when the difference between the pre-test and post-test results of the CT skills of both groups was examined separately, it was found that the CT skills of the two groups increased. Yadav et al. (2014) suggested that coding education be in the curricula in teacher education to improve teacher candidates' CT skills. Although it was found in the present study that pair programming is not more effective than solo programming, participants in both groups in this study improved their CT skills. That is, it is possible that providing programming training to pre-service teachers boosted their CT skills, regardless of how they received the training.

Another possible reason of the results might be the programming language used in the study. Peppler and Kafai (2005) asserted that scratch tool helps students to gain skills including teamwork, creativity, problem-solving, systematic analysis, and communication which are components of CT skills. Additionally, the usage of Scratch, a visual programming language with blocks that are geared for beginning programmers, may have been very beneficial even for novice programmers. As a result, they could advance their CT skills on their own without any collaboration (Shin & Park, 2014). This is further supported by studies conducted by Ilic (2021),

which showed a significant increase in CT skills of teacher candidates who used Scratch-assisted expressions and applications. Moreover, the research of Zhang and Nouri (2019) that included 55 research articles shows the improvement of CT skills via programming Scratch. In agreement with the existing literature, both groups in this study improved their CT skills, and using Scratch might be the reason for this.

Another reason might be group dynamics in pair programming. There is no consensus on how group dynamics enhance the development of CT abilities, manage the challenges of authentic programming tasks, shared understanding through conflict resolution (Teague & Lister, 2014). The fact that the pairs worked in an online environment may also have affected the dynamics and interaction between them. As a large body of research (e.g., Ambler & Lines, 2012; Stray & Klotins, 2021) indicated, applying collaborative techniques is more difficult in virtual environments. There are challenges with using digital communication tools to close physical gaps, such as lowered communication quality caused by unstable networks and the potential for misinterpretation of meaning, tone, and emotions in digital interactions (Rizvi, Bagheri & Gasevic, 2015). These challenges of pair programming in online environments might have resulted in a non-significant difference between participants' CT skills and coding achievement in the control and experimental groups in the current study. However, in the qualitative analysis, the pre-service teachers who worked in pairs reported strong communication and interaction among themselves, which in turn fostered a positive dynamic. The interaction may be attributed to the fact that teacher candidates from the same department were paired together. In addition, there are research studies that showed the benefits of pair programming in online settings (e.g., Li et al, 2021; Da Silva Estácio & Prikladnicki, 2015). For example, pair programming in online settings can

also help to increase productivity, code quality, and enhance knowledge exchange while also fostering communication among team members who are apart from each other (Da Silva Estácio & Prikladnicki, 2015). The participants who completed programming tasks in pairs mentioned that they established a solid communication with their partners throughout the process in the qualitative analysis of this study. The interaction, dialogues, and exchange of information between the pairs strengthened their communication skills.

The second research question focused on whether there was a significant difference in coding achievement between pre-service teachers involved in pair programming and individual programming. In the results, no significant difference was observed between the two groups. However, the mean of the experimental group's coding achievement test scores was higher than the control group. This result differs from other research studies that claimed pair programming increased learners' coding achievement (Dongo et al., 2016; McDowell et al., 2002). On the other hand, the study of Braught et al. (2008) revealed no significant difference in scores of achievement between pair programmers and individual programmers. Moreover, the study of Demir and Seferoğlu (2020) found no significant difference in terms of their participants'coding achievement. Demir and Seferoğlu (2020) explained this result with the concept of social loafing. Social loafing is described as the tendency for people to put up smaller effort when working in a group as opposed to working on a task individually (Latane, Williams & Harkins, 1979). Therefore, learning of coding may have been hindered for the participants in the study of Demir and Seferoğlu (2020). However, coding achievement scores in the present study show that participants did not experience social loafing, which contradicts with Demir and Seferoğlu's results (2020). A reason for this finding of the current study might be

that the pre-service teachers in both groups were extremely driven to succeed and advance their programming skills regardless of whether they worked in pairs or individually (Hwang et al., 2012). For instance, in the qualitative findings of this study, pre-service teachers who were involved in pair programming, indicated that their coding achievement was enhanced thanks to the opportunities for learning from and supporting one another. Furthermore, this research study revealed that while both groups made progress in their coding performance, the experimental group experienced a more substantial increase when pre-test and post-test results were compared.

The duration of the lab sessions in this study might not have been sufficient for the pair programming group to make a significant progress in their coding. As Dyba et al. (2007) indicated in their meta-analysis, compared to solo programming, pair programming demands more working hours. In addition, according to Lewis' (2011) study, solo programmers finished their tasks faster than pair programmers since pair programmers spend time on communication. In contrast with the current study, Salge and Berente (2016) suggested that learners engaged in pair programming completed the coding tasks rapidly. The study's qualitative findings highlight that the pre-service teachers working in pairs needed longer time to create their programming projects. Time allocated for Scratch activities in this study was the same for both groups. Therefore, it is possible that the participants in the pair programming groups might have needed more time to develop their programming through collaboration, which was not provided in the current study.

One of the findings in the current study is that pre-service teachers engaging in pair programming tend to be realize and identify their coding mistakes at an earlier stage. Code created by paired programmers was more accurate than individual

programmers' code since faults were fixed with the help of their partners' input (Isong et al., 2016). In pair programming, the presence of a partner in pair programming ensures continuous code review, which significantly enhances the probability of detecting errors and flaws early (Gómez et al., 2017). Consequently, fewer flaws occur in projects, which is largely attributed to the early detection and correction of errors (Zhong et al., 2016). This also leads to a reduced level of frustration experienced by programmers who work in pairs during the debugging process compared to solo programmers (Williams & Kessler, 2000). In the current study, pre-service teachers who worked in pairs might have also felt a reduced level of frustration and been able to promptly identify and effortlessly rectify errors due to the presence of a partner. In addition to this, Wray (2010) highlights another critical aspect related to the early identification of bugs: the phenomenon of unintentional blindness. This occurs when a programmer becomes excessively engrossed in the coding task, inadvertently overlooking the bugs that would otherwise have been caught. The role of programmers working in pairs becomes crucial in such situations, as they are responsible for catching coding errors and mitigating the occurrence of unintentional blindness (Wray, 2010). Aligned with this argument, the participants who worked in pairs might have developed higher-quality codes by both quickly noticing their mistakes and fixing those promptly in the current study.

In the context of this study, coding quality refers to how well a project was created using the best code structures within the Scratch programming environment. The findings showed that pre-service teachers who worked in pairs generated higher-quality projects including more complex codes. Pair programming relates to higher coding quality (Demir & Seferoğlu, 2020). For example, Williams and Kessler (2002) found that pair programming is more efficient in terms of code quality than

individual work as learners involved in pair programming create higher-quality programs with fewer software bugs and fewer coding errors (Nawahdah & Taji, 2016). This is due to the fact that a different individual continuously examines the code, the solution is discussed and developed by two programmers, and information is shared between these partners (Dongo et al., 2016). In the current study, the participants who were doing pair programming may have had a high-quality project because there was a constant exchange of ideas between them during programming, and the different perspectives in pairs were constantly updating them.

For the pair programmers in this study, the continuous exchange of ideas provided a knowledge transfer and sharing of this knowledge. According to the literature, learners that participate in pair programming exchange programming knowledge with one another, which enhances the learning process (Dongo et al., 2016). Similarly, Bailey and Mentz (2017) also suggested that learners are able to exchange their experiences and ideas through collaborative learning, which results in better achievement. In this study, the participants working in pairs stated that they constantly benefited from each other's experiences while doing weekly activities and projects together during the entire lab course. This may have improved their learning process more than learning of participants who worked individually. On the contrary, for example, in the study of Zieris and Prechelt (2014), paired students had trouble understanding one another when programming, and they were unable to properly share their programming knowledge. The discrepancy may have resulted from different programming knowledge levels of learners in pairs. Moreover, when paired learners are inexperienced programmers, they learn better by realizing the fundamentals of programming and applying logic (Karaoğlu, 2018). Pre-service teachers who were involved in pair programming in this study were novice

programmers, and they did not mention any problems about sharing knowledge and communication between them. This may have helped to improve paired pre-service teachers' learning of programming.

Researchers asserted that pair programmers could accomplish the objectives of activities by exchanging information to resolve programming problems (Karaoğlu, 2018). Additionally, pair programming in educational contexts was shown to be effective by the study of Chao and Atli (2006) because it helped learners overcome programming challenges, rectify errors, and develop complex code. In addition, Dongo et al. (2016) suggested that pair programming is an effective method that improves learners' problem-solving skills. Similar to the literature, paired groups of this study who encountered complex codes indicated that working with someone else who had a different point of view had a significant impact on identifying and solving a problem. This finding is agreeing with prior research suggesting that diverse perspectives can improve problem-solving abilities (e.g., Çakır et al., 2021; Kiss & Arki, 2017).

In programming, it is possible to write code by creating stories to produce a well-designed product (Papadakis, Kalogiannakis, Zaranis & Orfanakis, 2016). Coding requires creativity since it encourages learners to come up with fresh opinions (Grover & Pea, 2013). For example, Yang, Ng, and Su (2023) found that story-inspired programming improved students' CT skills which consist of creativity. The findings of the current study demonstrated that participants in both groups developed their creativity. The reason could be that the participants created projects by constructing stories for their weekly coding activities.

The current study directly examined the effect of pair programming on pre-service teachers' CT skills and coding achievement. Unlike Wei et al.'s study (2021),

which found a significant difference between CT skills of the students doing pair programming and solo programming, the current study did not find a difference between the CT skills of the groups. In Wei et al.'s (2021) study, the number of participants were 171 students. Moreover, McDowell et al. (2002) found a significant difference between the coding performance of pair and solo programmers. Unlike this study, the current study did not find a difference between the coding achievement of the groups. In the study of McDowell et al. (2002), they gathered data from 600 undergraduate students. Denner et al. (2014) also investigated pair programming, analyzing data from 320 participants who completed tasks individually or in pairs. They found pair programming is beneficial as it enhanced CT and programming skills of the participants. The number of participants of the current study were 36 pre-service teachers. Sample size may have led to the quantitative results in this study.

In addition, the other purpose of the current study is to explore the pre-service teachers' learning experiences of programming. The findings also showed that pair programming is an effective way for programming instruction because thanks to pair programming, the learners are able to recognize and get rid of the concerns they face concurrently together (Karaoğlu, 2018). Qualitative findings of the current study showed that pair programming enabled pairs to communicate and exchange information with each other, which allowed them to identify coding errors early on and collaboratively develop high-quality projects. This result aligns with existing literature, suggesting that pair programming can be an effective strategy for programming education (Braught et al., 2008; Ghorashi & Jensen, 2017; Iskrenovic-Momcilovic, 2019; Lyon & Magana, 2020).

In conclusion, this study provided valuable insights into the effects of pair programming on coding achievement and CT skills of pre-service teachers. Integrating quantitative and qualitative data, the findings illuminated the impact of pair programming on the participants' learning experiences and outcomes. The results revealed no significant difference in CT skills and coding achievement between the pair programming and individual programming groups, although the mean scores of the pair programming group were higher. Possible explanations for these results include the inherent integration of CT skills in programming education, the use of Scratch as a beneficial programming language for novice learners, and the influence of group dynamics and online environments on pair programming effectiveness. The study also highlighted the benefits of pair programming in early error detection, higher coding quality, knowledge transfer, problem-solving, and creativity development. The duration of lab sessions and sample size limitations may have impacted the quantitative findings. Yet, the qualitative findings supported the effectiveness of pair programming in enhancing the learning experiences and outcomes of pre-service teachers' programming. The study concludes by suggesting the inclusion of programming instruction in teacher education curricula and recommending further research to explore the optimal conditions and strategies for implementing pair programming in different contexts.

5.1 Recommendations and implications for future research

The current study is important because there is a lack of studies investigating pre-service teachers' CT skills and coding achievement when they complete programming tasks individually or collaboratively. From a practical point of view, the findings of the study recommend that educational technologists, instructional

designers, and instructors take advantage of pair programming for programming instruction of pre-service teachers in higher education.

This mixed-method study contributed to the literature by providing qualitative data regarding the impact of pair programming in online settings. Sharing information, exchanging ideas, and fixing mistakes were crucial elements in the study's approach to problem-solving during programming. The experiences of the pair programmers of the study can be useful to learners who are new to computer programming in determining the best approaches to solving the issues that arise during programming.

The study might help fill the gap in the literature about the role of pair programming in teacher education. This mixed-method study may help to show the impact of pair programming on CT skills and coding achievement by thoroughly exploring learning experiences of the pair and individual programmers because limited number of studies have been implemented for pre-service teachers' learning of programming so far. By looking at several characteristics such CT skills, coding achievement, information sharing, and code quality, the study may also provide a wide point of view about the use of pair programming in education. Future studies on how the pair programming technique influences pre-service teachers' programming instruction may use the study as a model. Further research may investigate the coding quality collected as quantitative data for pre-service teachers.

The study employed the Scratch visual programming environment which was very helpful for learners of the current study. For further studies, different programming tool may be utilized for applying pair programming. In addition, the participants of the study were pre-service teachers who were novice programmers. In

a further study, pair programming might be applied in a course with students who are advance programmers, and students at different levels.

