Mirror or Window? A Student Study on LLM-Generated Reflection Questions for Self vs. Peer Code

Helen Wu
helen.wu@vanderbilt.edu
Vanderbilt University
Nashville, TN, USA

Kritan Bhandari
kbhandari07@my.fisk.edu
Fisk University
Nashville, TN, USA

Gina R. Bai
rui.bai@vanderbilt.edu
Vanderbilt University
Nashville, TN, USA

Yu Huang
yu.huang@vanderbilt.edu
Vanderbilt University
Nashville, TN, USA

Kevin Leach
kevin.leach@vanderbilt.edu
Vanderbilt University
Nashville, TN, USA

Abstract

CS1 programming assignments (PAs) provide students with vital coding experience, but creating programs with the correct output does not guarantee full coding proficiency. Questions on Learner's Code (QLCs) are assessment questions about a student's own PA solution to evaluate their conceptual understanding. While traditional QLCs are about the student's authored code, this study investigates whether answering AI-generated questions about peer code can strengthen learning outcomes. We conducted a classroom study comparing CS1 students who received LLM-generated QLCs based on either their own code (Self; $n = 16$) or a classmate's code (Peer; $n = 19$), alongside an anonymized control group of students who did not answer QLCs (Baseline; $n = 56$). Results indicated similar scores and time spent on exercises between the Peer group and the Self group. The Peer group scored significantly higher ($p = 0.033$) on the final exam than the Baseline group, indicating latent learning benefits for cumulative exam outcomes. Peer group participants also rated the helpfulness of QLCs higher than those in the Self group, citing the value of exploring alternative implementations of familiar tasks. These findings indicate that answering LLM-generated questions based on peer code effectively encourages reflection and reinforces computing concepts.

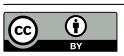
CCS Concepts

• **Social and professional topics → Computing education**; • **Applied computing → Interactive learning environments**.

Keywords

Questions on Learner's Code, peer code, CS1, LLMs, CS Education

ACM Reference Format:
Helen Wu, Kritan Bhandari, Gina R. Bai, Yu Huang, and Kevin Leach. 2027. Mirror or Window? A Student Study on LLM-Generated Reflection Questions for Self vs. Peer Code. In *Proceedings of the 58th ACM Technical Symposium on Computer Science Education V.1 (SIGCSE TS 2027), February 17–20, 2027, Sacramento, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3857222.3860134



This work is licensed under a Creative Commons Attribution 4.0 International License.
SIGCSE TS 2027, Sacramento, CA, USA
© 2027 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-3075-7/2027/02
https://doi.org/10.1145/3857222.3860134

1 Introduction

Most CS1 courses rely on programming assignments (PAs) to build students' coding experience through hands-on practice. However, novice programmers frequently struggle to develop code comprehension and programming skills alongside these PAs [10, 20]. Even if a student submits a working solution, there is not a guarantee they fully understand their code or the concepts behind it [13]. Lehtinen et al. [16] previously introduced Questions on Learner's Code (QLCs), questions asked about a student's own PA solution to evaluate code comprehension. The empirical study discovered one third of CS1 students failed to answer at least one QLC correctly, indicating an evident gap in conceptual understanding. Additionally, the authors found that answering QLCs correctly was higher correlated with CS1 student success and retention compared to submitting a working PA solution. This finding opens the potential for QLCs to be adapted as a novel reviewing strategy, not just an evaluation technique.

However, individual adoption is limited by scalability since manually creating QLCs for every student and PA would be highly time-intensive. Early automated QLCs frameworks used predefined templates and static parsers [15, 17, 22], but these rigid generation methods restrict the scope of questions that can be asked. Large Language Models (LLMs) offer a flexible alternative because they do not have restrictions on code style or programming language, allowing them to evaluate open-ended code solutions without needing strict, pre-defined parsing rules. Given their expanding adoption in CS education [21], recent work [5] has validated that LLMs can successfully generate high-quality QLCs for CS1 students.

Leveraging this automated adaptability of LLMs, we propose a novel pedagogical approach: investigating whether students benefit from answering QLCs derived from peer code submissions rather than their own. The inspiration comes from "unseen code analysis," where evaluating unfamiliar code snippets, either from peers [8] or external sources [7, 19], enhances code review capabilities, exposes students to alternative solution strategies, and refines their overall programming skills [8, 19]. However, to our knowledge, no prior work has attempted a peer approach with QLCs. Much like traditional peer code review, students who have already completed a PA approach a classmate's code with an established context. This pre-existing knowledge minimizes extra cognitive load [25] needed to understand the problem, allowing students to focus on

deeper conceptual reasoning rather than navigating an entirely new problem.

