Leveraging Generative Artificial Intelligence to Broaden Participation in Computer Science

Devang Jayachandran[1], Pranit Maldikar[1], Tyler S. Love[2], Jeremy J. Blum[1]
[1]Pennsylvania State University, Harrisburg
[2]University of Maryland, Eastern Shore
dxj5305@psu.edu, ppm5404@psu.edu, tslove@umes.edu, jjb24@psu.edu

Abstract

Generative Artificial Intelligence (AI) was incorporated into a competitive programming event that targeted undergraduate students, including those with little programming experience. The competition incorporated a range of challenge design approaches that promoted meaningful interaction with generative AI system, even while keeping the challenge difficulty level to an appropriate level. An analysis of survey responses and competition data showed that this format lowered barriers to participation, successfully engaged students throughout the competition, and increased the likelihood that they would participate in a similar event. In an extension of this work, a professional development workshop for high school teachers is being developed, along with a contest for high school students. Participant surveys and logs of interaction with the contest and generative AI systems will be analyzed to measure the effect of generative AI on student self-efficacy and suggest ways to integrate generative AI instruction into computer science curriculum.

Introduction

Despite the pedagogical benefits of engaging students in competitive programming opportunities, U.S. student participation remains low. This manuscript presents a work-in-progress that seeks to evaluate the ability of generative Artificial Intelligence (AI) tools to address barriers with engagement in competitive programming and broaden participation in computing (especially for underrepresented populations and students with limited coding experience). The project focus is competitive programming, where high school students will be provided with a generative AI system to aid them in solving the contest problems. The initial stage of the project examined lessons learned from offering a similar contest for undergraduate students that was used to inform the creation of the high school contest.

Preliminary findings from a ChatGPT-assisted contest, organized for undergraduate students, show that this approach holds promise for expanding participation in competitive programming events. The participation in the contest far exceeded organizers' expectations, and an analysis of the contest results and post-contest survey reveal that although many students had limited programming background, almost all teams were engaged in problem solving throughout the entire contest, and most teams solved multiple problems. A majority of students reported a desire to participate in a similar contest in the future.

A key contribution of this contest was the development of design features for accessible competitive programming problems that promote interaction with generative AI system. These features include challenges with missing requirements that can be provided through generative AI, deviations from very common patterns create challenges for generative AI systems, and challenges that require students to translate or transform requirements, for example, when presented as images.

In the next phase of the project, we are holding a contest for high school students which will be similar in nature to the preliminary undergraduate contest problems. Prior to the contest, a professional development workshop for high school teachers will be held. The focus of the workshop will be on prompt engineering, exploring how generative AI systems can be used by beginning computer science students to solve algorithmic puzzles. The workshop will teach strategies including effectively crafting initial prompts, techniques to evaluate AI generated output, and follow-on prompting to improve generative AI output, all within the context of contest programming. In addition, to evaluate the effect of participation in the contest, we are incorporating a survey that was validated in a prior K-12 coding study. The survey will evaluate high school students' interest in coding, their coding self-efficacy, and their views about coding related to future post-secondary education and career opportunities. The goal will be to analyze the survey data while controlling for specific covariates, such as student demographics and previous computer science training, and

whether their instructor completed the professional development workshop delivered prior to the contest.

Related Work

Competitive programming participation provides a range of improved student learning outcomes and employment opportunities, including improving student understanding of core concepts, building teamwork skills, and preparing students for technical interviews (Bloomfield and Sotomayor 2016). Despite these benefits, participation in competitive programming, particularly among U.S. students, has been declining (Blum 2023). This study of ICPC contest participation identified problem set difficulty as a key barrier for participation.

In general, various scaffolding approaches have been suggested to address difficulty in getting started with programming tasks. Consider, for example, the most common approaches for automated programming grading systems. In their review of automated feedback generation, Keuning *et al.* noted that in most platforms, the feedback is limited to knowledge about mistakes, including compiler errors, performance issues, solution errors, and test failures (Keuning, Jeuring, and Heeren 2018).