5.2 Limitations of the study

This study has some limitations. One of the limitations might be the participants' selection. Although the course sections were randomly assigned to control and experimental groups, the participants themselves were not randomly distributed into groups. For the aim of the study, the appropriate sample was selected at Boğaziçi University. It may cause some threats on the internal validity of the study. So, generalizability of results is restricted.

Another threat to the validity of the results may be the small sample size. This research involved only 36 pre-service teachers, which might not be enough for generalizing the findings of the study to a larger population, the study can be repeated by choosing participants from various programs with a bigger number.

The other limitation might be the environment of the study. This study was implemented in online settings. Since the environment was online, it was not possible to observe all the participants at the same time when they completed the tasks and activities. Therefore, the study may be repeated by selecting different learning environments.

APPENDIX A

PARTICIPANT INFORMATION AND CONSENT FORM

Research sponsoring institution: BOĞAZİÇİ UNIVERSITY
Title of the research project: A Mixed-Method Study: Pre-service Teachers' Learning of Block-Based Programming and Computational Thinking
Project Manager: Assisst. Prof. Duygu Umutlu | E-mail address:

Phone:

Researcher: İlke Topal | E-mail address: Phone:

Dear students,

Dr. Duygu Umutlu who is an instructor in the Department of Computer Education and Educational Technology at Boğaziçi University is carrying out a scientific research project. The research aims to explore how completion of block-based programming activities collaboratively or individually will affect preservice teachers' programming and computational thinking skills.

CET 360 Instructional Technologies and Material Development course prepares teacher candidates at Boğaziçi University to use technologies effectively when they start teaching in real classrooms. As a student in this course, you will complete several lab activities. And, Scratch, a block-based programming platform, will be used in lab sessions. During the semester, you will complete these lab activities either individually or in groups over 10 weeks.

To participate in the research, you should complete a test (it will not be graded) and a survey that will be sent to you by e-mail in the beginning and at the end of the semester. If you volunteer for this research, your test and survey scores will constitute data, and you can volunteer to join the interviews to share your learning experiences of block-based programming at the end of the semester. The interviews will be conducted via Zoom videoconference platform and they will be recorded as an audio file.

Your identities will not be shared at any part of the research. Your involvement in the study is voluntary, and you may choose not to participate or to stop participating at any time without penalty. The decision to be in the study or not to be in the study and researcher's evaluation of your work will not affect your grades. However, your participation in the research will make a contribution to support and create effective online discussions. The stored records will be destroyed after the completion of the research. There are no known risks or inconveniences associated with this study.

Please ask if you have any questions about the study before signing this form. If you have any questions later, you can ask the project coordinator (Phone:). You can consult the Boğaziçi University Ethics Committee for Master and PhD Theses in Social Sciences and Humanities (SOBETIK) about your rights regarding research (Mail:).

I have read the text above and understood the scope and purpose of the research I was asked to participate in. I realized that I could quit this study whenever I wanted, without having to give any reason, and that I would not encounter any negativity if I quit.

Participant Name-Surname:……. I approve to participate in the study. ☐ I approve to join an audio-recorded interview. ☐ Date (day/month/year):...../...../.....	Researcher Name-Surname: İlke Topal

APPENDIX B

CODING ACHIEVEMENT TEST

1) Herhangi bir problemin çözümü için oluşturulan yapının, görsel olarak simge ya da sembollerle ifade edilmiş şekline …….. denir.

a) Koşul ifadesi

b) Operatör

c) Akış şeması

d) Döngü

e) Değişken

2) Girilen verileri alan veya programın çalışmasıyla bazı verilerin atandığı ve bu verilerin tutulduğu yapılara …… denir.

a) Operatör

b) Akış diyagramı

c) Değişken

d) Döngü

e) Koşul ifadesi

3) Bir programda tekrar edilme işleminin ne kadar yapıldığı verisini kaydeden değişkene …….. denir.

a) Döngü

b) Sayaç

c) Koşul ifadesi

d) Operatör

e) Algoritma

4) Önceden tanımlanmış birtakım matematiksel ya da mantıksal işlemleri yapmak için kullanılan özel karakterlere …… denir.

a) Değişken

b) Döngü

c) Operatör

d) Koşul ifadesi

e) Algoritma

5) Bir programın akışının belirtilen şarta veya şartlara göre dallanmasını sağlayan yapılara …… denir.

a) Koşul ifadesi

b) Operatör

c) Veri

d) Değişken

e) Sayaç

6) Aşağıda bir program geliştirilirken izlenen adımlar verilmiştir.

I) Yöntemi bir programlama dili ile kodlama

II) Çözüm yöntemi bulma/geliştirme

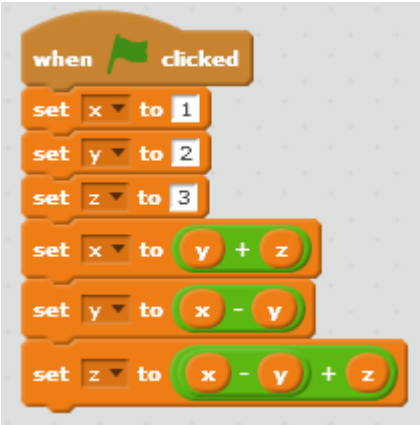
III) Problemi/görevi anlama

IV) Programı test edip varsa hataları saptama ve düzeltme

Yukarıdaki adımların doğru sırası nasıl olmalıdır?

a) III-I-IV-II b) II-IV-III-I c) III-II-I-IV d) IV-III-II-I e) II-III-I-IV

7) Aşağıdaki kod bloğu çalıştırıldığında x, y ve z değişkenleri sırasıyla en son hangi değerleri alırlar?



	x değişkeni	y değişkeni	z değişkeni
a)	1	2	3
b)	5	3	6
c)	3	2	1
d)	5	3	5
e)	5	3	-1

8) Aşağıdaki kod bloğu, verilen 3 uzunluğun bir üçgenin kenar uzunluğu olup olamayacağını kontrol eder (Üçgende bir kenarın uzunluğu diğer 2 kenarın uzunluğunun toplamından küçüktür).



Yukarıdaki bilgiler ışığında "?" işaretli iki yere sırasıyla ne gelmelidir?

	1. soru işareti	2. soru işareti
a)	and	or
b)	or	or
c)	or	and
d)	not	and
e)	and	and

9) Aşağıdaki kod bloğunun ekrana “Tek sayıdır” çıktısını vermesi için koşul bölümüne ne yazılmalıdır?



a) 

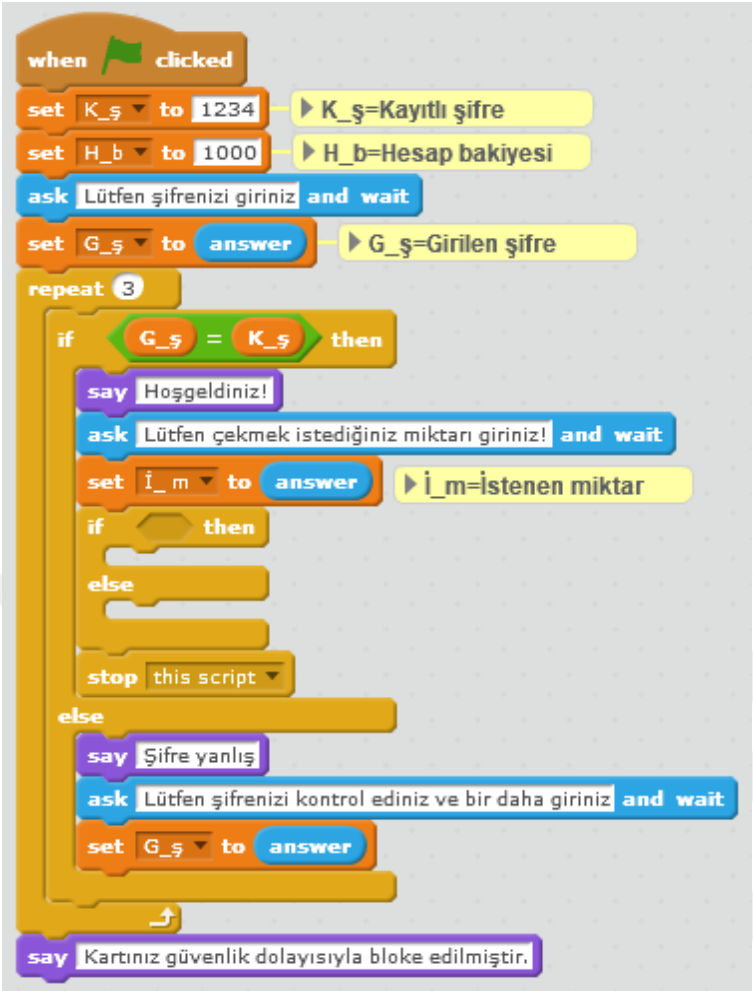
b) 

c) 

d) 

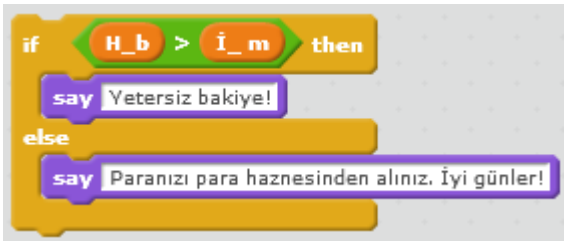
e) 

10) Aşağıda bir bankamatiğin çalışma ilkelerini içeren kodlar verilmiştir.

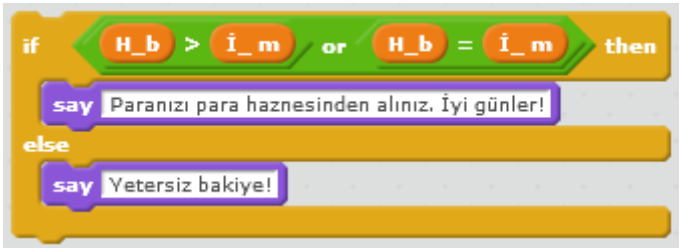


Yukarıdaki kodda bulunan "if-else" boşluğuna gelmesi gereken doğru ifade aşağıdaki seçeneklerden hangisinde yer almaktadır?

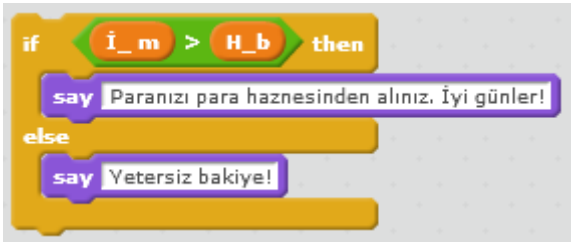
a)



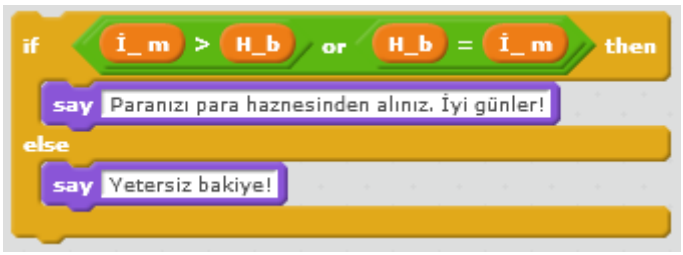
b)



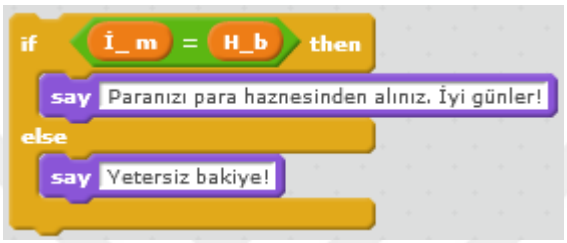
c)



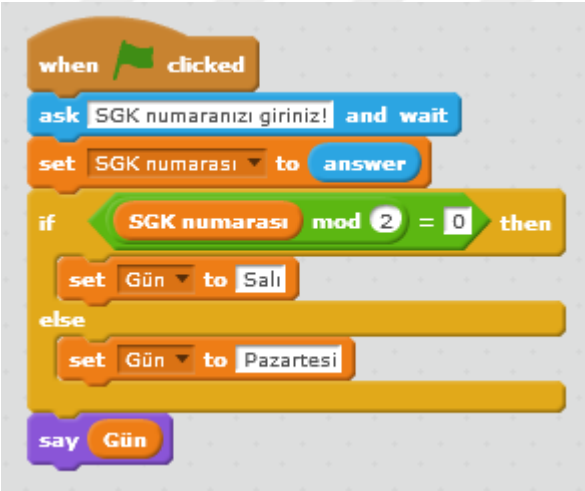
d)



e)



11) Aşağıda bir iş yerinde çalışan personelin maaş günüyle alakalı bir program verilmiştir.



Yukarıdaki programla ilgili aşağıda verilenlerden hangisi doğrudur? (SGK=Sosyal Güvenlik Kurumu)

a) SGK numarası ne olursa olsun herkes maaşını Salı günü alır.

b) SGK numarasının son hanesi 0 olanlar Pazartesi günü maaş alırlar.