We aim to answer the following questions:

- **RQ1**: How does answering LLM-generated QLCs on one's own code versus a peer's code impact student learning outcomes?
- **RQ2**: How do students perceive the relevance, comprehensibility, and helpfulness of LLM-generated QLCs when applied to their own code versus a peer's code?

We release the source code, system prompts, recruitment documents and anonymized data analysis to be publically available[1].

2 Background & Related Work

Questions on Learner's Code. Previous literature has indicated poor conceptual understanding [12], program comprehension [10], and code writing abilities [18, 20] among new programmers. Since functioning code can be written with trial-and-error or external assistance, a working PA solution does not guarantee coding proficiency [13]. QLCs were proposed by Lehtinen et al. [16] to support and evaluate a thorough understanding of a student's own code. These questions prompt students to critically analyze their implementation, which has been shown to benefit students' learning outcomes [26].

Two approaches have been used to implement QLCs in classrooms. Researchers either 1) manually wrote questions [16] or 2) extracted information from code and used pre-defined templates to generate questions [15, 17, 22]. While reliable, manually creating questions is not scalable for large numbers of students. Using templates allows for automation, but it limits the types of QLCs to primarily be syntax-focused, and an already-developed template is not easily generalizable to other PAs or programming languages.

LLM Question Generation. Modern LLMs can help provide a solution to these barriers of scalability, question diversity, and generalizability. Prior work has demonstrated that LLMs can effectively generate multiple-choice questions (MCQs) for CS classrooms automatically [6, 24, 27, 28], making LLMs a viable alternative for QLC generation. Since LLMs do not rely on specific programming languages, unlike previous QLC automation attempts [17, 22], LLM-based generation provides a generalizable approach for creating QLCs for CS1 students. Supporting this potential, recent work by Goodfellow et al. [5] explores a highly similar approach to dynamically generate QLCs using a Moodle Virtual Learning Environment[2] plugin powered by GPT-4o.

Unfamiliar Code Analysis. CS1 students perceived unseen code analysis as helpful for developing their program comprehension and coding proficiency [4, 7]. Previous investigations include explicit code tracing [7], plain-English explanations [19, 23], and think-aloud exercises [9, 10]. As discussed by Kapoor et al. [11], students believe that the familiar problem context provides essential scaffolding to promote code understanding because they already know the underlying logic and goals.

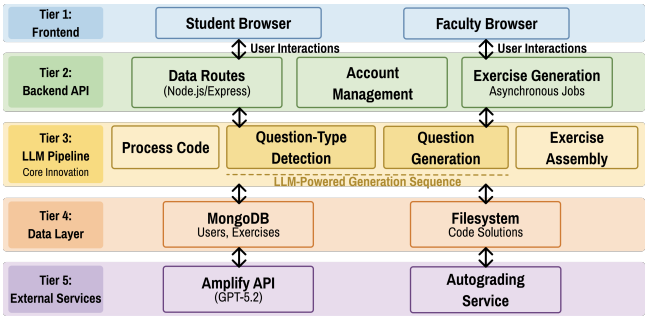


Figure 1: The architecture for the platform, including how QLCs are generated.

3 Study Design

In this section, we present our approach of automatic QLC generation to improve student learning outcomes and investigate the efficacy of answering peer code QLCs. We designed a custom platform leveraging ChatGPT to automatically generate QLCs based off self or peer code, assign exercises to students, and collect answers and observations about QLCs. We deployed our approach in a CS1 classroom at an accredited university in the US during the Spring 2026 semester. In the following subsections, we describe the website application and the details of the classroom study where this system was implemented.

3.1 QLC Generation Platform

Figure 1 outlines the architecture for the website application. The platform is deployed on an on-premises server for institutional compliance, it consists of a react-ts frontend with an Express and NodeJS backend through a RESTful API. Most data (e.g., users, exercises, PAs) is stored in a local MongoDB[3] database on the server.

When a user logs into the website, the SSO will redirect them to the Dashboard page. First-time users are given the student role and randomly assigned to the Self or Peer group as described in Section 3.3.

3.1.1 *Website Pages.*

- **Dashboard**: Students access their current exercises, and faculty get graphs about students' performance.
- **Chapters**: Students can review previous exercise results, and faculty can edit PA details.
- **Progress**: Faculty can view individual students' progress and download results in a CSV file.
- **Exercises**: Faculty can generate and look over exercises before they are released to students.