This limited feedback presents a significant challenge for beginner programmers who may need more help to produce a submission that can compile. Marwan *et al.* posited that this limited feedback hinders student success, and they developed an approach with frequent feedback built into the integrated development environment to help beginning students develop their correct submissions (Marwan *et al.* 2020). They noted that their approach of providing frequent, positive feedback was more appropriate for novice programmers who would otherwise not receive feedback until they can build a mostly complete code submission.

Increasing support for students through robust feedback can help to motivate students and encourage them to persist in challenging tasks. For example, when there is rapid and extensive feedback through autograders, students experience higher levels of motivation, and they are more likely to persist until the assignments are perfect (Furubotten 2015).

Recent advances in generative AI have investigated the ability of large language models to provide this scaffolding and feedback for students. A study using ChatGPT, for example, found that the tool was for the most part a beneficial aid to students, answering their questions, providing template code, facilitating debugging, and increasing student self-confidence (Yilmaz and Yilmaz 2023). A study that used Github's Copilot found similar benefits for novice programmers (Prather *et al.* 2023).

In the competitive programming space, these tools have shown a remarkable ability to solve contest problems. AlphaCode, for example, achieved an average ranking in the top 54.3% in simulated evaluations of recent programming competitions on the Codeforces platform (Li *et al.* 2022). However, studies have shown that it is still possible to create problems that are accessible to novice programmers but not readily solvable by these systems. For example, a study using GPT-3.5 found that these systems still struggle with contest programming problems that have image-based, visual explanations and problems that combine more than one solution idea, as well as standard problems with small variations on common problems (Jayachandran *et al.* 2023).

Preliminary ChatGPT-Assisted Contest

The Women in Technology club at Penn State University approached the authors to organize a programming contest that allow students, even with limited programming background, to be successful. The contest was structured so that students worked in teams of three, assisted by generative AI, to solve a set of problems that were designed to require non-trivial interaction with the generative AI. The contest drew a larger than expected group of contestants from a range of majors. Data from the competition and a post-contest survey supported the assertion that generative AI successfully made the competitive programming event more equitable regardless of programming experience.

Problem Set

The contest consisted of nine algorithmic challenges, hosted on the HackerRank platform[1]. The challenges in the contest were designed so that many of them would be solvable by students with limited programming experience. Our goal was to encourage students to leverage ChatGPT as part of their problem-solving process. It is important to note that the problem set was developed with the GPT 3.5 large language model from April 2023, and this model has been updated since then.

At the time of the contest, if students simply copied and pasted the code from the challenge, ChatGPT would be able to provide template code for many of the challenges. However, some the other problems were chosen to encourage students to interact with the generative AI system in a range of different ways. These included specifying the initial prompt, generating missing details from the challenge specification, and debugging the generated code output.

The *We Are!* Challenge asked students to implement the response to the Penn State chant. The chant typically includes 3 calls "We are…", each followed by a response "Penn State." The problem specification included these calls

[1] The challenges from the contest are available at: https://www.hackerrank.com/psh-contest

and responses, but it did not include the final call "Thank you", which is followed by a "You are welcome" response. At the time of the contest, ChatGPT, with the GPT3.5 backend, provided template code for the first calls and responses, but was unable to provide the final call and response. Students, however, needed to recognize deficiencies and identify patterns from the partial template solution to develop a complete solution.