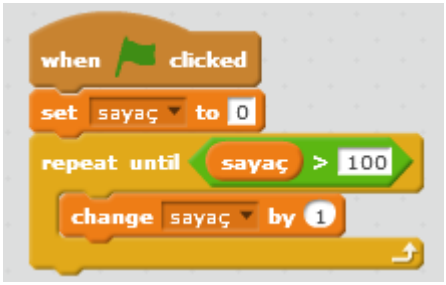
c) SGK numarasının son hanesi 2 olanlar Pazartesi günü maaş alırlar.

d) SGK numarası tek olanlar Salı günü, diğerleri Pazartesi günü maaş alırlar.

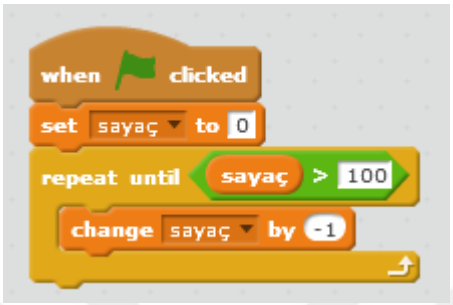
e) SGK numarası çift olanlar Salı günü, diğerleri ise Pazartesi günü maaş alırlar.

12) Aşağıda verilen döngülerin hangisi sonsuz döngüye girer?

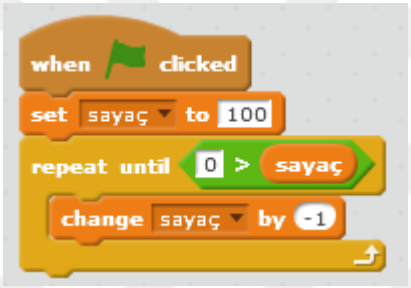
a)



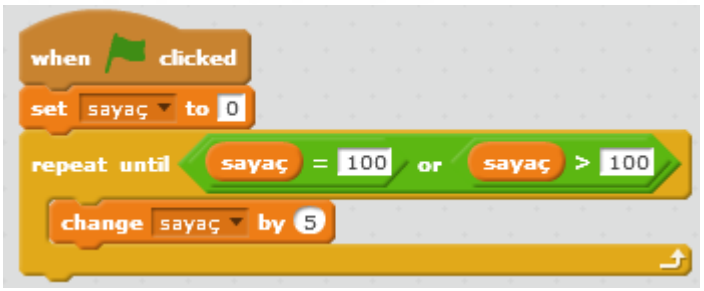
b)



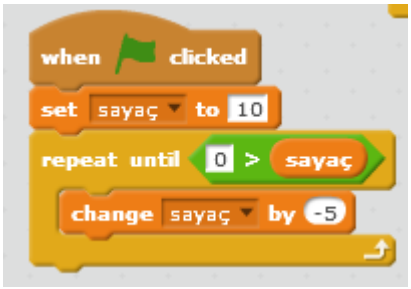
c)



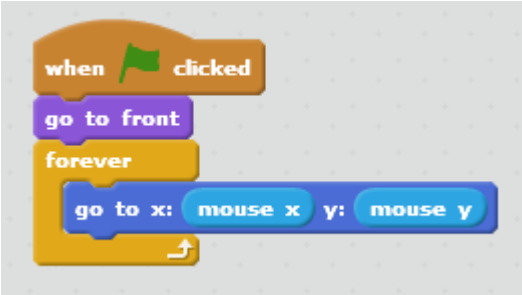
d)



e)



13) Aşağıda bir döngü içine yazılmış bir satırlık kod bulunduran bir kod bloğu verilmiştir.



Yukarıdaki kod bloğuyla ilgili olarak aşağıda belirtilenlerden hangisi doğrudur?

a) Nesne sürekli olarak fare imlecinin x ve y koordinatlarına gider.

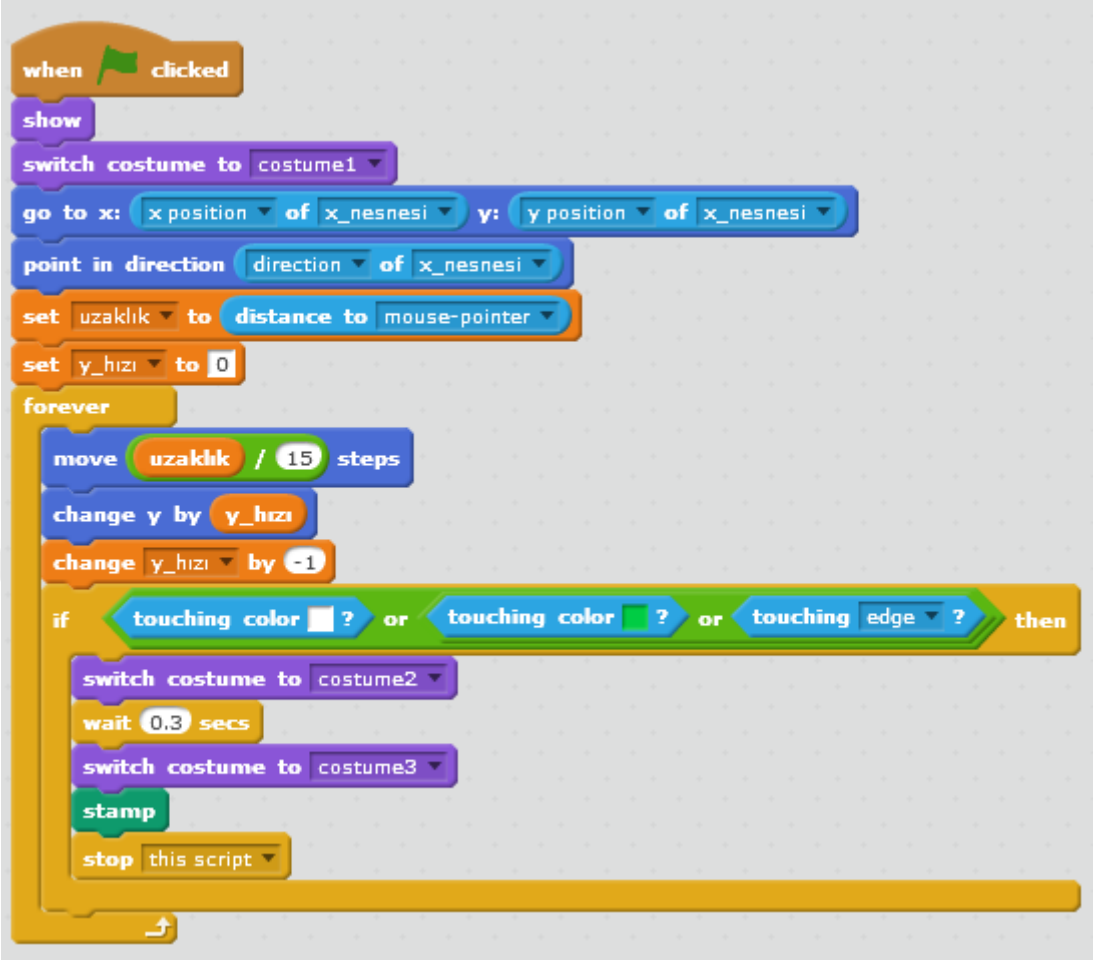
b) Nesne sadece program başladığında fare imlecinin x ve y koordinatlarına gider.

c) Nesne, “mouse x” ve “mouse y” isimli iki nesnenin arasına gider.

d) Nesne, “mouse x” nesnesinin x koordinatına ve “mouse y” nesnesinin y koordinatına gider.

e) Fare imleci sürekli olarak nesnenin x ve y koordinatlarına gider.

14) Aşağıda bir Scratch nesnesinin bir kod bloğu verilmiştir.



Yukarıdaki kod bloğuyla ilgili olarak aşağıda verilenlerden hangisi yanlıştır?

a) Döngüden önce nesnenin fare imlecine olan uzaklığı "uzaklık" değişkenine aktarılır.

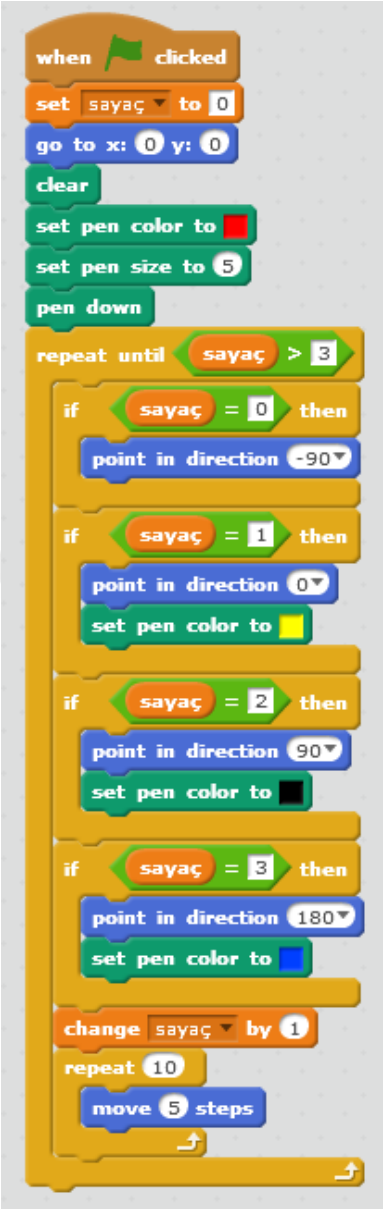
b) Nesnenin ilk başta fare imlecine uzaklığı ne kadar fazlaysa nesne o kadar hızlı hareket eder.

c) Eğer nesne beyaz veya yeşil renklerden birine veya kenara değerse nesne toplamda 3 kez kostüm değiştirmiş olur.

d) Eğer nesne beyaz veya yeşil renklerden birine veya kenara değerse nesne silinir.

e) Nesne ilk başta "x_nesnesi"nin x ve y koordinatlarına gider ve yine "x_nesnesi"yle aynı doğrultuya döner.

15) Aşağıda bir Scratch nesnesinin hareketine ilişkin bir kod bloğu verilmiştir.



-90 yönünün sol, 0 yönünün yukarı, 90 yönünün sağ ve 180 yönünün aşağı olduğu düşünüldüğünde yukarıdaki kod bloğuyla ilgili olarak aşağıda belirtilenlerden hangisi doğrudur?

a) Ekranda kenarları eşit olmayan bir dikdörtgen oluşur.

b) Ekranda 4 farklı renkten meydana gelen düz bir çizgi oluşur.

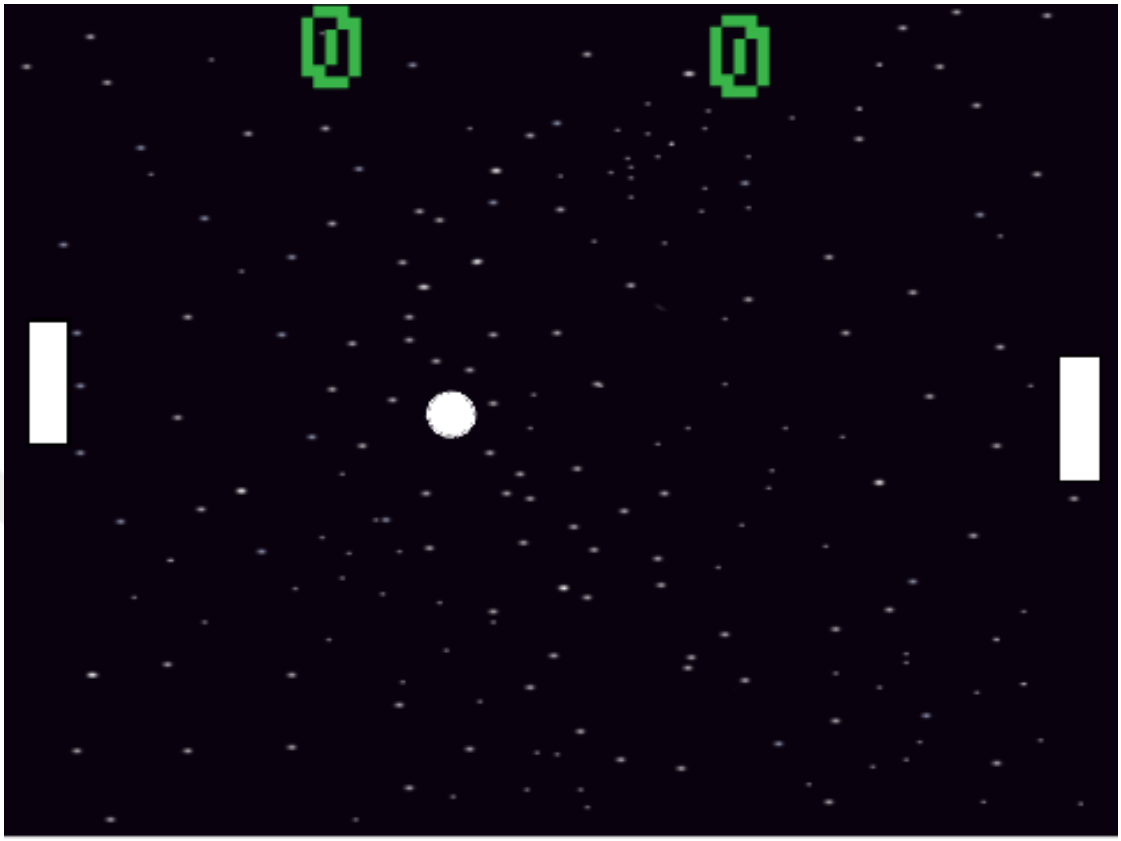
c) Nesne ilk başta ekranda çizgi çizmeden hareket eder.

d) Nesnenin sağ kenarının rengi kırmızı olur.

e) Ekranda her kenarının rengi farklı bir kare oluşur.