3.1.2 Exercise Generation. We generate QLCs with the GPT-5.2 model from Amplify API [4] in two steps. We first provide the LLM with a PA solution and ask it to provide 1) a list of relevant question types and 2) directives to generate the exercise. These artifacts are fed to the same LLM to output an exercise with five QLCs, which is returned as a structured JSON object. Each QLC includes

[1] https://github.com/largehappygroup/qlcs-files
[2] https://moodle.org/
[3] https://www.mongodb.com/try/download/community
[4] https://www.vanderbilt.edu/generative-ai/custom-software-pilot-amplify/

a correct answer with a corresponding explanation and three incorrect answers with tailored hints [2] for each option. We chose this two-stage pipeline based on our preliminary exploration of various prompt strategies. Providing structured question types and directives yielded more technically correct results than single-step prompting. This approach removed ambiguous code quality critiques prone to multiple valid answers. The system prompts for question generation can be modified according to an instructor's needs to incorporate specific topics and learning objectives.

To evaluate our final strategy, we generated 54 questions from 13 CS1 PA solutions of varying complexity from three online sources[5]. These questions were verified by two co-authors who have knowledge of CS fundamentals. The reviewers evaluated the questions against a rubric for linguistic clarity, course appropriateness, code tracing reliance, objective answerability, and overall technical correctness. Across all the categories, 80% of the generated questions were rated usable, with a prevalence-adjusted and bias-adjusted kappa (PABAK) score of 0.74, indicating substantial inter-rater reliability [14].

3.1.3 Answering QLCs. When a student opens an exercise, they receive a PA code submission, a query and four answers in random order. An incorrectly-chosen answer will result in a hint being displayed and a prompt to try again. Since questions may contain LLM hallucinations, we provided a hyperlink for students to contact the research team if they suspect any part of the question to be incorrect (e.g., incorrect answer or explanation, unreasonable query). Students cannot move on until they eventually provide the correct answer, which the backend verifies with the correct answer stored in the database.

Figure 2 shows the student view upon submitting the correct answer, where they are presented with a detailed explanation on why the option was right. Students were asked to rate the question's *clarity* (1: Very hard to 5: Very easy to understand) and *helpfulness* (1: Not helpful to 5: Very helpful for understanding coding concepts). Students could also flag student-written code they suspected was AI-generated. To ensure no student code had unfamiliar syntax, the research team discussed flagged submissions post-hoc to ensure there was a consensus for appropriateness for the CS1 level.

3.1.4 Feedback on Overall Experience. The feedback form includes the following statements:

- **S1**: The exercises are easy to understand.
- **S2**: The exercises are asking reasonable questions.
- **S3**: The exercises are helping me better understand the programming assignment code they are based on.
- **S4**: The exercises are overall helping me better understand the Java programming language.
- Please give any additional feedback you have about the exercises.

Except for the last free-response question, all are rated on a Likert scale: Strongly Disagree (1), Disagree (2), Neutral (3), Agree (4), and Strongly Agree (5).

[5]https://www.w3resource.com/java-exercises/index.php, https://github.com/HarryDulaney/intro-to-java-programming, https://github.com/pavel-rossinsky/java-a-beginners-guide-herbert-schildt

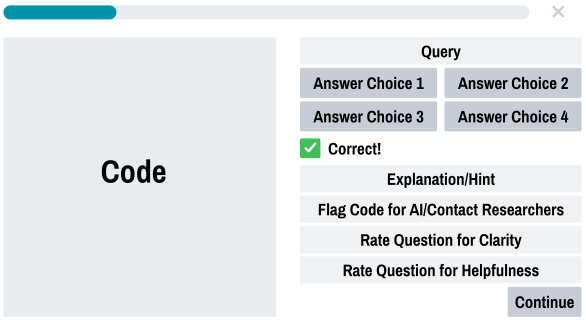


Figure 2: The layout of an exercise popup for a correctly-answered question.

3.2 Classroom Setting

We conducted an IRB-approved classroom study during the Spring 2026 semester with undergraduate students enrolled in a CS1 course at an accredited university in the United States. The class is required for all CS majors and minors, some engineering majors, and is open to non-majors[6]. The goal of this introductory course is for students to learn Java fundamentals and object-oriented design.

Weekly PAs are built into the course for coding practice, and each student independently completes them (excluding assistance from instructional staff). Automated test cases on an external autograding service evaluate the correctness of the code solutions with respect to a pre-defined test suite. There are 29 PAs and four exams (3 midterms, 1 final) during the 16-week semester. The study procedure started after the second midterm (Week 7) and ended the week of the third midterm (Week 11). The study was conducted over 10 PAs, with topics ranging from if-statements, loops, arrays, file input/output, and the Random object.