Three challenges specifically addressed limitations of the ChatGPT in producing correct answers when there are deviations from standard patterns. For example, the challenge, *A Better Rock-Paper-Scissors Game*, asked students to evaluate a version of Rock-Paper-Scissors game, with a deviation of the traditional rules. In this version, Rock beat Paper but lost to Scissors. Interestingly, ChatGPT was unable to produce code for the new version of the game, perhaps because its training set includes many versions of the original game. Nonetheless, it was able to produce template code that required a fairly simple modification to pass all test cases. In the *Color Game* challenge, the goal was to order colors based on a preference. However, the color "fuchsia" was misspelled in the challenge specification, and ChatGPT tended to produce answers which were incorrect with respect to the specification, since ChatGPT fixed the spelling error. The *Mirror View* challenge presented part of the specification in the form of images, which prevented the problem from simply being copied and pasted into the ChatGPT system. The problem involved a square matrix consisting of 0s and 1s, along with a specified direction. Participants were expected to return the matrix after a mirroring operation was applied. The details of the mirroring operation were presented in images. To use ChatGPT, the participants would need to describe each of the mirroring operations presented in the challenge's images.

The *Moving Day* challenge asked students to specify the room in which various furniture or household items should be placed. The complete list of rooms was specified, as "bedroom," "dining room," "kitchen," "living room," or "office." However, the complete list of items was not given. We anticipated that participants could use generative AI to create template code for the program. However, they would need to expand on the item lists. The complete list of items could be generated by prompting the ChatGPT to list the most common items found in each of the rooms.

ChatGPT produced incorrect or inefficient code that could be debugged with further interaction with the system. For example, the challenge, *Water Is Your New Best Friend*, is a more advanced problem involving breadth-first search. The variation is that the search should only terminate after finding all paths within a grid to multiple destinations. The tests with ChatGPT found that it typically produced an incorrect solution because it terminated after finding the first path.

The *Big Eats* challenge asked students to select integers from two lists that are closest to a target value. Perhaps, due to the common computer science problem patterns, ChatGPT tended to produce solutions that considered only integers whose sum was less than the target value. However, if participants created a follow up prompt that described this shortcoming, ChatGPT could fix its code.

Similarly, a conversation with ChatGPT could be used to debug the output for the challenge *The Tournament Results*. In this problem, the participants are required to determine the remaining teams in a single elimination tournament. They are given a list of games that have occurred so far, in a random order. The most common solution produced by ChatGPT was incorrect because it assumed that the game results were chronological, not random. However, if this incorrect assumption were described in a follow up prompt, it would typically produce a correct solution.

Contest Analysis

The contest had a high participation rate. An analysis of the contest data and post-contest survey indicated that generative AI was a key reason for participants choosing to participate, and that it enabled students to be successful in the contest, even when they had limited programming background.

The contest was held over 1 hour and 45 minutes on a Friday in April. Announcements included prizes, with $250 for the first-place team, $150 for the second-place team, and $100 for the third-place team.

The organizers anticipated that it would draw at most 50 participants. The contest greatly exceeded this estimate, with more than 132 participants forming 52 teams during the contest. Most participants were in their first year of their respective majors.

The post-contest survey was sent to the participants, and it was completed by 45 out of the 132 participants (34% response rate). The survey results found that the majority of students (73%) were computer science majors. Other majors that were represented at the contest included Electrical Engineering, Cybersecurity, Public Policy, and Information Systems. Most of the survey respondents (61%) had no previous competitive programming contest experience.

Despite the limited experience in coding and contest programming, participants were engaged across the 120 minutes in which the contest took place. Figure 1 shows the average scores for all teams on each of the challenges. There is a spike of submissions in the first 10 minutes for the "We Are!" problem which had an average score of 68 implying that most participants were able to solve it completely or almost completely. Teams struggled finding the unspecified response. The "Moving Day" problem, which was designed to require interaction with ChatGPT to generate a list of