16) Klasik bir oyun olan ping pong (masa tenisi) oyununda amaç size doğru gelen topa dokunarak/vurarak topu karşı tarafa göndermektedir. Eğer oyunculardan biri topa dokunamazsa/vuramazsa oyunu kaybeder.

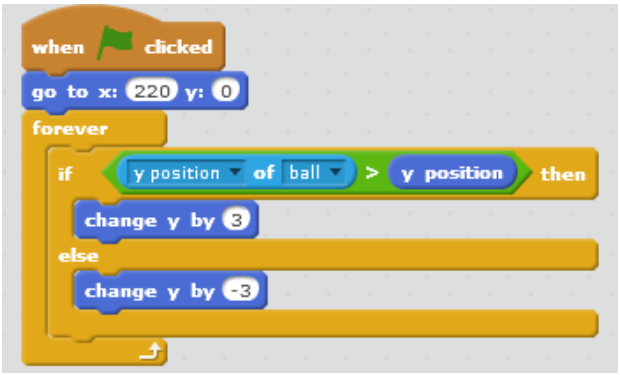
Aşağıdaki oyunda “ball” olarak isimlendirilen top ilk başta rastgele hareket etmektedir. Koordinat sistemine göre “y” koordinatları yukarı çıktıkça artarken aşağı indikçe düşmektedir. Oyunda sol taraftaki oyuncu bir insan tarafından, sağ taraftaki oyuncu ise program tarafından kontrol edilmektedir. Oyunun aşağıda bir ekran



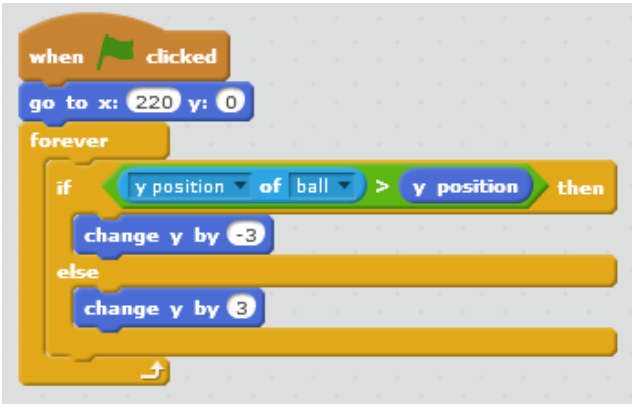
görüntüsü verilmiştir.

Yukarıda verilenler düşünüldüğünde aşağıdakilerden hangisi sağ taraftaki oyuncunun kodu olabilir?

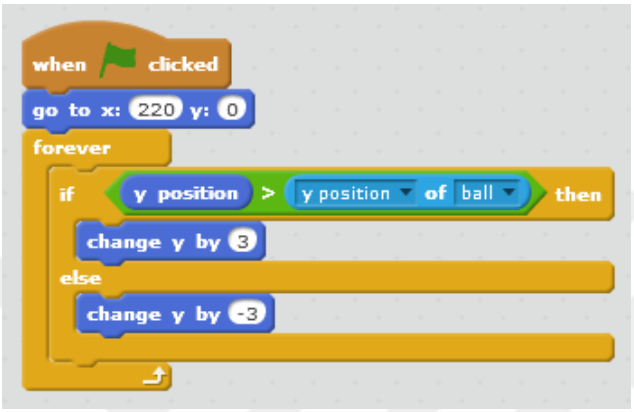
a)



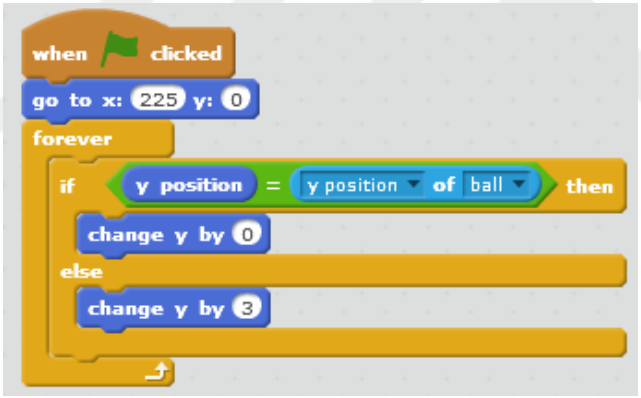
b)



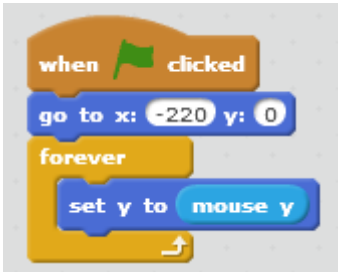
c)



d)



e)

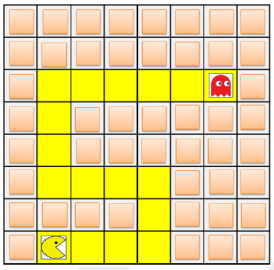


17) Aşağıdaki klasik bir oyun olan Pacman verilmiştir. Bu oyunda kırmızı renkteki hayalet karakteri sarı renkteki Pacman'i yemek istemektedir. Oyunda "x" koordinatları sola gittikçe, "y" koordinatları ise aşağı gittikçe azalmaktadır. Oyunda her bir kare bir birimdir. Aşağıda program tarafından kontrol edilen "Hayalet" isimli karakterinin kodu verilmiştir.

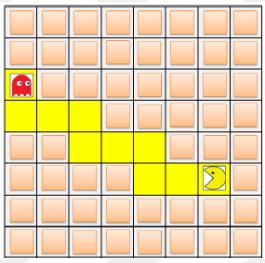


Yukarıdaki kod dikkate alındığında "Hayalet" karakteri aşağıdaki şıklardaki durumlardan hangisinde Pacman karakterini hiçbir duvara çarpmadan yer?

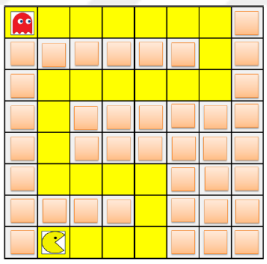
a)



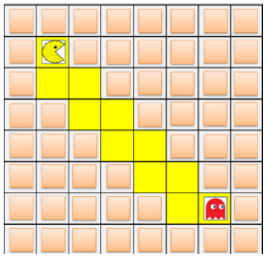
b)



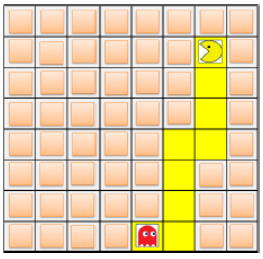
c)



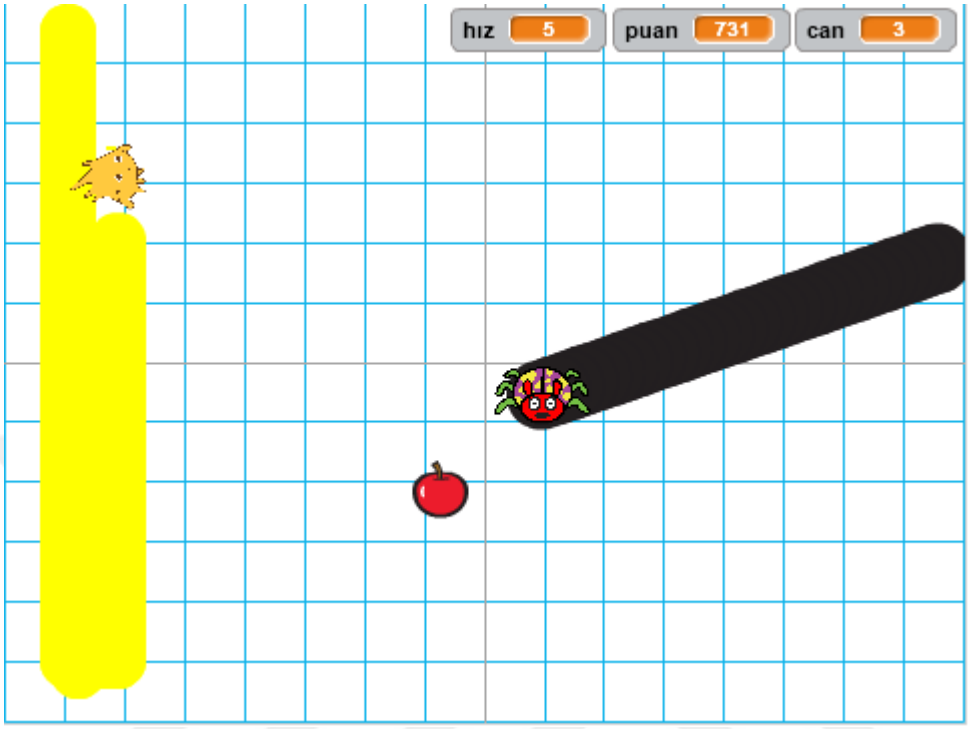
d)



e)

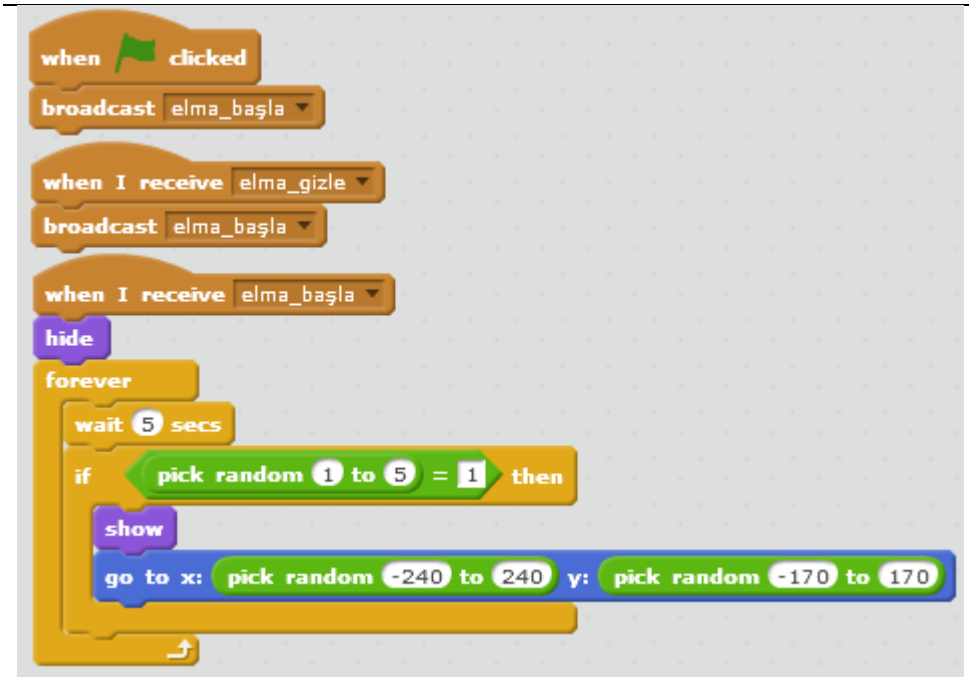


18) Boyacı isimli oyunda klavyeden kontrol edilen boyacı ekranı sarıya boyamaya çalışırken, program tarafından kontrol edilen örümcek ekranı siyaha boyamaya çalışmaktadır. Elma ise boyacı tarafından yenildiğinde boyacının hızını artırmaktadır. Aşağıda oyunun bir ekran görüntüsü verilmiştir.

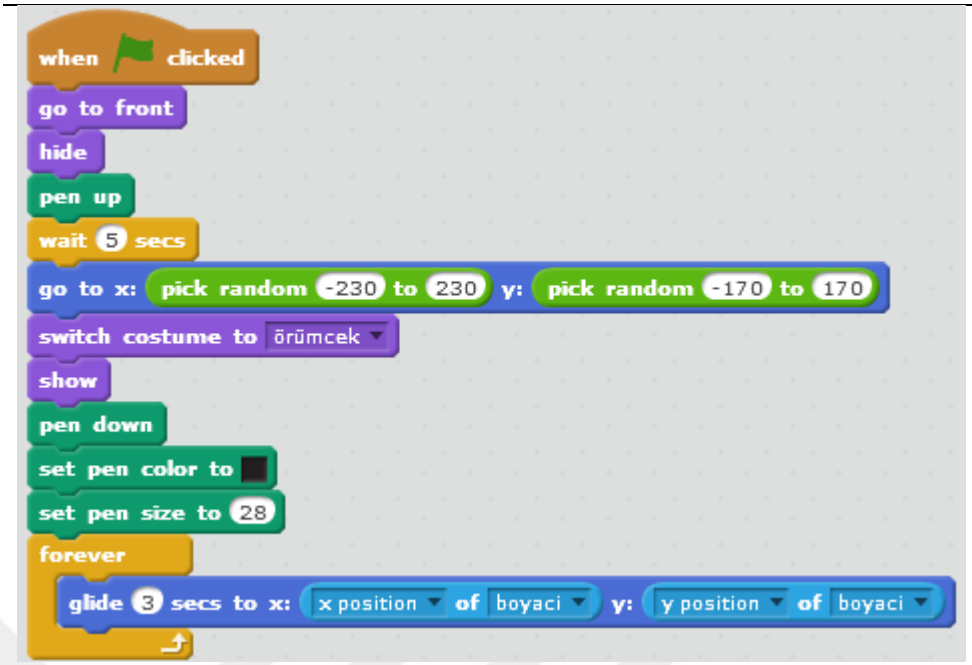


Yukarıdaki oyunda yer alan "boyacı, örümcek ve elma" Scratch nesnelerinin kodlarının bir kısmı aşağıda verilmiştir.

