2021 ASEE ANNUAL CONFERENCE
Virtual Meeting | July 26–29, 2021 | Pacific Daylight Time
Paper ID #34830

Computational Thinking: A Pedagogical Approach Developed to Prepare Students for the Era of Artificial Intelligence

Dr. Gulustan Dogan, University of North Carolina Wilmington

Gulustan Dogan is an assistant professor at University of North Carolina Wilmington in Computer Science department. She worked at Yildiz Technical University, Istanbul, Turkey as an Associate Professor. She worked at NetApp and Intel as a software engineer in Silicon Valley. She received her PhD degree in Computer Science from City University of New York. She received her B.Sc degree in Computer Engineering from Middle East Technical University, Turkey. She is one of the founding members of Turkish Women in Computing (TWIC), a Systers community affiliated with Anita Borg Institute. She also serves as Ambassador of Women In Data Science Stanford.

Dr. Yang Song, University of North Carolina Wilmington
Ms. Damla Surek, Yildiz Technical University

Ms. Damla Surek is a Computer Education and Instructional Technology student in her third year at Yildiz Technical University in Istanbul, Turkey.

Computational Thinking: A Pedagogical Approach Developed to Prepare UNCW Students for the Era of Artificial Intelligence

Gulustan Dogan
dogang@uncw.edu
UNCW - Computer Science Dept.

Yang Song
songy@uncw.edu
UNCW - Computer Science Dept.

Damla Surek
l9118044@std.yildiz.edu.tr
Yildiz Technical University

Abstract

We propose Computational Thinking (CT) as an innovative pedagogical approach with broad application. Research and current industry trends illustrate that students should have a solid computational thinking ability in order to have the skills required for future jobs in Artificial Intelligence. Due to current social issues regarding COVID-19 and natural disasters, we are rapidly moving towards a cyberspace era where many citizens will conduct their work online. Understanding the foundations and tools of computation – e.g., abstraction, decomposition, pattern recognition – is critical for any student to be prepared for the digital AI age. Believing students should be fully prepared for future jobs that involve computation, we developed a CT module on a Learning Management System (LMS). We have collected data of students who took our CT course module. We looked into the students' activity records and analyzed the number of students' views on the pages and the number of participants on each quiz. We counted the total number of engagements of the ten components in the CT course module. Ultimately, we believe that our modules had a greater impact on those students who were newer to computational thinking, over those who had prior experience and were enrolled in upper-level computational courses.

1 Introduction

According to Wing, Computational Thinking (CT) is the thought processes involved in formulating a problem and expressing its solution(s) in such a way that an information processor – human or machine – can effectively carry out that solution [1]. The educational philosophy behind Computational Thinking is that problems in every discipline can be solved by the tools of computation such as algorithmic thinking, decomposition, abstraction, pattern recognition. For instance, one of the pillars of computational thinking is algorithmic thinking. Algorithmic thinking is the use of step-by-step sets of instructions to complete a task. It is proven scientifically

that algorithmic thinking capability helps students process tasks in a step-by-step approach reducing anxiety in learning. In all disciplines, tasks/processes/events can be written in the form of an algorithm. For instance, solubility in chemistry can be introduced to students as an algorithmic flowchart [2]. Especially in disciplines such as physics which rely on equations and data, computational thinking is critical [3].

In addition to developing a rich mental toolkit to draw from computational thinking capabilities, we believe that formal exposure to algorithmic/computational thinking should be required of all undergraduate students to prepare them for the era where most of the job definitions will be drastically changed.

As unfortunate CORONA virus and natural disasters such as Australian wildfires have shown, we are rapidly moving towards a cyber era where most of the tasks will be done online. Understanding the foundations and tools of computation – e.g., abstraction, decomposition, pattern recognition – is a critical tool for any student to be prepared for the digital AI age [4]. Students should be fully prepared for future jobs that involve computation [5]. Moreover, there is a shortage in tech skill force, and if students are introduced to computing in classes of the field they are majoring where they feel confident, they might have higher chances to develop interests in technology. They will be encouraged to go into the interdisciplinary fields without feeling disadvantaged compared to computer science majors. For instance, a student majoring in biology will be more confident pursuing a degree/career in fields like biotechnology, bioengineering which spans technology and biology after being introduced to computational thinking in a biology class [6]. Moreover, there has been a lot of effort by government and research agencies to encourage young people into STEM and technology careers. With this work of ours, we believe University of North Carolina Wilmington (UNCW) will be doing its due diligence in supporting these efforts. With this project, we want to take the initiative to start the CT pedagogical movement in our institution to have graduates with the required applied computational thinking skill set which aligns with the strategic goals of our institution. We believe that all our institution undergraduate students should have some mastery of computational thinking, as they do with physical, mathematical, biological, and chemical thinking, and as they do with the critical thinking embedded in the humanities, arts, and social sciences.