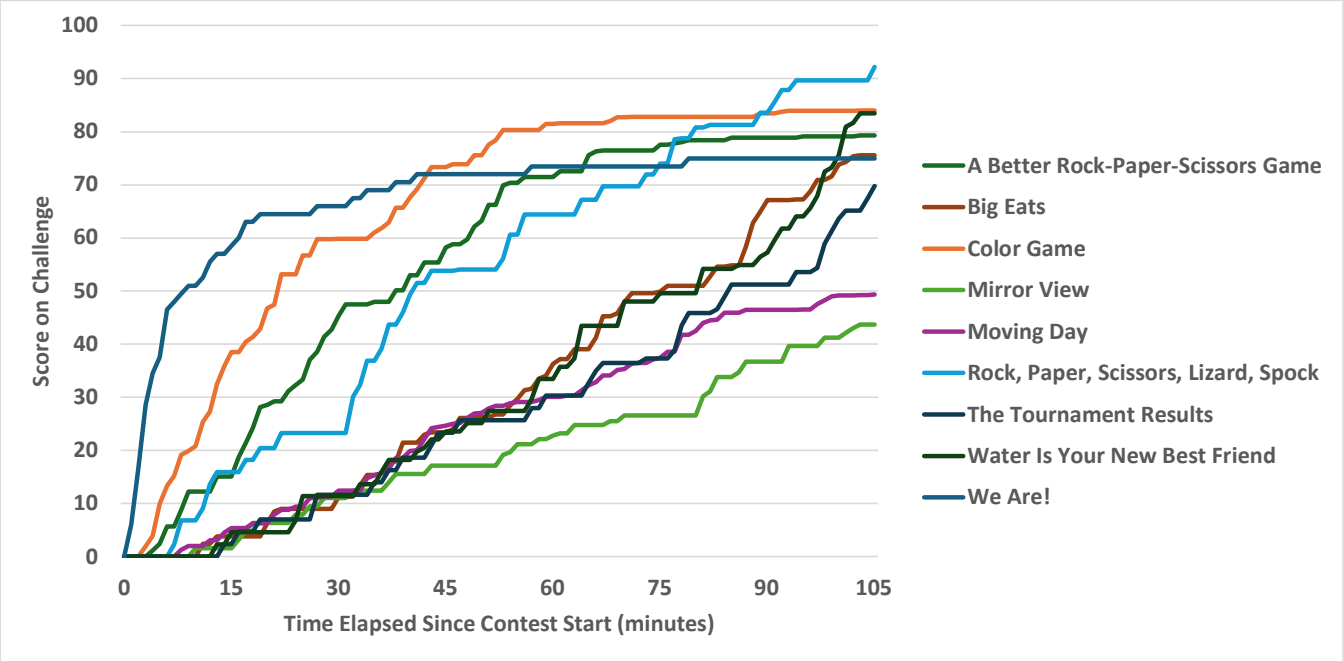

Figure 1: Average Score on Challenges for all Teams throughout Contest Duration. The maximum possible score for a challenge is 100 points.

household items, experienced a steady stream of submissions from the 10-minute mark up until the end of the competition.

The figure depicts the level of engagement for teams throughout the competition. The median time for the first submission was six minutes into the competition, and the median time of the team's last submission was 103 minutes into the competition, just two minutes prior to the end of the competition. The median number of submissions was 15. In total, teams made 1147 submissions, of which 154 were accepted. All 52 teams submitted their code, 49 teams had one completely correct submission. The lowest scoring team earned only 41 points. However, the next two lowest scoring teams earned 147 and 266 points, indicating that they were able to find solutions that could pass a portion of the test cases on multiple problems.

Respondents to the post-competition survey indicated that the competition was challenging, access to ChatGPT was key in their decision to participate in this contest, and similar access would make them likely to participate in another competitive programming event. Over 70% of the participants found the competition challenging. In addition, more than 70% of those surveyed indicated that access to ChatGPT played an extremely influential role in their decision to enter the competition. When assessing the utility of ChatGPT, 90% of survey respondents found ChatGPT to be a helpful resource throughout the competition. Over 66% of respondents agreed that access to ChatGPT increased their likelihood of participating in future programming competitions.