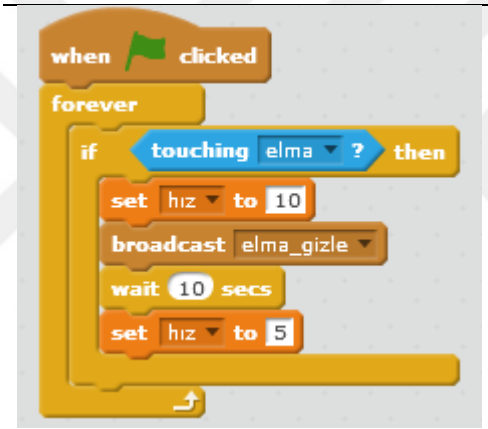
Elma



Örümcek



Boyacı



Yukarıdaki kodlar dikkate alındığında aşağıdakilerden hangisi yanlıştır?

a) Boyacı elmayı yediğinde boyacının hızı 10 saniye boyunca 10 birim olur.

b) Örümcek oyun başladıktan 5 saniye sonra rastgele bir konumda ortaya çıkar.

c) Oyun başladıktan sonra elmanın nerede ve ne zaman ortaya çıkacağı kestirilemez.

d) Örümcek ortaya çıktığı anda boyacının konumuna doğru hareket eder.

e) Boyacı elmayı yediği anda elma hemen başka bir yerde tekrar ortaya çıkar.

APPENDIX C

COMPUTATIONAL THINKING SKILLS SCALE

Items		Strongly Agree (5)	Agree (4)	Neutral (3)	Disagree (2)	Strongly Disagree (1)
Problem Solving	I am usually able to finish a given task on time.					
	I can decide what I want to do before I create a new product.					
	I believe that I can solve problems that I encounter for the first time.					
	I can run a design application using appropriate commands.					
	I assess each stage separately when solving a problem.					
	If I run into a problem when trying to find a solution, I review the stage at which I encountered the problem instead of starting over.					
	I believe that I will have better results if I perform tasks in a planned manner.					
	I can determine what to do step by step when I am striving to achieve a goal.					
	I know that everything has a certain order and a principle for working it out.					
	I plan what needs to be done before I start performing a task.					

	I believe that everything must be done in a logical order.					
	When faced with a problem, I first decide on what to do.					
	When I experience a problem, I can think about everything which might cause it.					
	I learn from the mistakes I've made when solving a problem.					
	When I encounter a problem, I first try to understand the cause of the problem.					
	I use a systematic method to compare options and make a decision.					
	When performing a task, I try to decide on my next step.					
	I can test the accuracy of any operation which I have performed.					
	When I encounter a problem, I try to use solutions which have worked for me in the past.					
	I try to find a more effective solution for a given problem.					
Cooperative learning & critical thinking	It is a waste of time to try to understand different opinions related to how to solve a problem.					
	I have difficulties when communicating with other group members in cooperative learning groups.					
	Learning in cooperative learning groups is more difficult for me.					
	Cooperative learning reduces my eagerness to learn.					

	The accuracy of a solution depends on the number of people who accept the said solution.					
	If how to solve the problem is known, there is no need to look for a better solution.					
	When I experience a problem, I apply the solution used by others around me without thinking.					
	Not everybody makes the necessary effort to perform tasks in cooperative learning.					
Creativity	I enjoy coming up with new ideas that nobody has thought of before.					
	I get bored of doing the same thing.					
	I enjoy designing systems to perform a task automatically.					
	I am curious about how the structure of systems that perform a task and how they work.					
	I am interested in the design of systems which make people's work easier.					
	I enjoy solving similar problems.					
	I enjoy finding a solution which has not been used before.					
	It makes me proud to solve a problem using a different method.					
	It makes me happier to try to find new things.					
Algorithmic Thinking	Once I finish a task, I ask myself whether or not there is an easier way to do it.					
	If I encounter a problem in any of the steps needed to solve it, I start over.					

	I try to apply the solutions that I have found to other problems as well.					
	I think about how to achieve my goals more easily in relation to all subjects.					
	Before performing a task, I plan out how to do it in my mind.					

APPENDIX D

INTERVIEW QUESTIONS

Kısaca kendinden bahsedebilir misin? Adın, bölümün, kaçıncı sınıfta olduğun.

(*Can you briefly talk about yourself? Your name, department, and what grade you are in.*)

Bu dönem CET 360 dersi boyunca Scratch'te blok tabanlı kodlama etkinlikleri tamamladınız. Bu etkinlikler hakkında ne düşünüyorsun? Deneyimlerini genel olarak paylaşabilir misin?

(*Throughout this term in CET 360 course, you completed block-based coding activities on Scratch. What do you think about these activities? Can you share your overall experiences*)

Bu kodlama etkinliklerini bireysel olarak mı eşli olarak mı tamamladın? Tamamladığın formatı nasıl değerlendirirsin? Diğer formatı nasıl değerlendirirsin?

(*Did you complete these coding activities individually or in a pair? How would you evaluate the format you completed? How would you evaluate the other format?*)

Bireysel kodlamayı mı eşli kodlamayı mı tercih ederdin?

(*Would you prefer individual or pair programming?*)

Bireysel/eşli kodlamanın artıları ve eksileri sana göre nedir?

(*What are the pros and cons of individual/pair programming, in your opinion?*)

Sence bu kodlama etkinliklerinin problem çözme becerisiyle bir ilişkisi var mıdır? Varsa nasıl? Bireysel/eşli olarak kodlama yapmayı problem çözme açısından nasıl değerlendirirsin?

(*Do you think there is a relationship between these coding activities and problem-solving skills? If so, how? How do you evaluate individual/pair programming in terms of problem-solving?*)

Kodlama sırasında grup arkadaşınla iletişimini ve işbirliğini nasıl değerlendirirsin?

(*How do you evaluate your communication and collaboration with your pair mates during coding?*)

Eşli kodlama sırasındaki iletişim ve iş birliği senin kodlama başarını ve motivasyonunu nasıl etkiledi sence?

(*How do you think the communication and collaboration during pair programming affected your coding success and motivation?*)

Kodlama etkinlikleri sırasında yaratıcılığını kullandığını düşünüyor musun?

(*Do you think you used your creativity during the coding activities? Can you give examples of where you used your creativity?*)

Yaratıcılığını kullandığın örnekler verebilir misin?

(*Can you give examples of where you used your creativity?*)

> Bu verdiğin örneklerde sence bireysel/grup çalışması yapıyor olmanın nasıl bir etkisi olmuş olabilir?
>
> (*What kind of impact do you think doing individual/pair work may have had on these examples?*)

Kodlama etkinlikleri sırasında takıldığın noktalar oldu mu? Bu durumlarda ne gibi yöntemler kullandın? Örnek verebilir misin? Bu problemleri çözerken bireysel/grup çalışmasının ne gibi katkıları olmuş olabilir?

(*Were there any points where you got stuck during the coding? What methods did you use in these situations? Can you give an example? What contributions could individual/pair work have had in solving these problems?*)

Dönem boyunca lab etkinliklerinin en son aşaması olan 'Create' aşamasında projeler tasarladın. Bu projeleri hazırlarkenki deneyimlerini paylaşabilir misin?

(*Throughout the term, you designed projects during the final phase of lab activities, the 'Create' stage. Can you share your experiences while preparing these projects?*)

> Kullanmakta zorladığın kodlar oldu mu? Olduysa hangileri? Neden?
>
> (*Were there any codes you struggled to use? If so, which ones? Why do you think?*)
>
> Bu projeleri oluştururken sana kolay ve zor gelen kısımlar hangileriydi? Örnek verebilir misin?
>
> (*What were the easy and challenging parts for you while creating these projects? Can you give an example?*)
>
> Bu projeleri tasarlarken takıldığın noktalar oldu mu? Bu sorunları nasıl çözdün? Hangi CT stratejilerini kullandın?
>
> (*Were there any points where you got stuck while designing these projects? How did you solve these problems? Which CT strategies did you use?*)

APPENDIX E

THE PERMISSION FROM ETHICS COMMITTEE OF BOGAZICI UNIVERSITY

Evrak Tarih ve Sayısı: 01.11.2021-36417

T.C.
BOĞAZİÇİ ÜNİVERSİTESİ
SOSYAL VE BEŞERİ BİLİMLER YÜKSEK LİSANS VE DOKTORA TEZLERİ ETİK İNCELEME KOMİSYONU
TOPLANTI KARAR TUTANAĞI

Toplantı Sayısı : 22
Toplantı Tarihi : 13.10.2021
Toplantı Saati : 14:00
Toplantı Yeri : Zoom Sanal Toplantı
Bulunanlar : Prof. Dr. Ebru Kaya, Prof. Dr. Fatma Nevra Seggie, Dr. Öğr. Üyesi Yasemin Sohtorik İlkmen
Bulunmayanlar :

İlke Topal
Bilgisayar ve Öğretim Teknolojileri Eğitimi Bölümü

Sayın Araştırmacı,

"A Mixed-Method Study: Pre-service Teachers' Learning of Block-Based Programming and Computational Thinking" başlıklı projeniz ile ilgili olarak yaptığınız SBB-EAK 2021/62 sayılı başvuru komisyonumuz tarafından 13 Ekim 2021 tarihli toplantıda incelenmiş ve uygun bulunmuştur.

Bu karar tüm üyelerin toplantıya çevrimiçi olarak katılımı ve oybirliği ile alınmıştır. COVID-19 önlemleri kapsamında kurul üyelerinden ıslak imza alınamadığı için bu onay mektubu üye ve raportör olarak Fatma Nevra Seggie tarafından bütün üyeler adına e-imzalanmıştır.

Saygılarımızla, bilgilerinizi rica ederiz.

Prof. Dr. Fatma Nevra SEGGIE
ÜYE

e-imzalıdır
Prof. Dr.Fatma Nevra SEGGIE
Raportör

SOBETİK 22 13.10.2021

Bu belge 5070 sayılı Elektronik İmza Kanununun 5. Maddesi gereğince güvenli elektronik imza ile imzalanmıştır.

APPENDIX F

DESCRIPTION OF SCRATCH TASKS

Week	Scratch Tasks	Parts of the Course Session		
		Phase 1	Phase 2	Phase 3
#1	When Clicked	Participants will first explore some projects about 'When Clicked' code. Also, they will explore *when this sprite clicked* block or *mouse down?* block. After they explore these code blocks, they will answer this question: What happens when you click? In the second step, they will read the code for a 'When Clicked' project. Then, they will explore what happens in the project.	In this phase, participants will first create a remixing activity for a 'When Clicked' code by changing three things in the given project. In the second step, they will create a short project by using the 'When clicked' code.	This phase will be for participants' reflection. First, they will give an example of one or two things they did well in their short project. Second, they will answer this question: What changes would you make if you coded this project again? Finally, they will share their responses with other participants in the course.
#2	Parallelism	Participants will first explore some projects about 'Parallelism' code. Also, they will explore two *when green flag clicked* blocks. After they explore this code block, they will answer this question: What are two things that happen when the green flag is clicked? In the second step, they will read the code for a 'Parallelism' project. Then, they will explore what happens in the project.	In this phase, participants will first create a remixing activity for a 'Parallelism' code by changing three things in the given project. In the second step, they will create a short project by using the 'Parallelism' code.	This phase will be for participants' reflection. First, they will give an example of one or two things they did well in their short project. Second, they will answer this question: What changes would you make if you coded this project again? Finally, they will share their responses with other participants in the course.