3.3 Participant Recruitment

A research team member held a 10-minute recruitment presentation during CS1 class periods. Students were sent a follow-up email with 1) a link to a consent form to be signed by eligible students, 2) a link to the website application, and 3) a link to the recruitment slides and a platform user manual.

Eligible students (at least 18 years old and enrolled in CS1) had half a week to opt into the study through the consent form and make an account with their institution credentials. Participants provided basic identifiable information (e.g., name, email address, student ID) and their level of programming experience (years and previous coursework) prior to CS1 in the consent form. All CS1 students, including non-participants, who completed at least 90% of exercises and completed the feedback form were eligible to receive extra credit applied to their final CS1 grade. We excluded all non-participating students' exercise and feedback data from analysis.

There are four groups defined within the scope of this paper:

- **Self**: Students received QLCs about their own PA code.

[6]The university does not require declaration of majors during the first year, which is when students are expected to take this course.

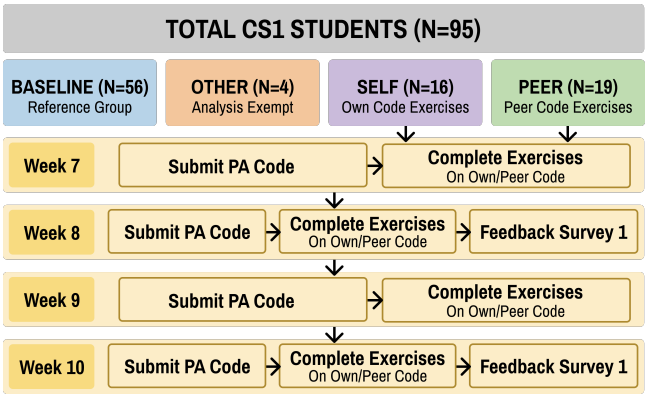


Figure 3: The classroom deployment procedure, from groups to assigning exercises

- **Peer**: Students received QLCs about an anonymized peers' PA solution.
- **Baseline**: Students did not complete any QLCs.
- **Other**: Students under special cases (e.g., completed at least one exercise and didn't opt into the study, didn't take an exam). These students were exempt from all data analysis.

Figure 3 outlines the experimental procedure. Upon account creation, the platform randomly divided participants into the Self and Peer groups, and both groups used the platform unmonitored. Following each PA deadline, a researcher uploaded student code and test case reports from the external autograding service into the application server. To ensure quality, the researcher briefly screened the generated outputs and fully regenerated any exercises that contained duplicate questions or answers.

The application backend routed exercises based on group assignment while enforcing several system constraints:

- **Filtering:** To guarantee code quality, the Self group automatically received a peer's code if their own code failed to compile or scored below 75% of test cases.
- **Anonymization:** All author-identifying information (e.g., comments and filenames) was strictly stripped from code, and no author was notified of who reviewed their work.
- **Privacy:** Consented study participants received exercises exclusively generated from other participants' code.

Students completed the feedback form in Section 3.1.4 every other week to evaluate their experience.

4 Results

Out of 95 students enrolled in CS1 for Spring 2026, 44 opted into the study. A total of 35 active participants (Self: $n = 16$; Peer: $n = 19$) completed 276 of 427 assigned exercises. The distribution of active Self and Peer group students are shown in Table 1. The Self and Peer groups submitted 19 and 22 feedback forms, respectively. The Baseline group ($n = 56$) contain the remaining CS1 students who did not participate, including eight opted-in students who did not complete any exercises. Finally, four students were put in the Other group and excluded from any data analysis: one opted in late, two non-participants completed exercises, and one non-participant missed at least one required exam.

Table 1: Demographics for Self and Peer groups.

Demographics		Self	Peer
Gender	Male	11	8
	Female	4	7
	Others	1	4
Prior Coding Experience	No Experience	3	5
	Beginner	10	9
	Intermediate	3	5
	Advanced	0	0
Prior Programming Courses	Taken	4	8
	Not Taken	12	11

The exceptional scenarios described in Section 3.3 required intervention. Specifically, 7.73% of assigned exercises (2.8% of completed exercises) required a Self group participant to receive a peer-based exercise because their own code failed to meet requirements. 3.98% of assigned exercises were regenerated due to duplicate questions or answers. Students also reported 13 suspected cases of LLM hallucinations. After manual review by the research team, 12 instances were verified as hallucinations. Nine of them occurred from the LLM miscounting characters in a string, two were from arithmetic errors, and one was due to the UI not being distinct with whitespace differences between answers. The corresponding questions were marked as correct in the database, and the affected students received a detailed explanation of the error.