Students were asked open-ended questions about ChatGPT. In their answers, respondents cited several key benefits. Multiple students cited its ability to provide code that represented a strong starting point in problem-solving. In addition to the code that ChatGPT provided, respondents also appreciated the chatbot's ability to offer a comprehensive analysis and overview of problems, as well as its ability to assist in debugging. Other students used ChatGPT to help with code syntax. When asked about the most surprising aspects they had learned, respondents noted the chatbot's intelligence and accuracy in generating solutions, as well as its capacity to optimize solutions when requested. They were also impressed by the speed at which ChatGPT, and similar AI tools could nearly solve problems in a fraction of the time it would take a human. However, they did acknowledge some drawbacks, such as the chatbot timing out during the generation of large solutions, resulting in incomplete outputs. They also observed that slight changes in the prompts could lead to significantly different answers. Some respondents mentioned that ChatGPT could not solve 100% of the problems and occasionally failed to fix identified bugs. Ultimately, multiple students emphasized that ChatGPT's effectiveness is contingent on the user's understanding of the problem and the underlying concepts.

Planned High School Contest

Based on the success of the aforementioned competition with undergraduate students, we plan to hold a 2-hour programming contest for high school students in a Maryland school system. The purpose of this study is to address two key research questions. First, we plan to use a pre- and post-contest survey to measure whether access to generative AI to solve computer science problems changes students' views about coding. In addition, we will provide teachers with a professional development (PD) workshop to demonstrate how they can incorporate generative AI as an instructional tool to scaffold student learning and enhance students' coding problem solving skills. By controlling for whether students have been exposed to these strategies, we will evaluate whether this PD in generative AI improves the ability of students to successfully interact with these tools when solving computer science-related challenges.

Professional Development Workshop

In the summer prior to the contest, a half-day workshop will be provided for high school teachers. The workshop will start by providing a background in competitive programming problems, and then move to discussing effective strategies for incorporating generative AI tools to solve these problems. Specifically, three key areas will be covered: (1) using generative AI to analyze the problem requirements, (2) prompt engineering strategies to improve generative AI output, and (3) debugging strategies for iterative refinement of generative AI output.

Using the problems from the 2023 contest as an example, the workshop will show various techniques to use generative AI to analyze problem requirements. For example, the workshop will highlight the use of generative AI to generate missing requirements using the *Moving Day* challenge as an exemplar. In addition to prompting the generative AI system for a complete solution, the workshop will also illustrate how generative AI can summarize the problem requirements and recommend a solution approach.

The workshop will also cover prompt engineering strategies that can improve generative AI output. Examples include improving correctness by providing tests with the problem description (Murr, Grainger, and Gao 2023), summarizing the problem to provide clearer and more concise instructions (Busch *et al.* 2023), and providing goals and context for the prompts (Marvin *et al.* 2024).

In addition, the workshop will cover iterative refinement of prompts. The contest platform will often provide feedback on whether the submitted code is failing general cases or corner cases. In these situations, follow-up prompting with additional testcases can help the system correct its error. Alternatively, the platform may indicate that the solution is correct for small input tests, but takes too long to solve longer ones, and follow-up prompting for more efficient approaches can lead to an acceptable solution. In addition, the role of randomness in output will be described in the context of these systems, for example, the role of temperature in ChatGPT (Ouyang *et al.* 2023). Because of this randomness, if the first output of the generative AI system is incorrect, repeating the prompt a second time can provide correct output.

Data Collection Endpoints

Data from the contest systems and surveys will be used to address the study's research questions. A front-end for the GPT 3.5 engine has been developed for the contest that logs student interaction with the generative AI system while providing access that shields their identities from the OpenAI. In addition, the students will voluntarily complete a pre- and post-contest survey to report their interest in coding, self-efficacy toward coding, and perceptions of coding related to careers. We will use a modified version of Mason and Rich's Elementary Student Coding Attitudes Survey (ESCAS) (Mason and Rich 2020). This instrument consists of 23, six-point Likert scale items measuring the following five constructs: (1) coding self-efficacy, (2) interest in coding, (3) usefulness of coding related to future career plans, (4) social values toward coding, and (5) perceptions of coders.