Based on our investigation, such courses are not offered at our institution or in many others. In addition to this, according to our research on department and course webpages, STEM classes in UNCW have not explicitly stated computational thinking as a learning outcome. Our CT pedagogy is innovative for our department and other disciplines. We took the initiative to start the CT pedagogical movement at UNCW in order to produce graduates with required applied computational thinking skill sets. As part of a grant we received, we developed a curriculum in LMS that teaches students computational thinking in high traffic STEM gate-keeping courses. Our one-week curriculum was used in four courses (with data from three courses reported in this paper, the data from the fourth course is jeopardized) in our institution in the spring semester of 2021.

This study aims to analyze the course activity reports of students who took the CT course on LMS to assess the students' learning patterns. Also, as part of this study, the developed CT course will be made available on Canvas Commons for any educator to import to their courses.

2 Related Work

To further solidify the importance of computational thinking (CT) skills in college students, it is crucial to examine additional studies regarding this subject. Hsu et al. proposed that traditionally, the manner in which computational thinking is taught to students has been rather challenging [7]. However, over the last decade, significant improvements have been made to the way that CT is taught. Advancements regarding programming, learning tools, demographics, and applied courses have all contributed to the augmentation of CT in everyday learning for students.

Another work conducted by Gong et al. examines some of the most driving factors in college student's computational thinking skills (CTS) through the use of the "flipped classroom" approach [8]. This research revealed that many college students that were sampled agreed on the use of group learning, learning incentives, and targeted learning all contributed to the quality of CTS skills absorbed by those individuals.

Due to the relatively recent implementation of computation thinking (CT) in higher education, Lyon et al. assert that more solid definitions, methods, and learning strategies of CT must be implemented [9]. The difficulty surrounding what CT means is due to the similarities among mathematical and algorithmic thinking. CT is often defined differently in varying pieces of literature, thus making the task of implementing a concrete learning system difficult in higher education. One of the best ways to help incorporate a CT curriculum, is to look at those that have already been implemented and have proven to be successful in the classroom.

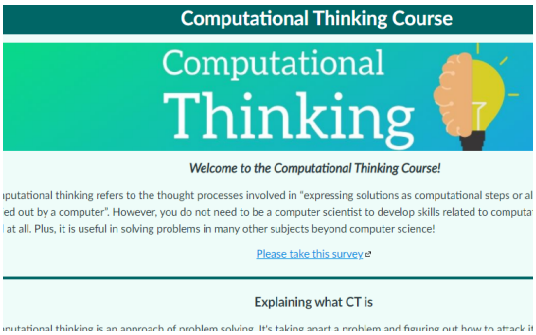
Agbo et. al review the impacts and effects that computational thinking (CT) has had on the programming and critical thinking skills of higher education students [10]. Students that are interested in improving their programming skills and technological expertise often find that through the use of a CT regiment, they become increasingly more proficient in their field. CT is so important in today's society due to the nature of the skills and knowledge that it encompasses to a student. CT builds not only a better communication outlet for a student, but also the application of problem-solving skills to real-world situations. From the research data gathered in this study, one may presume that a student may be expect to be more successful in their computer-related field, given that they are provided a CT approach.

3 Overview of the CT Module

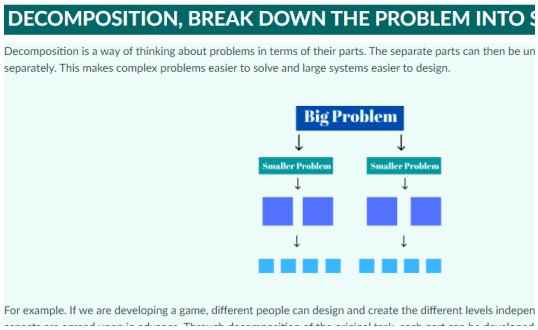
CT course on LMS has four modules namely Decomposition, Abstraction, Pattern Recognition, Algorithms as seen in Figure 1 and 2.