In our experiment, there were participants who answered QLCs based on code written by others. However, we note that it is possible two students may coincidentally produce highly similar code for CS1-level assignments. Therefore, we used the Measure Of Software Similarity (MOSS) [1] to quantify the similarity between the Peer group participants' code and the student code from the QLCs they answered. We measured an average MOSS similarity of 0.26 (out of 1.0), indicating a very low lexical and syntactic similarity. This suggests that Peer group participants were examining structurally different code from their own submission.

4.1 RQ1: Impact on Learning Outcomes

Across all completed exercises, students in the Self group achieved a similar average score (0.65 ± 0.29) to those in the Peer group (0.63 ± 0.29). Total time spent per exercise was also similar between the two groups, with the Self group averaging $417s \pm 414s$ and the Peer group averaging $415s \pm 415s$. Pairwise t-tests indicated the differences in both score ($p = 0.678$) and total time spent ($p = 0.960$) per exercise were not statistically significant.

We analyzed student performance across exams to evaluate the impact of exercises on learning outcomes. The average exam score for each group is shown in Table 2. Midterms 1 and 2 were combined to create an exam score baseline since both were given before the study began. Pairwise t-tests were conducted between the three groups for each exam period. No statistically significant differences were discovered for the combined exams or Exam 3. On the final exam, most comparisons were similarly insignificant, except for

Table 2: Exam Scores for Each Student Group

Group	Exams		
	1 & 2	3	Final
Self	82.93 ± 10.52	79.53 ± 14.07	82.73 ± 12.76
Peer	83.72 ± 6.25	83.61 ± 9.34	83.24 ± 8.52
Baseline	80.64 ± 11.31	78.18 ± 16.24	76.35 ± 18.69

Table 3: Average and Distribution of Per Question Ratings (1-5).

Metrics	Self		Peer	
	Average	Distribution	Average	Distribution
Clarity	3.54	1 5	3.59	1 5
Helpfulness	3.41	1 5	3.63	1 5

Table 4: Average and Distribution of Feedback Form Likert Scale Responses.

Statements	Self		Peer	
	Mean	Distribution	Mean	Distribution
S1: Understanding	3.08	1 5	3.83	1 5
S2: Reasonable	3.58	1 5	4.25	1 5
S3: Helpful Code	3.58	1 5	3.67	1 5
S4: Helpful Java	3.67	1 5	3.83	1 5

a significant difference observed between the Peer and Baseline groups ($p = 0.033$; Cohen's $d = 0.411$).

4.2 RQ2: Student's Perceptions of QLCs

4.2.1 Per Question Ratings. To evaluate users' perception of the generated question quality, we aggregated the per-question ratings for clarity and helpfulness across all completed exercises. Table 3 shows the average and distributions of those aggregated Likert scale ratings. Pairwise t-tests were conducted to compare these metrics between groups. While the difference in perceived clarity was not statistically significant ($p = 0.401$), the helpfulness t-test observed a highly significant difference ($p < 0.001$; Cohen's $d = 0.200$). This indicates the Peer group rated the individual questions as more helpful than the Self group.

4.2.2 Feedback Form Responses. Feedback form responses as outlined in Section 3.1.4 were collected during the second and fourth week of the study. For students who successfully responded in both instances, a paired t-test revealed no significant differences, so the Likert scale ratings were averaged in such cases. Table 4 displays the average and distribution of Likert scale ratings for each statement across the Self and Peer groups.

Pairwise t-tests conducted on the four survey statements revealed statistically significant differences between the groups for statements S1 ($p = 0.015$; Cohen's $d = 1.081$) and S2 ($p = 0.021$; Cohen's $d = 1.033$), indicating the Peer group rated the questions as easier to understand and more reasonable than the Self group did. S3 ($p = 0.732$) and S4 ($p = 0.368$) did not reveal a statistically significant difference between the two groups.

We also gave students the option to add their comments as free text. We conducted a Braun and Clarke [3] thematic analysis on the statements for each group.