#3	Key Press	Participants will first explore some projects about 'Key Press' code. Also, they will explore *when key pressed* block or *a key pressed* block. After they explore these code blocks, they will answer this question: What key does the user press? What happens? In the second step, they will read the code for a 'Key Press' project. Then, they will explore what happens in the project.	In this phase, participants will first create a remixing activity for a 'Key Press' code by changing three things in the given project. In the second step, they will create a short project by using the 'Key Press' code.	This phase will be for participants' reflection. First, they will give an example of one or two things they did well in their short project. Second, they will answer this question: What changes would you make if you coded this project again? Finally, they will share their responses with other participants in the course.
#4	Loops	Participants will first explore some projects about 'Loops' code. Also, they will explore *repeat* block *or a forever* block. After they explore these code blocks, they will answer this question: What happens multiple times in the project? In the second step, they will read the code for a 'Loops' project. Then, they will explore what happens in the project.	In this phase, participants will first create a remixing activity for a 'Loops' code by changing three things in the given project. In the second step, they will create a short project by using the 'Loops' code.	This phase will be for participants' reflection. First, they will give an example of one or two things they did well in their short project. Second, they will answer this question: What changes would you make if you coded this project again? Finally, they will share their responses with other participants in the course.
#5	Broadcast	Participants will first explore some projects about 'Broadcast' code. Also, they will explore *broadcast block and a when I receive block with a matching message*. After they explore these code blocks, they will answer this	In this phase, participants will first create a remixing activity for a 'Broadcast' code by changing three things in the given project. In the second step, they will create a short project by	This phase will be for participants' reflection. First, they will give an example of one or two things they did well in their short project. Second, they will answer this question: What changes would you make if you

		question: What happens when the message is broadcast? In the second step, they will read the code for a 'Broadcast' project. Then, they will explore what happens in the project.	using the 'Broadcast' code.	coded this project again? Finally, they will share their responses with other participants in the course.
#6	Color Sensing	Participants will first explore some projects about 'Color Sensing' code. Also, they will explore *touching color* block. After they explore these code blocks, they will answer this question: What happens when that color is touched? In the second step, they will read the code for a 'Color Sensing' project. Then, they will explore what happens in the project.	In this phase, participants will first create a remixing activity for a 'Color Sensing' code by changing three things in the given project. In the second step, they will create a short project by using the 'Color Sensing' code.	This phase will be for participants' reflection. First, they will give an example of one or two things they did well in their short project. Second, they will answer this question: What changes would you make if you coded this project again? Finally, they will share their responses with other participants in the course.
#7	Random	Participants will first explore some projects about 'Random' code. Also, they will explore *pick random* block. After they explore this code block, they will answer this question: What is something that happens randomly in this project? In the second step, they will read the code for a 'Random' project. Then, they will explore what happens in the project.	In this phase, participants will first create a remixing activity for a 'Random' code by changing three things in the given project. In the second step, they will create a short project by using the 'Random' code.	This phase will be for participants' reflection. First, they will give an example of one or two things they did well in their short project. Second, they will answer this question: What changes would you make if you coded this project again? Finally, they will share their responses with other participants in the course.

#8	Ask and Answer	Participants will first explore some projects about ‘Ask and Answer’ code. Also, they will explore *ask and wait* block and *an answer* block. After they explore these code blocks, they will answer these questions: What does the user have to type into the project? What does the project do with that answer? In the second step, they will read the code for a ‘Ask and Answer’ project. Then, they will explore what happens in the project.	In this phase, participants will first create a remixing activity for a ‘Ask and Answer' code by changing three things in the given project. In the second step, they will create a short project by using the ‘Ask and Answer' code.	This phase will be for participants’ reflection. First, they will give an example of one or two things they did well in their short project. Second, they will answer this question: What changes would you make if you coded this project again? Finally, they will share their responses with other participants in the course.
#9	Variables	Participants will first explore some projects about ‘Variables’ code. Also, they will explore and find a *variable that changes how something happens in the given project*. After they explore the variable, they will answer these questions: What is the name of the variable you found? What does it do? In the second step, they will read the code for a ‘Variables’ project. Then, they will explore what happens in the project.	In this phase, participants will first create a remixing activity for a ‘Variables’ code by changing three things in the given project. In the second step, they will create a short project by using the ‘Variables' code.	This phase will be for participants’ reflection. First, they will give an example of one or two things they did well in their short project. Second, they will answer this question: What changes would you make if you coded this project again? Finally, they will share their responses with other participants in the course.
#10	Lists	Participants will first explore some projects about ‘Lists' code. Also, they will explore and find *some list blocks and read*	In this phase, participants will first create a remixing activity for a ‘Lists’ code by changing three	This phase will be for participants’ reflection. First, they will give an example of one or two things they did well in their

	the code around them to see what they do. After they explore these code blocks, they will answer this question: What list blocks do you see in the project? What do they do? In the second step, they will read the code for a ‘Lists’ project. Then, they will explore what happens in the project.	things in the given project. In the second step, they will create a short project by using the ‘Lists' code.	short project. Second, they will answer this question: What changes would you make if you coded this project again? Finally, they will share their responses with other participants in the course.

APPENDIX G

TRANSLATED QUOTES

Q1. Experimental Group\Participant3	
Bu etkinliklerdeki özellikle fikir alışverişleri, yaptığımız projeleri ilk başta benim hayal ettiğim yerden çok daha ileri bir yere taşıyabiliyor. Imm.. Benim aklımda bir fikir varsa partnerimin yaptığı eklemeler projeyi daha da eğlenceli, görsel olarak daha güzel veya daha verimli bir hale getirdi	The exchange of ideas in these activities can take our projects much further than I initially imagined. Um... If I have an idea, the additions by my pair have made the project even more fun, more visually beautiful or more efficient.
Q2. Experimental Group\Participant2	
İkili kodlama bence çok verimli oldu çünkü tek başıma çalıştığımda her hafta yeni bir fikir bulmak zor oluyor. Partnerim ve ben projeler için olması veya olmaması gereken kısımlarda ortak fikir yürütüyorduk. Birbirimizin fikirlerini değerlendirmiş oluyorduk aslında. Bu yüzden ben ikili kodlamanın daha verimli olduğunu düşünüyorum.	I think pair programming has been very efficient because it's hard to come up with a new idea every week when I'm working alone. My pair and I had a consensus on what should or should not be a part of projects. We were evaluating each other's ideas. So, I think pair programming is more efficient.
Q3. Experimental Group\Participant1	
[Etkinlikleri] grup arkadaşımla birlikte yaptım. Benim için projeleri yapmak bu şekilde daha kolay oldu. Çünkü labda bu projeleri nasıl yapacağımızı konuşuyorduk. Eğer tek başıma yapsaydım zorlanacağımı düşünüyorum. Fikir alışverişi yaparak nasıl bir proje üreteceğimi düşünmemiz, proje üzerine konuşmamız, hikaye oluşturmamız, grup arkadaşımla çok daha kolay oldu.	I did [the activities] with my pair. It was easier for me to do projects this way. Because we were talking about how to do these projects in the lab sessions. I think it would be difficult if I did it alone. It was much easier for me and my pair to think about how to produce a project by exchanging ideas, to talk about the project, to create a story.
Q4. Experimental Group\Participant1	
Ben genelde grup çalışmalarını sevmem fakat sanırım grup arkadaşım ile proje yükünü ortak paylaştığımız için, her derste fikir alışverişi yaptığımız için ve çalışmanın yükü tek bir kişiye kalmadığı için bu derste grup çalışmasını sevdim. Benim için konuşarak sürekli iletişimde kaldığımız için grup çalışması daha kolay oldu. Birimiz ekran paylaşımı yapıyordu, birimiz konuşuyordu. Görevleri bölüşerek yapabiliyorduk. İletişimimiz bu yüzden partnerimle gayet iyiydi.	I don't usually like group work, but I think I liked group work in this course because I shared the burden of the project with my pair, we exchanged ideas in every session, and the burden of work was not left to a single person. For me, it was easier to work in a group because we kept in constant communication. One of us was sharing screens, one of us was controlling coding. We were doing the tasks by dividing them up. That's why our

	communication was very good with my pair.
Q5. Experimental Group\Participant3	
İkili kodlamanın artı özellikleri, yeni bir bakış açısının daha olması. Ve ayrıca iletişim yeteneklerini geliştirme fırsatını bulmam oldu.	The perk of pair programming is that it brings a new perspective. And I also had the opportunity to improve my communication skills.
Q6. Experimental Group\Participant3	
Partnerim bana gerçekten etkili bir iletişim sağladı. Her zaman da projeleri birlikte tartışarak yapabildik. Iımm… o da yeni fikirlere çok açıktı ve ben de onun fikirlerini hep dinlemek istedim. Kısıtlı bir zamanda birbirimizin fikirlerine eklemeler yaparak ortaya güzel bir şeyler çıkarmaya çalıştık. Bence genel olarak güzeldi ve etkiliydi.	My partner and I established good communication. We have always been able to discuss projects together. Um… she was also very open to new ideas, and I always wanted to listen to her ideas. In a limited time, we tried to create something sound by contributing to each other's ideas. Overall, I think it was nice and effective.
Q7. Experimental Group\Participant2	
Mesela, [kodlamaları] yaparken bir hata olduğunda fark etmek zor olabiliyor. Scratch'te eğer bir hata olursa, baştan almak gerekiyor. Ama o sırada [kodlamayı] yaparken birisi ekran paylaşıp diğeri de onu izlediğinde hatayı fark etmek çok daha kolay oluyor.	For example, it can be difficult to notice when there is a mistake when [coding]. If something goes wrong with Scratch, you have to start over. But it's much easier to spot the error when one person is sharing a screen and the other is controlling it while [coding].
Q8. Control Group\Participant8	
Kodlama yaparken sürekli bir şey üretmeye çalışıyoruz, hatayı bulmaya çalışıyoruz ve o hatayı düzeltmeye çalışıyoruz. Bence bu durum oldukça beyin açıcı. Ben bireysel olarak çalıştığım için yaptığım hataları açıkça göremiyorum. Başkasına soramadığım için size de danışıyoruz ama kendi yaşıtlarımızla daha çok tartışıp onlardan yararlanmamız daha faydalı. Nerede sorun olabileceğine dair tek bir kişi olarak baktığım için problemler görünmüyor bazen.	We are constantly trying to produce something, trying to find the error and trying to fix that error while coding. I think this is mind-opening. Whenever I work individually, I cannot clearly see the mistakes I have made. Since I can't ask anyone else -- we also consult you -- but it is more beneficial for us to discuss and benefit from our peers more. Sometimes I cannot see the problem because I look at where there might be a problem as a single person.
Q9. Experimental Group\Participant7	
[Kodlama etkinlikleri sırasında] takıldığım çok problem oldu. Örneğin, hareket etmesini istediğim arka ekran istediğim şekilde başlamıyordu ve baştan gelmiyordu. Görüntüde absürt bir durum vardı ve hiçbir şekilde çözemiyordum.	I had a lot of problems [during coding activities]. For example, the backdrop I wanted to move didn't start the way I wanted it to, and it didn't move back to its original position. There was an absurd situation in the image, and I could not solve it in any way.
Q10. Control Group\Participant4	
Ben bu kodlar üzerine bireysel olarak çalışırken, zorlandığımı düşünüyorum.	When I work on these codes individually, I find it difficult. As I

Daha öncesinde bahsettiğim gibi, başka birine sorup başka bir bakış açısı görmek daha iyi oluyor diye düşünüyorum. Ama bireysel olarak çalıştığımızda bu bakış açısını kendimiz değiştirmek zorundayız.	mentioned earlier, I think it's better to ask someone else and get a different perspective. But when we work individually, we have to change our perspective ourselves.
Q11. Experimental Group\Participant3	
Bir çözüme ulaşmak için yanında fikir alabileceğin birinin olması her zaman daha iyi oluyor bence.	For finding a solution, it's always better to have someone with you from whom you can get ideas.
Q12. Experimental Group\Participant3	
Dediğim gibi tek başımayken bazen problemin bile nerden kaynaklandığını belirlemekte zorluk çekebiliyorum. Iıı.. çünkü bazı kodlar çok karmaşık oluyor. Tam olarak nerede hata olduğunu göremiyoruz. Başka bir bakış açısına sahip olmak yanında kolaylaştırıyor işimi. Çoğu zaman da iki kişi olduğumuz için çözüyoruz.	As I said, I sometimes have difficulty determining where the problem comes from when I'm alone. Uh, because some of the codes are too complicated. We can't see exactly where the error is. Having a different perspective makes my task easier. Most of the time, we have solved [the problem] because there were two of us.
Q13. Experimental Group\Participant4	
Bana göre grup kodlamasının en büyük artısı, bir sorun ile karşılaştığında ve eğer bir yanıt bulamazsan ikinci bir kişi daha farklı o soruna daha farklı yaklaşabilir. Böylelikle o soruna daha hızlı çözüm bulabilirsin.	In my opinion, the biggest advantage of pair programming is that when you encounter a problem and if you cannot find an answer, a second person can approach that problem differently. That way, you can find a quicker solution to that problem.
Q14. Experimental Group\Participant3	
[Eşli kodlama] kodlama başarımı kesinlikle daha da arttırdı. Çünkü dediğim gibi benim aklıma fikir gelmiyor ve nerden başlamam gerektiğini bilemiyorum gibi problemle yaşayabiliyorum.Ya da bir noktada [projeye] bir özellik eklemek istiyorum. Ama nasıl yapacağıma dair fikri partnerim veriyor. O yüzden başarımı kesinlikle arttırdı. Tek başına yapamayacağım şeyleri yapabildiğimi hissediyorum.	[Pair programming] definitely increased my coding achievement even more. Because, as I said, I encounter problems. For example, I can't think of an idea and I don't know where to start, or I want to add a feature [to the project] at some point. But my partner gives me an idea of how to do it. So, it definitely increased my achievement. I feel like I could do things I couldn't do alone.
Q15. Experimental Group\Participant1	
Grup arkadaşımla birlikteyken onunla birlikte fikir yürüttüğümüzde, onun düşündüklerini benim düşünemediğim oluyordu. Ve kodlamayı bu şekilde yapabilirdik diye ikna oluyordum. Kodlama başarımı iyi yönde etkiledi genel olarak, nasıl ifade edeceğimi	When I was with my pair and we were brainstorming, I couldn't come up with ideas she had. And I was convinced that we could code this way. It has affected my coding achievement in an effective way. The most important thing for me was the exchange of ideas in group work.