Decomposition is a way of thinking about problems in terms of their parts. The separate parts can then be understood and evaluated separately. This makes complex problems easier to solve and large systems easier to design. The decomposition page in our CT module can be seen in Figure 1b.

Pattern recognition is finding similarities between or within problems. Essential parts of pattern recognition are introduced in Pattern Recognition Module, as seen in Figure 2a. Abstraction is the process of filtering out, or ignoring, the characteristics of patterns that are not needed in order to concentrate on those that are. Abstraction is also introduced as a module, as seen in Figure



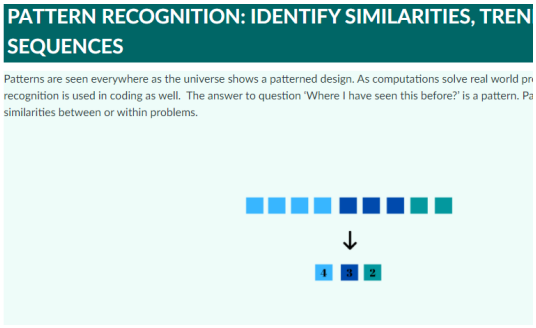
(a) Introduction page



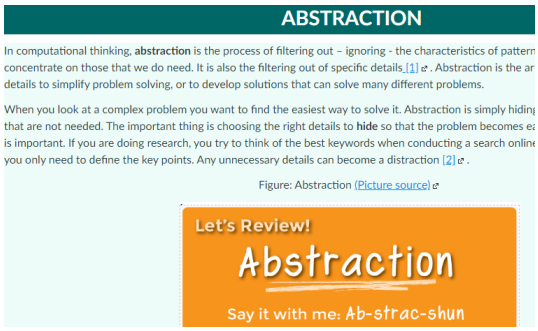
(b) Decomposition page

Figure 1

2b.



(a) Pattern Recognition page



(b) Abstraction page

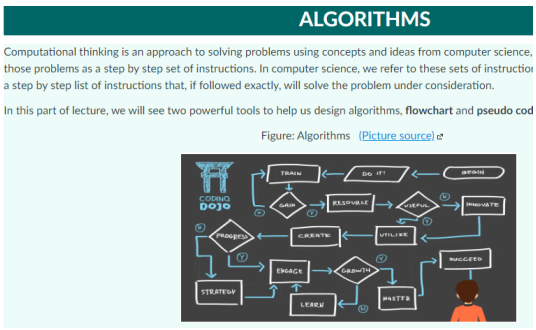
Figure 2

In CT, we refer to a step-by-step set of instructions as algorithms. An algorithm is a step-by-step list of instructions that, if followed exactly, will solve the problem under consideration. As the last module of the course Algorithms is introduced, as seen in Figure 3a. The final page of the course can be seen in Figure 3a.

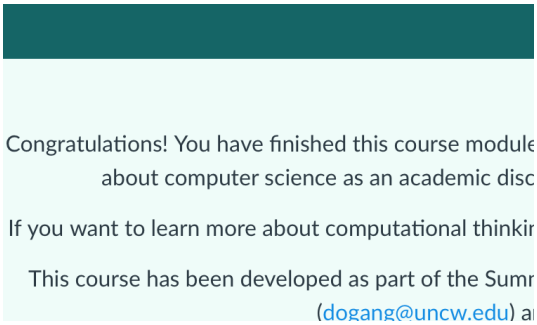
4 Data Collection

In the spring semester of 2021, the CT course modules were presented as additional reading tasks and given to four different courses taught by three faculty members. The courses are all offered by the Department of Computer Science in UNCW, which is an R2 university on the east coast of the U.S. The description of the participating courses are as follows:

- Course 1: a required CS1 course by the computer science major, but also open as a general education course. Roughly half of the students are in computer science major;
- Course 2: a special-topic course on AI open to both undergraduate and graduate students. Students are all major in computer science or closely related majors;
- Course 3: a graduate-level algorithm course also open to undergraduate students; Students are all major in computer science or closely related majors.



(a) Algorithms page



(b) Final page

Figure 3

- Course 4: removed from this study because the data was jeopardized.