4.2.3 UI/UX and Formatting Barriers. This was the most prevalent theme throughout the feedback, with eight Self group and three Peer group students noting interface and formatting obstacles that prevented them from understanding a question fully. Some mentioned an issue of extremely long answers (typically questions where the student had to predict the output given an input), for example:

> "The formatting of the questions cuts off the end of a lot of the answers, and some all have the same beginning so I end up just randomly choosing an answer." (Self participant)

Some noted that the LLM would choose questions where the difference between answers would be subtle formatting details (e.g., newline character or trailing whitespace) with no font differentiation, which made selecting an answer difficult:

> "...there were two answer choices that looked the same, but...the newline was placed differently...it is hard to tell visually." (Self participant)

> "...lack of font difference makes it hard to tell where the 'Run 1' output ends and where 'Run 2' starts..." (Self participant)

Additionally, the website application was designed to shuffle the answer choices to ensure LLM generated output wasn't predictable based on position. However, this actually became a pain point and added unnecessary cognitive load for two Self group students:

> "...it would be better if the answer choices stay in the same spots after you get a question wrong because it takes too much time to read through the same answer choices again..." (Self participant)

4.2.4 LLM-generated Artifacts. Four Self group students and one Peer group student identified the queries from some questions as too long. This is likely an artifact of the LLM-generated output:

> "...the prompts are rather long and more about code application than actually understanding." (Peer participant)

4.2.5 Student Cognitive Engagement. One Self group student and two Peer group students mentioned reading the code itself wasn't as necessary to answer the question since they were already familiar with the task:

> "...it doesn't help all that much to use the PA assignments as examples, as we can work through the problems without even looking at the code." (Self participant)

Two Self group students and one Peer group student also mentioned a disconnect between the exercise questions and the ones on an exam, for example:

> "More complex questions would be nice. Maybe further applications like what might be seen on a test." (Peer participant)

4.2.6 Positive Reinforcement. Two students in the Peer group mentioned they found the exercises helpful for reinforcement and for seeing code implementations they hadn't done in their own programs:

> "I also havent implemented the for each things in my own code, so this was helpful in helping me understand that more." (Peer participant)

5 Discussion

Students who answered questions about a classmate's code performed just as well (Self: 0.65 vs. Peer: 0.63) and spent the same amount of time (Self: 417s vs. Peer: 415s) on the exercises as those who answered questions about their own code. This is an indication that unfamiliar code questions do not inherently hinder student performance on these code exercises.

Additionally, the Peer group on average achieved 6.89% higher on the final exam than the Baseline group, which was a statistically significant difference ($p = 0.033$). However, this significance was absent for exam 3 ($p = 0.366$). This discrepancy suggests that the peer code questions may have advantages that are latent or cumulative rather than immediate.

The Self group was much more critical of the readability and fairness of the questions compared to the Peer group, and they left more comments about UI/UX and formatting. One explanation outcome could be constraints of cognitive load [25], where understanding complex logic consumes most of a problem-solver's processing capacity. Because the Self group was already highly familiar with their own code, their intrinsic cognitive load was lower than that of the Peer group. This freed up the necessary bandwidth to notice and critique more external factors like formatting and interface design.

Students in both groups generally held a positive perception regarding the helpfulness of QLCs. This is more apparent by looking at the distribution of S3 and S4 in Table 4. This aligns with findings from previous studies [5, 22], where students reported QLCs as helpful in developing a better understanding of their code and programming concepts. However, despite these positive perceptions, only the Peer group experienced a significant increase in exam scores, and this occurred only for the final exam. Similar to [5], the immediate impact of LLM-generated QLCs on summative assessments for a student's own code remains non-significant. This suggests that while most students value the exercises, the measurable cognitive benefits may be more subjective or might not be evident in short-term exam performance.

Future research could be a longitudinal study to measure the long-term impact of QLCs on student performance, especially as students transition to more rigorous programming courses. Furthermore, the current implementation of the system is limited to providing students with an explanation of the correct answer. Although it generates hints on why the other answers are incorrect, supplying direct and targeted feedback on specific misconceptions may be more effective in improving immediate learning outcomes.

6 Threats to Validity

Students who opted into the study were offered extra credit, which may have led specific students (e.g., those with lower grades or more concern over GPA) to participate. Additionally, code submissions were filtered to include only those that successfully compiled and passed 75% of the autograding test cases. This baseline requirement ensured that no student was unfairly disadvantaged by receiving non-functional code that could either result in unpredictable question generation or hinder the students' ability to answer the QLCs. Given the large volume of generated content, exhaustive manual verification of every exercise before giving them to students was unfeasible. We mitigated the risk of LLM errors by performing a high-level screening process of the generated questions and a direct reporting system for students to flag any remaining issues.

7 Conclusion

We conducted a controlled experiment with 35 CS1 students to investigate whether LLM-generated Questions on Learner's Code (QLCs) derived from peer code offer distinct learning benefits compared to QLCs generated from a student's own code. While we observed no immediate differences in performance or time spent during the exercises, both groups generally perceived the activities as moderately understandable and helpful. Notably, the Peer group rated the exercises as significantly more helpful ($p < 0.001$) after completion. Additionally, the results of this study imply that answering questions about peer code may lead to long-term benefits on cumulative exam outcomes. These findings suggest that LLM-generated QLCs based on peer code serve as a viable and valuable tool for broadening conceptual understanding in introductory programming courses.