bilmiyorum. Benim için en önemli şey grup çalışmasındaki fikir alışverişiydi	
Q16. Experimental Group\Participant2	
İkili çalışmanın kodlama başarımı da arttırdığını düşünüyorum. Ben partnerimden çok şey öğrendim kodlama konusunda	I think that working in pairs also increases my coding achievement. I learned a lot about coding from my pair.
Q17. Experimental Group\Participant4	
Ben arkadaşımla birlikte çalışmanın kodlama başarımı iyi yönde etkilediğini düşünüyorum, çünkü daha dönemin en başındayken ben kodlama yapmaktan korkuyordum. [Kodlama] gerçekten hiç bilmediğim bir şeydi. Lablarda yaptığımız son proje ile ilk yaptığımız proje arasında bayağı bir gelişme vardı. Yaptığımız her hafta projelerimizin geliştiğini gözlemledik. Bunları da görmek hem benim motivasyonumu arttırıyor hem de mutlu ediyordu.	I think that working with a pair had a positive effect on my coding success, because at the very beginning of the semester I was afraid of coding. [Coding] was something I never really knew. There was a lot of development between the last project we did in the labs and the first project we did. We observed the development of our projects every week we did. Seeing them both increased my motivation and made me happy.
Q18. Control Group\Participant5	
Ben bu dönem yaratıcılığımı hiç olmadığı kadar kullandığımı düşünüyorum. Çünkü her hafta yeni bir konseptle, basit düzeyde olsa da yeni bir proje yapmaya çalıştık. Her bir proeye baktığımda arkadaşlarımla hepimiz farklı şeyler yapmıştık. Bu da aslında yaratıcılığımızı gösteriyor.	I think I've used my creativity more than ever, because every week we tried to do a new project with a new concept, albeit at a simple level. When I looked at each project, my classmates and I all did different projects. It shows our creativity.
Q19. Experimental Group\Participant2	
Ben ve partnerim kendi aklımızdaki fikirleri bu kod bloğunu kullanarak Scratch'e nasıl aktarabiliriz diye düşünerek ve sürekli bir yeni hikaye oluşturma çabasındaydık. Yani sadece kodu aktarmak yetmiyordu çünkü o kodu bir hikayenin içinde anlamlı bir hale getirmemiz gerekiyordu. Aslında sürekli bir şekilde yaratıcılığımızı kullandığımız bir serüvendi.	My pair and I were constantly thinking about how to transfer ideas from our minds to Scratch using this code block, and we were constantly trying to create a new story. So, it wasn't enough just to transfer the code because we had to embed a meaningful story in that code. In fact, constantly using our creativity was an adventure.

REFERENCES

Aho, A. V. (2012b). Computation and computational thinking. *The Computer Journal, 55*(7), 832–835. https://doi.org/10.1093/comjnl/bxs074

Ambler, S. W., & Lines, M. (2012). *Disciplined agile delivery: A practitioner's guide to agile software delivery in the enterprise*. Indianapolis, IN, USA: IBM Press.

Ayub, M., Karnalim, O., Risal, L., & Wijanto, M. C. (2020). The impact of pair programming on the performance of slow-paced students: a study on data structure courses. *Journal of Information and Organizational Sciences, 44*(2), 211-229. https://doi.org/10.31341/jios.44.2.1

Bailey, R., & Mentz, E. (2017). The value of pair programming in the IT classroom. *The Independent Journal of Teaching and Learning, 12*(1), 90-103.

Beasley, Z. J., & Johnson, A. R. (2022). The impact of remote pair programming in an upper-level CS Course. In *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 1* (pp. 235-240). Dublin, Ireland. https://doi.org/10.1145/3502718.3524772

Beck, L., & Chizhik, A. (2013). Cooperative learning instructional methods for CS1: Design, implementation, and evaluation. *ACM Transactions on Computing Education (TOCE), 13*(3), 1-21. https://doi.org/10.1145/2492686

Berenson, S. B., Slaten, K. M., Williams, L., & Ho, C. W. (2004). Voices of women in a software engineering course: reflections on collaboration. *Journal on Educational Resources in Computing (JERIC), 4*(1), 3. https://doi:10.1145/1060071.1060074

Braught, G., Eby, L. M., & Wahls, T. (2008). The effects of pair-programming on individual programming skill. *SIGCSE Bulletin, 40*(1), 200–204. https://doi.org/10.1145/1352322.1352207

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American Educational Research Association,* (pp. 1–25). Vancouver, Canada. Retrieved from http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

Buitrago Flórez, F., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., & Danies, G. (2017). Changing a generation's way of thinking: teaching computational thinking through programming. *Review of Educational Research, 87*(4), 834–860. https://doi.org/10.3102/0034654317710096

Burden, K., Aubusson, P., Brindley, S., & Schuck, S. (2016). Changing knowledge, changing technology: implications for teacher education futures. *Journal of Education for Teaching, 42*(1), 4-16. https://doi.org/10.1080/02607476.2015.1125432

Chao, J., & Atli, G. (2006). Critical personality traits in successful pair programming. In *Agile 2006,* (pp. 5-93). Minnepolis, USA. https://doi.org/10.1109/agile.2006.20

Chiu, M. M. (2008). Flowing toward correct contributions during group problem solving: a statistical discourse analysis. *The Journal of the Learning Sciences, 17*(3), 415-463. https://doi.org/10.1080/10508400802224830

Clements, D. H. (1995). Teaching creativity with computers. *Educational Psychology Review, 7*(2), 141–161. https://doi.org/10.1007/bf02212491

Creative Computing Lab (2021). *Getting unstuck*. Retrieved from Harvard Graduate School of Education website: https://gettingunstuck.gse.harvard.edu/

Creswell, J. W., & Plano Clark, V. L. (2018). *Designing and conducting mixed methods research*. (3rd ed.). Thousand Oaks, CA: Sage

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., & Woollard, J. (2015). *Computational thinking - A guide for teachers*. Retrieved from https://eprints.soton.ac.uk/424545/1/150818_Computational_Thinking_1_.pdf

Csizmadia, A., Standl, B., & Waite, J. (2019). Integrating the constructionist learning theory with computational thinking classroom activities. *Informatics in Education, 18*(1), 41-67.

Çakır, R., Korkmaz, Ö., İdil, Ö., & Erdoğmuş, F. U. (2021). The effect of robotic coding education on preschoolers' problem solving and creative thinking skills. *Thinking Skills and Creativity, 40,* 100812. https://doi.org/10.1016/j.tsc.2021.100812

Çoklar, A. N., & Akçay, A. (2018). Evaluating programming self-efficacy in the context of inquiry skills and problem-solving skills: A perspective from teacher education. *World Journal on Educational Technology, 10*(3), 153–164. https://doi.org/10.18844/wjet.v10i3.3556

Da Silva Estácio, B. J., & Prikladnicki, R. (2015). Distributed pair programming: A systematic literature review. *Information and Software Technology, 63,* 1–10. https://doi.org/10.1016/j.infsof.2015.02.011

Dellas, M., & Gaier, E. L. (1970). Identification of creativity: The individual. *Psychological Bulletin, 73*(1), 55–73. https://doi:10.1037/h0028446

Demir, Ö., & Seferoğlu, S. S. (2019). Developing a Scratch-based coding achievement test. *Information and Learning Sciences, 120*(5/6), 383–406. https://doi.org/10.1108/ILS-08- 2018-0078

Demir, Ö., & Seferoglu, S. S. (2020). A comparison of solo and pair programming in terms of flow experience, coding quality, and coding achievement. *Journal of Educational Computing Research, 58*(8), 1448–1466. https://doi.org/10.1177/0735633120949788

Denner, J., & Werner, L. (2007). Computer programming in middle school: how pairs respond to challenges. *Journal of Educational Computing Research, 37*(2), 131–150. https://doi.org/10.2190/12t6-41l2-6765-g3t2

Denner, J., Werner, L., Campe, S., & Ortiz, E. (2014). Pair programming: under what conditions is it advantageous for middle school students? *Journal of Research on Technology in Education, 46*(3), 277–296. https://doi.org/10.1080/15391523.2014.888272

Denning, P. J. (2009). The profession of IT Beyond computational thinking. *Communications of the ACM, 52*(6), 28–30. https://doi.org/10.1145/1516046.1516054

Denning, P. J. (2017). Remaining trouble spots with computational thinking. *Communications of the ACM, 60*(6), 33–39. https://doi.org/10.1145/2998438

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.

Doleck, T., Bazelais, P., Lemay, D. J., Saxena, A., & Basnet, R. B. (2017). Algorithmic thinking, cooperativity, creativity, critical thinking, and problem solving: exploring the relationship between computational thinking skills and academic performance. *Journal of Computers in Education, 4*(4), 355 369. https://doi.org/10.1007/s40692-017-0090-9

Dongo, T. A., Reed, A. H., & T O'Hara, M. (2016). Exploring pair programming benefits for MISM majors. *Journal of Information Technology Education. Innovations in Practice, 15*, 223–239. https://doi.org/10.28945/3625

Dyba, T., Dingsoyr, T., & Hanssen, G. K. (2007). Applying systematic reviews to diverse study types: An experience report. In *First International Symposium on Empirical Software Engineering and Measurement (ESEM 2007)* (pp. 225-234). Madrid, Spain: IEEE.

Emurian, H. H., Holden, H. K., & Abarbanel, R. A. (2008). Managing programmed instruction and collaborative peer tutoring in the classroom: Applications in teaching Java™. *Computers in Human Behavior, 24*(2), 576 614. https://doi.org/10.1016/j.chb.2007.02.007

Fagerlund, J., Häkkinen, P., Vesisenaho, M., & Viiri, J. (2021). Computational thinking in programming with Scratch in primary schools: A systematic review. *Computer Applications in Engineering Education, 29*(1), 12–28. https://doi.org/10.1002/cae.22255

Faja, S. (2014). Evaluating effectiveness of pair programming as a teaching tool in programming courses. *Information Systems Education Journal, 12*(6), 36–44.

Fanchamps, N. L. J. A., Slangen, L., Hennissen, P., & Specht, M. (2019). The influence of SRA programming on algorithmic thinking and self-efficacy using Lego robotics in two types of instruction. *International Journal of Technology and Design Education, 31*(2), 203–222. https://doi.org/10.1007/s10798-019-09559-9

Fisher, L., Byrne, J. R., & Tangney, B. (2016). Teacher experiences of learning computing using a 21st century model of computer science continuing professional development. In *International Conference on Computer Supported Education (2)* (pp. 273-280). Dublin, Ireland: University of Dublin

Ghorashi, S., & Jensen, C. (2017). Integrating collaborative and live coding for distance education. *IEEE Computer, 50*(5), 27–35. https://doi.org/10.1109/mc.2017.131

Gómez, O. S., Aguileta, A. A., Aguilar, R., Ucán, J. P., Rosero, R. H., & Cortes-Verdin, K. (2017). An empirical study on the ımpact of an IDE tool support in the pair and solo programming. *IEEE Access, 5*, 9175–9187. https://doi.org/10.1109/access.2017.2701339

Greene, J. C. (2007). *Mixed Methods in Social Inquiry*. San Francisco: Jossey-Bass. *Journal of Mixed Methods Research, 2*(2), 190–192. https://doi.org/10.1177/1558689807314013

Grover, S., & Pea, R. (2013). Computational thinking in K–12. *Educational Researcher, 42*(1), 38–43. https://doi.org/10.3102/0013189x12463051

Hanks, B. (2008). Empirical evaluation of distributed pair programming. *International Journal of Human-computer Studies, 66*(7), 530–544. https://doi.org/10.1016/j.ijhcs.2007.10.003

Haseski, H. İ., Ilic, U., & Tugtekin, U. (2018). Defining a new 21st century skill-computational thinking: Concepts and trends. *International Education Studies, 11*(4), 29–42. https://doi.org/10.5539/ies.v11n4p29

Hensley, N. (2020). Educating for sustainable development: Cultivating creativity through mindfulness. *Journal of Cleaner Production, 243*, 118542. https://doi.org/10.1016/j.jclepro.2019.118542

Hesse, F., Care, E., Buder, J., Sassenberg, K., & Griffin, P. (2015). A framework for teachable collaborative problem solving skills. In *Assessment and teaching of 21st century skills* (pp. 37-56). Springer, Dordrecht. https://doi.org/10.1007/978-94-017-9395-7_2

Hwang, W. Y., Shadiev, R., Wang, C. Y., & Huang, Z. H. (2012). A pilot study of cooperative programming learning behavior and its relationship with students' learning performance. *Computers & Education, 58*(4), 1267-1281. https://doi.org/10.1016/j.compedu.2011.12.009

Ilic, U. (2021). The impact of scratch-assisted instruction on computational thinking (CT) skills of pre-service teachers. *International Journal of Research in Education and Science, 7*(2), 426–444. https://doi.org/10.46328/ijres.1075

Instefjord, E. J., & Munthe, E. (2017). Educating digitally competent teachers: A study of integration of professional digital competence in teacher education. *Teaching and Teacher Education, 67,* 37–45. https://doi.org/10.1016/j.tate.2017.05.016

Iskrenovic-Momcilovic, O. (2019). Pair programming with scratch. *Education and Information Technologies*, *24*(5), 2943–2952. https://doi.org/10.1007/s10639-019-09905-3

Isong, B., Moemi, T., Dladlu, N., Motlhabane, N., Ifeoma, O., & Gasela, N. (2016). Empirical confirmation of pair programming effectiveness in the teaching of computer programming. In *2016 International Conference on Computational Science and Computational Intelligence (CSCI)* (pp. 276-281). Las Vegas, NV, USA: IEEE. https://doi.org/10.1109/csci.2016.0060

ISTE (2015). *ISTE standards for computational thinking*. Retrieved from https://www.iste.org/standards/iste-standards-for-computational-thinking

ISTE (2016). *ISTE standards for students*. Retrieved from https://www.iste.org/standards/standards/for-students-2016

ISTE & CSTA (2011). Operational definition of computational thinking for K-12 education. Retrieved from https://cdn.iste.org/www-root/Computational_Thinking_Operational_Definition_ISTE.pdf

Kalelioglu, F., Gülbahar, Y., & Kukul, V. (2016). A framework for computational thinking based on a systematic research review. *Baltic Journal of Modern Computing*, *4*(3), 583.