The detailed statistics of the participating courses can be found in the Table 1 below. We found there were four students registered for at least two courses listed below. For the students who participated in the CT course module more than once, we dropped the records from the courses that released the course module later in the later data analysis.

Table 1. Information about the four participating courses.

	# of students	Male/Female	# of undergrad.	# of graduate	Major in CS or related subjects	# of students who complete
Course 1	24	16/8	24	0	11	23
Course 2*	14	12/2	8	6	13	13
Course 3*	17	14/3	7	10	17	14

* There are four duplicated students who took both classes.

4.1 Students' participation in each course component

The CT course module has a total of ten components, including six pages and four quizzes. They can be seen in Figure 4. Other than the introduction page and the final page, the four major topics of CT are presented on a page and concluded by a quiz. The sequence of the topics is: decomposition, pattern recognition, abstraction, and algorithm. The screenshot below is an example of how the CT course module is presented to students in an LMS. In all the participating courses, students have limited time, usually around two weeks, to finish this course module (the CT course module is designed to be doable in one week as a reading assignment, we gave two weeks because this was not in the course curricula). For each quiz, the instructors allow students to re-take up to three times.

We collected the students' activity records on the LMS and analyzed the number of students' views on the pages and the number of participants on each quiz. We counted the total number of engagements of the ten components in the CT course module. The students' participation is shown in Figure 5. Each red box in Figure 5 shows two components of the same topic: a page of instruction followed by a quiz.

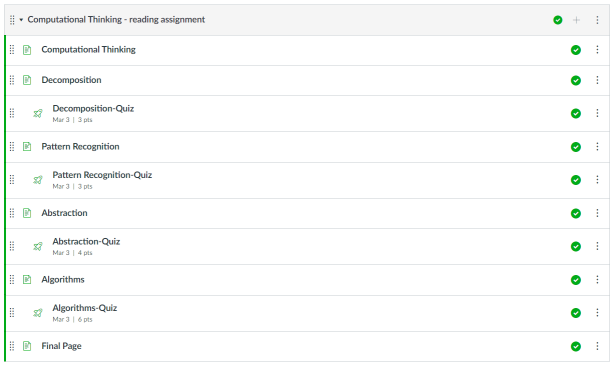


Figure 4: A screenshot of our CT course module in an LMS

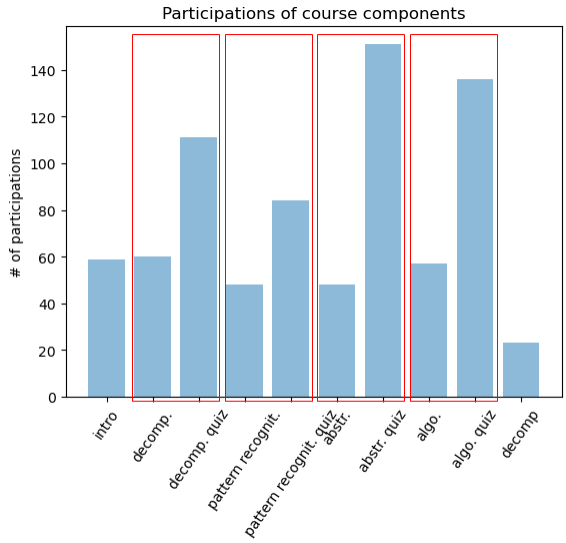


Figure 5: Students' participation in the CT course components

From Figure 5, we found that the page that received the most number of reads was the decomposition, the page of pattern recognition and abstraction received fewer views (48 each), but the number of views was higher on the algorithms page. We believe the reason for this was the number of quiz questions being the highest for the algorithms topic (6 quiz questions). We also noticed that among the four CT topics, students generally tend to read the pages less and participate in the quizzes more. This hints that students are making use of the re-takes to finish the quizzes without spending efficient time reading and comprehending the information provided.

4.2 Time Spent on Quizzes in Minutes

We further investigate the time students spent finishing the quiz questions. The LMS system log provides the timestamp that a student opens and submits a quiz. Therefore we are able to calculate the duration of a student's attempt on a quiz. For each student, we further aggregate the number of minutes on attempting all the quizzes in our CT course module.

We also define the students in Course 1 as the first group. Most of those students are in their first or second year of college, and those courses are very likely to be their first course related to computation. Students in Course 2 and 3 are in the second group. The students in the second group are veteran computer science students who should have already taken at least one programming course.