Acknowledgments

This work was funded in part by grants NSF 2544037, NSF 2211429, NSF 2622415, and NSF 2312057. The authors would also like to express their sincere gratitude to all the study participants whose time and effort made this research possible.

References

[1] Alex Aiken. 1994. MOSS: A System for Detecting Software Plagiarism. https://theory.stanford.edu/~aiken/moss/. Stanford University.

[2] John R. Anderson, Albert T. Corbett, Kenneth R. Koedinger, and Ray. Pelletier. 1995. Cognitive Tutors: Lessons Learned. *Journal of the Learning Sciences* 4, 2 (1995), 167–207. arXiv:https://doi.org/10.1207/s15327809jls0402$_2$ doi:10.1207/s15327809jls0402_2

[3] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative Research in Psychology* 3, 2 (2006), 77–101. arXiv:https://doi.org/10.1191/1478088706qp063oa doi:10.1191/1478088706qp063oa

[4] Tamaike Brown, Mourya Reddy Narasareddygari, Maninder Singh, and Gursimran Walia. 2019. Using Peer Code Review to Support Pedagogy in an Introductory Computer Programming Course. In *2019 IEEE Frontiers in Education Conference (FIE)*. 1–7. doi:10.1109/FIE43999.2019.9028509

[5] Martin Goodfellow, Alasdair Lambert, Robbie Booth, and Andrew Fagan. 2026. From Code to Questions: Leveraging Generative AI to Support Code Comprehension in Introductory Programming. *Hum-Cent Intell Syst* (2026). doi:10.1007/s44230-026-00150-9

[6] Christian Grévisse. 2024. Comparative Quality Analysis of GPT-Based Multiple Choice Question Generation. In *Applied Informatics*, Hector Florez and Marcelo Leon (Eds.). Springer Nature Switzerland, Cham, 435–447.

[7] Mohammed Hassan and Craig Zilles. 2023. On Students' Usage of Tracing for Understanding Code. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (Toronto ON, Canada) *(SIGCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 129–136. doi:10.1145/3545945.3569741

[8] Theresia Devi Indriasari, Andrew Luxton-Reilly, and Paul Denny. 2020. A Review of Peer Code Review in Higher Education. *ACM Trans. Comput. Educ.* 20, 3, Article 22 (Sept. 2020), 25 pages. doi:10.1145/3403935

[9] Cruz Izu, Cheryl Pope, and Amali Weerasinghe. 2017. On the Ability to Reason About Program Behaviour: A Think-Aloud Study. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education* (Bologna, Italy) *(ITiCSE '17)*. Association for Computing Machinery, New York, NY, USA, 305–310. doi:10.1145/3059009.3059036

[10] Maria Kallia. 2023. The Search for Meaning: Inferential Strategic Reading Comprehension in Programming. In *Proceedings of the 2023 ACM Conference on International Computing Education Research - Volume 1* (Chicago, IL, USA) *(ICER '23)*. Association for Computing Machinery, New York, NY, USA, 1–14. doi:10.1145/3568813.3600135

[11] Amanpreet Kapoor, Tianwei Xie, Leon Kwan, and Christina Gardner-Mccune. 2023. Logistics, Affordances, and Evaluation of Build Programming: A Code Reading Instructional Strategy. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1* (Toronto ON, Canada) *(SIGCSE 2023)*. Association for Computing Machinery, New York, NY, USA, 179–185. doi:10.1145/3545945.3569756

[12] Cazembe Kennedy and Eileen T. Kraemer. 2018. What Are They Thinking? Eliciting Student Reasoning About Troublesome Concepts in Introductory Computer Science. In *Proceedings of the 18th Koli Calling International Conference on Computing Education Research* (Koli, Finland) *(Koli Calling '18)*. Association for Computing Machinery, New York, NY, USA, Article 7, 10 pages. doi:10.1145/3279720.3279728

[13] Cazembe Kennedy and Eileen T. Kraemer. 2019. Qualitative Observations of Student Reasoning: Coding in the Wild. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education* (Aberdeen, Scotland Uk) *(ITiCSE '19)*. Association for Computing Machinery, New York, NY, USA, 224–230. doi:10.1145/3304221.3319751

[14] J. Richard Landis and Gary G. Koch. 1977. The Measurement of Observer Agreement for Categorical Data. *Biometrics* 33, 1 (1977), 159–174. http://www.jstor.org/stable/2529310