Karaoğlu, H. (2018). *The influence of pair-programming technique on secondary school students' confidence and achievement in computer programming* (Master dissertation). Retrieved from https://tez.yok.gov.tr/UlusalTezMerkezi/tezSorguSonucYeni.jsp (Order No: 503649).

Kazimoglu, C., Kiernan, M., Bacon, L., & MacKinnon, L. (2012). Learning programming at the computational thinking level via digital game-play. *Procedia Computer Science*, *9*, 522-531. https://doi.org/10.1016/j.procs.2012.04.056

Khudhur, O.M. (2022). *The Effect of Pair Programming On The Understandability Of Flowcharts: A Case Study In A C Programming Course* (Master dissertation). Retrieved from https://tez.yok.gov.tr/UlusalTezMerkezi/tezSorguSonucYeni.jsp (Order No: 723892).

Kim, S., Chung, K., & Yu, H. (2013). Enhancing digital fluency through a training program for creative problem solving using computer programming. *Journal of Creative Behavior*, *47*(3), 171–199. https://doi.org/10.1002/jocb.30

Kiss, G., & Arki, Z. (2017). The influence of game-based programming education on the algorithmic thinking. *Procedia-Social and Behavioral Sciences*, *237*, 613–617. https://doi.org/10.1016/j.sbspro.2017.02.020

Koray, Ö., & Azar, A. (2008). Ortaöğretim öğrencilerinin problem çözme ve mantiksal düşünme becerilerinin cinsiyet ve seçilen alan açisindan incelenmesi. *Kastamonu Eğitim Dergisi*, *16*(1), 125-136.

Korkmaz, Ö., Çakir, R., & Özden, M. Y. (2017). A validity and reliability study of the computational thinking scales (CTS). *Computers in Human Behavior, 72*, 558–569. https://doi.org/10.1016/j.chb.2017.01.005

Kotluk, N., & Kocakaya, S. (2015). Digital storytelling for developing 21st century skills: From high school students' point of view. *Journal of Research in Education and Teaching*, *4*(2), 354-363.

Latané, B., Williams, K., & Harkins, S. (1979). Many hands make light the work: The causes and consequences of social loafing. *Journal of Personality and Social Psychology, 37*(6), 822–832. https://doi.org/10.1037/0022-3514.37.6.822

Lewis, C. M. (2011). Is pair programming more effective than other forms of collaboration for young students? *Computer Science Education, 21*(2), 105–134. https://doi.org/10.1080/08993408.2011.579805

Li, L., Da Xu, L., He, Y., He, W., Pribesh, S., Watson, S. M. R., & Major, D. A. (2021). Facilitating online learning via zoom breakout room technology: A case of pair programming involving students with learning disabilities. *Communications of the Association for Information Systems, 48*, 88–92. https://doi.org/10.17705/1cais.04812

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior, 41*, 51–61. https://doi.org/10.1016/j.chb.2014.09.012

Lyon, J. A., & J. Magana, A. (2020). Computational thinking in higher education: A review of the literature. *Computer Applications in Engineering Education*, *28*(5), 1174–1189. https://doi.org/10.1002/cae.22295

Maguire, P., Maguire, R., Hyland, P., & Marshall, P. (2014). Enhancing collaborative learning using pair programming: Who benefits? *AISHE-J: The All Ireland Journal of Teaching and Learning in Higher Education*, *6*(2), 14111-14124.

Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B., & Resnick, M. (2004). Scratch: a sneak preview [education]. In *Proceedings. Second International Conference on Creating, Connecting and Collaborating through Computing, 2004.* (pp. 104-109). Kyoto, Japan: IEEE.

McDowell, C., Werner, L., Bullock, H., & Fernald, J. (2002). The effects of pair-programming on performance in an introductory programming course. In *Proceedings of the 33rd SIGCSE technical symposium on Computer science education* (pp. 38-42). Cincinnati, KY, USA. https://doi.org/10.1145/563340.563353

Mentz, E., Van Der Walt, J. P., & Goosen, L. (2008). The effect of incorporating cooperative learning principles in pair programming for student teachers. *Computer Science Education, 18*(4), 247–260. https://doi.org/10.1080/08993400802461396

MIT Media Lab. (2008). *Scratch*. Retrieved from http://scratch.mit.edu

National Research Council. (2010). *Report of a workshop on the scope and nature of computational thinking*. National Academies Press.

National Research Council. (2011). *Report of a workshop of pedagogical aspects of computational thinking*. Retrieved from http://www.nap.edu/catalog.php?record_id=13170

Nawahdah, M., & Taji, D. (2016, April). Investigating students' behavior and code quality when applying pair-programming as a teaching technique in a Middle Eastern society. In *2016 IEEE Global Engineering Education Conference (EDUCON)* (pp. 32-39). Abu Dhabi, United Arab Emirates. https://doi.org/10.1109/educon.2016.7474527

Papadakis, S., Kalogiannakis, M., Zaranis, N., & Orfanakis, V. (2016). Using Scratch and App Inventor for teaching introductory programming in secondary education. A case study. *International Journal of Technology Enhanced Learning, 8*(3/4), 217-233. https://doi.org/10.1504/ijtel.2016.082317

Papert, S. (1996). An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning, 1*(1), 95-123.

Passey, D. (2017). Computer science (CS) in the compulsory education curriculum: Implications for future research. *Education and Information Technologies, 22*(2), 421–443. https://doi.org/10.1007/s10639-016-9475-z

Peppler, K., & Kafai, Y. (2005). Creative coding: Programming for personal expression. *Retrieved August, 30*(2008), 314.

Psycharis, S., & Kallia, M. (2017). The effects of computer programming on high school students' reasoning skills and mathematical self-efficacy and problem solving. *Instructional Science, 45*(5), 583-602.

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.

Rizvi, B., Bagheri, E., & Gasevic, D. (2015). A systematic review of distributed Agile software engineering. *Journal of Software, 27*(10), 723–762. https://doi.org/10.1002/smr.1718

Ross, K. A. (1998). Doing and proving: the place of algorithms and proofs in school mathematics. *American Mathematical Monthly*, *105*(3), 252–255. https://doi.org/10.1080/00029890.1998.12004875

Saeli, M., Perrenet, J. J., Jochems, W. W., & Zwaneveld, B. (2011). Teaching Programming in Secondary School: A Pedagogical Content Knowledge Perspective. *Informatics in Education*, *10*(1), 73–88. https://doi.org/10.15388/infedu.2011.06

Saldaña, J. (2021). *The coding manual for qualitative researchers*. Sage.

Salge, C. A. D. L., & Berente, N. (2016). Pair programming vs. solo programming: What do we know after 15 years of research? In *2016 49th hawaii international conference on system sciences (hicss)* (pp. 5398-5406). NW Washington, DC, USA: IEEE.

Scherer, R., Siddiq, F., & Sánchez Viveros, B. (2019). The cognitive benefits of learning computer programming: A meta-analysis of transfer effects. *Journal of Educational Psychology, 111*(5), 764–792. https://doi.org/10.1037/edu0000314

Selby, C. C. (2014). *How can the teaching of programming be used to enhance computational thinking skills?* (Doctoral dissertation, University of Southampton). Retrieved from https://eprints.soton.ac.uk/366256/1/soton.ac.uk_ude_personalfiles_users_al4_mydesktop_Selby_cc_New_Amendments.pdf

Seo, Y. H., & Kim, J. H. (2016). Analyzing the effects of coding education through pair programming for the computational thinking and creativity of elementary school students. *Indian Journal of Science and Technology*, *9*(46), 1-5. https://doi.org/10.17485/ijst/2016/v9i46/107837

Shin, S., & Park, P. (2014). A study on the effect affecting problem solving ability of primary students through the scratch programming. *Advanced Science and Technology Letters*, *59*(1), 117-120. https://doi.org/10.14257/astl.2014.59.27

Slavin, R. E. (1980). Cooperative learning. *Review of Educational Research*, *50*(2), 315–342. https://doi.org/10.3102/00346543050002315

Snalune, P. (2015). The benefits of computational thinking. *ITNOW, 57*(4), 58–59. https://doi.org/10.1093/itnow/bwv111

Teague, D., & Lister, R. (2014). Programming: reading, writing and reversing. In *Proceedings of the 2014 conference on Innovation & technology in computer science education* (pp. 285-290). Uppsala, Sweden.

Thompson, B. (2008). *Foundations of behavioral statistics: An insight-based approach* (1st ed.). New York: Guilford.

Tikva, C., & Tambouris, E. (2021). Mapping computational thinking through programming in K-12 education: A conceptual model based on a systematic literature Review. *Computers & Education*, *162*, 104083.

Turchi, T., Fogli, D., & Malizia, A. (2019). Fostering computational thinking through collaborative game-based learning. *Multimedia Tools and Applications*, *78*(10), 13649–13673. https://doi.org/10.1007/s11042-019-7229-9

Vanhanen, J., & Korpi, H. (2007). Experiences of using pair programming in an agile project. In *2007 40th Annual Hawaii International Conference on System Sciences (HICSS'07)* (pp. 274b-274b). Big Island, Hawaii: IEEE.

Veenman, S., Van Benthum, N., Bootsma, D., Van Dieren, J., & Van Der Kemp, N. (2002). Cooperative learning and teacher education. *Teaching and Teacher Education*, *18*(1), 87–103. https://doi.org/10.1016/s0742-051x(01)00052-x

Wei, X., Lin, L., Meng, N., Tan, W., Kong, S. C., & Kinshuk. (2021). The effectiveness of partial pair programming on elementary school students' computational thinking skills and self-efficacy. *Computers & Education*, *160*, 104023. https://doi.org/10.1016/j.compedu.2020.104023

Williams, L., Kessler, R. R., Cunningham, W., & Jeffries, R. (2000). Strengthening the case for pair programming. *IEEE Software*, *17*(4), 19–25. https://doi.org/10.1109/52.854064

Wing, J. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *366*(1881), 3717–3725. https://doi.org/10.1098/rsta.2008.0118

Wray, S. (2010). How pair programming really works. *IEEE software*, *27*(1), 50–55. https://doi.org/10.1109/ms.2009.199

Yadav, A., Stephenson, C., & Hong, H. L. (2017). Computational thinking for teacher education. *Communications of the ACM*, *60*(4), 55–62. https://doi.org/10.1145/2994591

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), 1–16. https://doi.org/10.1145/2576872

Yağcı, M. (2019). A valid and reliable tool for examining computational thinking skills. *Education and Information Technologies*, *24*(1), 929-951.

Yang, W., Ng, D. T. K., & Su, J. (2023). The impact of story-inspired programming on preschool children's computational thinking: A multi-group experiment. *Thinking Skills and Creativity*, *47*, 101218. https://doi.org/10.1016/j.tsc.2022.101218

Zhang, L., & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. *Computers & Education, 141*, 103607. https://doi.org/10.1016/j.compedu.2019.103607

Zhong, B., Wang, Q., & Chen, J. (2016). The impact of social factors on pair programming in a primary school. *Computers in Human Behavior*, *64*, 423–431. https://doi.org/10.1016/j.chb.2016.07.017

Zieris, F., & Prechelt, L. (2014). On knowledge transfer skill in pair programming. In *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement* (pp. 1-10). Torino, Italy.