Figure 6 shows the distribution of the number of minutes students spend on all four quizzes. Blue bars represent students in group 1 and the orange bars represent students in group 2. The width of each bin is 40 minutes. We found that the majority of students spend no more than 80 minutes on the four quizzes. The students in group 2 courses are more likely to complete all four quizzes in less than a total of 40 minutes, and the majority of students in group 1 spend between 40 to 80 minutes on the quizzes. This is expected by us since the students in group 1 are new to computer science, and the students in group 2 have taken computer science-related courses for several semesters.

We also noticed that there is a noticeable number of outliers on the right side of Figure 6. Some students spent more than 1000 minutes on the quizzes. We believe those records are caused by students who opened the quiz but did not try to submit it until hours later.

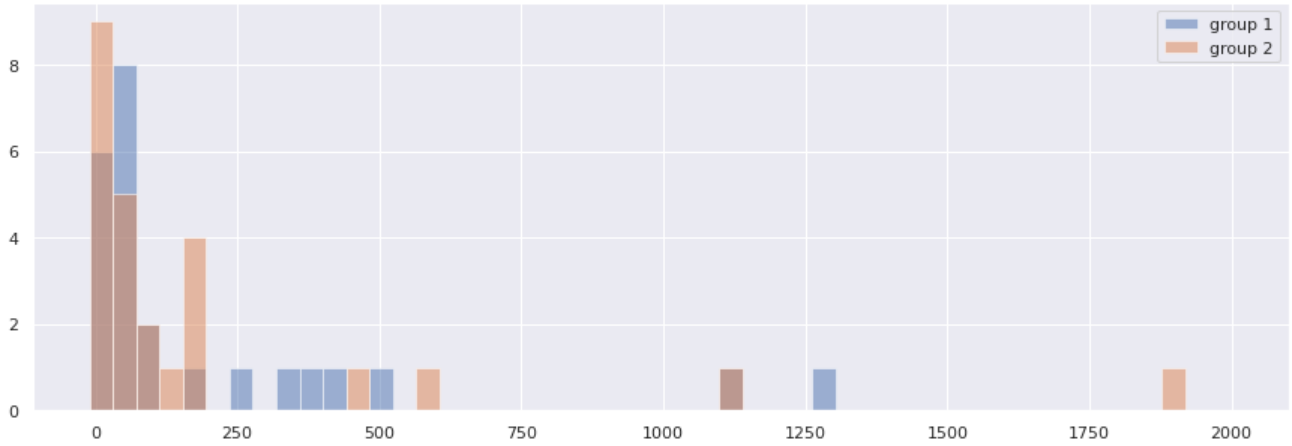


Figure 6: The distribution of the time students spent in finishing all four quizzes

4.3 Number of Page Views

For the first two bins, the students must have jumped into the quizzes without reading the information provided. We also counted the number of views of each student on the six pages. As we noticed that students tend to read less and attempt the quizzes more when they finish the CT course module, we are interested in the number of views by the students in the different course groups.

Figure 7 shows the number of students' views of the pages (for four CT topics, plus the introduction and the final page). The LMS system log only provides the number of views but does not provide the duration that a student stays on a page. The width of each bin is two reads.

From Figure 7 we found that the majority of the students in group 1 courses tend to conduct four