Mason and Rich developed the instrument based on the Bandura's Self-Efficacy Theory, Expectancy Value Theory, and literature on gender and cultural stereotypes and perceptions (Mason and Rich 2020). Confirmatory factor analyses for each construct and a structural equation model indicated the instrument had strong validity measures. Cronbach's Alpha was calculated and showed acceptable to strong internal reliability for each construct.

We selected this instrument due to the constructs it investigates in alignment with our research questions, the brevity of the survey, which is important when adolescent participants are involved, and the strong validity and reliability measures. A series of demographic questions will be added to the beginning of the pre-contest survey. The three ESCAS items about social influence and five ESCAS items about perceptions of coders will be omitted from the pre- and post-contest surveys. We elected to omit these items because they are written for elementary students and do not relate to the research questions in our study. The 15 items from the coding self-efficacy, coding interest, and usefulness of coding constructs are appropriate for high school students and directly align with our research questions. Internal reliability tests will be reconducted on each construct in the pre- and post-contest surveys.

Paired samples t-tests for each construct will be used to analyze changes in participants' responses from pre- to post-contest. Additionally, factorial ANCOVA tests will be

conducted for each construct. In these analyses the dependent variable will be the post-contest survey scores, the pre-contest survey responses will serve as the covariate, and student demographics, previous computer science training, and whether their instructor completed the PD workshop delivered prior to the contest will serve as the independent variables. These analyses will allow us to examine to what extent using the generative AI system influenced students' interest in coding, their self-efficacy toward coding, and their perceptions about the usefulness of coding for future careers while controlling for specific demographic and training experiences. They will also allow us to examine if student results vary according to the preparation experiences of their teacher (e.g., if their teacher participated in the pre-contest PD workshop). In addition to examining whether teacher participation in the PD workshop produced better team performance in the contest, the logs of the generative AI system will be analyzed to determine the extent to which student interaction with the system is affected by their teacher's participation in the PD workshop.

Conclusions and Future Work

A generative AI assisted contest for undergraduate students seems to indicate that access to these systems can broaden participation in competitive programming events. The competition drew students from a variety of backgrounds. The students were engaged throughout the contest, and the majority reported that they would participate in similar contests in the future.

For the contest, a problem set was assembled that illustrates challenge features that promote non-trivial interaction with the generative AI, even though the challenges were accessible to beginning programmers. These features include providing challenges that require the use of generative AI to summarize or provide missing requirements, variations of common problems to which generative AI has difficulty adapting, and providing input that must be summarized by contestants.

We also describe the future expansion of this research into secondary education settings. Prior to the contest, a PD workshop will be provided for teachers to introduce them to a variety of strategies that can leverage the power of generative AI in competitive programming problem solving.

We anticipate that after students are provided with access to generative AI during a competitive programming contest, they will report greater interest in coding, improved coding self-efficacy, and an improved perception of the usefulness of coding skills for future careers. Moreover, we hypothesize that instruction on using generative AI prior to the contest will improve student's ability to more fully use the power of these tools. Should the data support this hypothesis, we plan to use the experiences described in this paper to develop curricular materials that can assist teachers with incorporating generative AI into secondary level computer science curricula.

Acknowledgments

This research was funded by the Institute for Computational and Data Sciences at Penn State University through the Inter-Institutional Partnerships for Diversifying Research (IPDR) Program, the Penn State University Center for Applications of Artificial Intelligence and Machine Learning to Industry Core Facility (RRID:SCR_022867), and a seed grant from Penn State Harrisburg's Office of Research and Outreach.

References

Bloomfield, A. and Sotomayor, B. 2016. A Programming Contest Strategy Guide. In Proceedings of the 47th ACM Technical Symposium on Computing Science Education (SIGCSE '16). Association for Computing Machinery. New York: Association for Computing Machinery. doi.org/10.1145/2839509.2844632.

Blum, J. J. 2023. Competitive programming participation rates: an examination of trends in US ICPC regional contests. *Discover Education,* 2(11). doi.org/10.1007/s44217-023-00034-1.