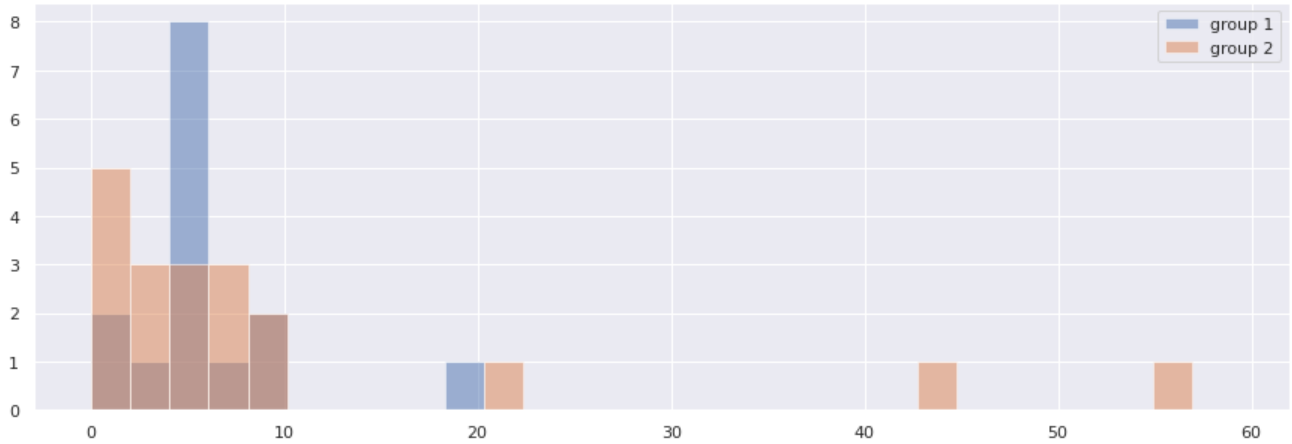


Figure 7: The distribution of students' views on the pages of the CT course module

reads of the information. Considering we expect students to read the pages of four topics before taking the quizzes, students should have at least four reads to finish the CT course module. By contrast, roughly half of the students in the group 2 courses only showed 0-4 views on the course pages. This means that those students are confident enough to skip the reading and attempt the quizzes directly.

5 Conclusion

To conclude, we found students did not read the course component pages due to the option of quiz re-takings. Considering retaking quizzes are allowed in this research, students may find that taking quizzes multiple times may be a more efficient way to learn instead of reading the pages. However, for the more challenging topics such as algorithms, students more often referred to the information pages for aid in answering the following quiz questions. The veteran computer science students were shown to have completed the quizzes quicker, most likely due to their knowledge acquired in earlier computational education. On top of this, the upper-level students were more likely to skip over the information pages given, and immediately take the quizzes. All of the findings indicate that our CT course module challenge junior students to learn CT as a new set of concepts , and help senior students to revisit the CT-related concepts they have already learned.

The limitation of this project is that this project only used students' time-on-task information from different courses. In the future, the authors plan to get students' feedback toward their experience and the perceived usefulness of CT concepts from student survey(s).

To download the CT course module we have designed, please visit:
`https://bit.ly/3bfxohR`.

6 Acknowledgement

We would like to thank all the students and teachers who participated in this project. This material is based upon work supported by the Center of Teaching Excellence Summer Pedagogy Development Award of University of North Carolina Wilmington.

References

[1] J. M. Wing, "Computational thinking," in *2011 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*, 2011, pp. 3–3.

[2] Ibpossum, "Re thinking Chemistry (identifying ions) with Computational Thinking," https://ibpossum.com/2018/06/16/re-thinking-chemistry-identifying-ions-with-computational-thinking/, 2019, [Online; accessed 8-March-2021].

[3] C. Orban and R. Teeling-Smith, "Computational Thinking in Introductory Physics," https://arxiv.org/pdf/1907.08079.pdf, 2019, [Online; accessed 8-March-2021].

[4] V. J. Shute, C. Sun, and J. Asbell-Clarke, "Demystifying computational thinking," *Educational Research Review*, vol. 22, pp. 142–158, 2017.

[5] D. J. Deming and K. L. Noray, "Stem careers and technological change," 2018.

[6] H. Qin, "Teaching computational thinking through bioinformatics to biology students," in *Proceedings of the 40th ACM technical symposium on Computer science education*, 2009, pp. 188–191.

[7] T. Hsu, S. Chang, , and Y. amp; Hung, *July 03)*. How to learn and how to teach computational thinking: Suggestions based on a review of the literature, 2021.

[8] D. Gong, H. Yang, , and J. amp; Cai, "(2020," *June 08). Exploring the key influencing factors on college students' computational thinking skills through flipped-classroom instruction. Retrieved April*, vol. 21, 2021.

[9] J. Lyon, , and A. amp; Magana, "(2020," *July 02). Computational thinking in higher education: A review of the literature. Retrieved April*, vol. 21, 2021.

[10] F. J. Agbo, S. S. Oyelere, J. Suhonen, and S. Adewumi, "2019," *A Systematic Review of Computational Thinking Approach for Programming Education in Higher Education Institutions. In Proceedings of the 19th Koli Calling International Conference on Computing Education Research (Koli Calling '19). Association for Computing Machinery, New York, NY, USA, Article*, vol. 12.