Busch, K.; Rochlitzer, A.; Sola, D.; and Leopold, H. 2023. Just tell me: Prompt engineering in business process management. In Proceedings of the International Conference on Business Process Modeling, Development and Support. Cham: Springer Nature Switzerland, 3-11. doi.org/10.1007/978-3-031-34241-7_1.

Furubotten, H. 2015. The Autograder Project: Improving software engineering skills through automated feedback on programming exercises. MS Thesis, Department of Electrical and Computer Engineering (TN-IDE), University of Stavanger, Stavanger, Norway.

Jayachandran, D.; Blum. J. J.; and Maldikar, P. S. 2023. An Analysis of the Impact of Advances in Generative Artificial Intelligence on Programming Assignments. In Proceedings of ASEE Zone 1 Conference, 10 pages.

Keuning, H.; Jeuring, J.; and Heeren, B. 2018. A Systematic Literature Review of Automated Feedback Generation for Programming Exercises. *ACM Transactions on Computing Education*, 19(1), 1-43. doi.org/10.1145/3231711.

Li, Y.; Choi, D.; Chung, J.; Kushman, N.; Schrittwieser, J.; Leblond, R.; Eccles, T.; Keeling, J.; Gimeno, F.; Lago, A. D.; Hubert, T.; Choy, P.; d'Autume, C. d. M.; Babuschkin, I.; Chen, X.; Po-Sen, H.; Welbl, J.; Gowal, S.; Cherepanov, A.; Molloy, J.; Mankowitz, D. J.; Robson, E. S.; Kohli, P.; Freitas, N. d.; Kavukcuoglu, K.; and Vinyals, O. 2022.

Competition-Level Code Generation with AlphaCode. *Science*, 378(6624): 1092-1097. doi.org/10.1126/science.abq1158.

Marvin, G.; Hellen, N.; Jjingo, D.; and Nakatumba-Nabende, J. 2024. Prompt Engineering in Large Language Models. In: *Data Intelligence and Cognitive Informatics. ICDICI 2023. Algorithms for Intelligent Systems*, edited by Jacob, I.J., Piramuthu, S., Falkowski-Gilski, P. Singapore: Springer. doi.org/10.1007/978-981-99-7962-2_30.

Marwan, S.; Gao, G.; Fisk, S.; Price; T. W.; and Barnes, T. 2020. Adaptive Immediate Feedback Can Improve Novice Programming Engagement and Intention to Persist in Computer Science. In Proceedings of ACM Conference on International Computing Education Research (ICER '20). New York: Association for Computing Machinery, 194–203. doi.org/10.1145/3372782.3406264.

Mason, S. L. and Rich, P. J. 2020. Development and Analysis of the Elementary Student Coding Attitudes Survey. *Computers & Education*. 153, 103898. doi.org/10.1016/j.compedu.2020.103898.

Murr, L.; Grainger, M.; and Gao, D. 2023. Testing LLMs on Code Generation with Varying Levels of Prompt Specificity. arXiv:2311.07599.

Ouyang, S.; Zhang, J. M.; Harman, M.; and Wang, M. 2023. LLM is Like a Box of Chocolates: the Non-determinism of ChatGPT in Code Generation. arXiv:2308.02828.

Prather, J.; Reeves, B. N.; Denny, P.; Becker, B. A.; Leinonen, J.; Luxton-Reilly, A.; Powell, G.; Finnie-Ansley, J.; and Santos, E. A. 2023. "It's Weird That It Knows What I Want": Usability and Interactions with Copilot for Novice Programmers. *ACM Transactions on Computer-Human Interaction*, 31(1): 1-31. doi.org/10.1145/3617367.

Yilmaz, R.; and Yilmaz, F. G. K. 2023. Augmented intelligence in programming learning: Examining student views on the use of ChatGPT for programming learning. *Computers in Human Behavior: Artificial Humans*, 1(2): 1-7, doi.org/10.1016/j.chbah.2023.100005.