[15] Teemu Lehtinen, Lassi Haaranen, and Juho Leinonen. 2023. Automated Questionnaires About Students' JavaScript Programs: Towards Gauging Novice Programming Processes. In *Proceedings of the 25th Australasian Computing Education Conference* (Melbourne, VIC, Australia) *(ACE '23)*. Association for Computing Machinery, New York, NY, USA, 49–58. doi:10.1145/3576123.3576129

[16] Teemu Lehtinen, Aleksi Lukkarinen, and Lassi Haaranen. 2021. Students Struggle to Explain Their Own Program Code *(ITiCSE '21)*. Association for Computing Machinery, New York, NY, USA, 206–212. doi:10.1145/3430665.3456322

[17] Teemu Lehtinen, André L. Santos, and Juha Sorva. 2021. Let's Ask Students About Their Programs, Automatically. In *2021 IEEE/ACM 29th International Conference on Program Comprehension (ICPC)*. 467–475. doi:10.1109/ICPC52881.2021.00054

[18] Raymond Lister, Elizabeth S. Adams, Sue Fitzgerald, William Fone, John Hamer, Morten Lindholm, Robert McCartney, Jan Erik Moström, Kate Sanders, Otto Seppälä, Beth Simon, and Lynda Thomas. 2004. A multi-national study of reading and tracing skills in novice programmers. *SIGCSE Bull.* 36, 4 (June 2004), 119–150. doi:10.1145/1041624.1041673

[19] Laurie Murphy, Renée McCauley, and Sue Fitzgerald. 2012. 'Explain in plain English' questions: implications for teaching. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education* (Raleigh, North Carolina, USA) *(SIGCSE '12)*. Association for Computing Machinery, New York, NY, USA, 385–390. doi:10.1145/2157136.2157249

[20] Sara Nurollahian, Anna N. Rafferty, Noelle Brown, and Eliane Wiese. 2024. Growth in Knowledge of Programming Patterns: A Comparison Study of CS1 vs. CS2 Students. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1* (Portland, OR, USA) *(SIGCSE 2024)*. Association for Computing Machinery, New York, NY, USA, 979–985. doi:10.1145/3626252.3630865

[21] Nishat Raihan, Mohammed Latif Siddiq, Joanna C.S. Santos, and Marcos Zampieri. 2025. Large Language Models in Computer Science Education: A Systematic Literature Review. In *Proceedings of the 56th ACM Technical Symposium on Computer Science Education V. 1* (Pittsburgh, PA, USA) *(SIGCSETS 2025)*. Association for Computing Machinery, New York, NY, USA, 938–944. doi:10.1145/3641554.3701863

[22] André Santos, Tiago Soares, Nuno Garrido, and Teemu Lehtinen. 2022. Jask: Generation of Questions About Learners' Code in Java. In *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 1* (Dublin, Ireland) *(ITiCSE '22)*. Association for Computing Machinery, New York, NY, USA, 117–123. doi:10.1145/3502718.3524761

[23] David H. Smith, Paul Denny, and Max Fowler. 2024. Prompting for Comprehension: Exploring the Intersection of Explain in Plain English Questions and Prompt Writing. In *Proceedings of the Eleventh ACM Conference on Learning @ Scale* (Atlanta, GA, USA) *(L@S '24)*. Association for Computing Machinery, New York, NY, USA, 39–50. doi:10.1145/3657604.3662039

[24] Tian Song, Qinqin Tian, Yijia Xiao, and Shuting Liu. 2024. Automatic Generation of Multiple-Choice Questions for CS0 and CS1 Curricula Using Large Language Models. In *Computer Science and Education. Computer Science and Technology*, Wenxing Hong and Geetha Kanaparan (Eds.). Springer Nature Singapore, Singapore, 314–324.

[25] John Sweller. 1988. Cognitive load during problem solving: Effects on learning. *Cognitive Science* 12, 2 (1988), 257–285. doi:10.1016/0364-0213(88)90023-7

[26] Lasang Jimba Tamang, Zeyad Alshaikh, Nisrine Ait Khayi, Priti Oli, and Vasile Rus. 2021. A Comparative Study of Free Self-Explanations and Socratic Tutoring Explanations for Source Code Comprehension. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education* (Virtual Event, USA) *(SIGCSE '21)*. Association for Computing Machinery, New York, NY, USA, 219–225. doi:10.1145/3408877.3432423

[27] Andrew Tran, Kenneth Angelikas, Egi Rama, Chiku Okechukwu, David H. Smith, and Stephen MacNeil. 2023. Generating Multiple Choice Questions for Computing Courses Using Large Language Models. In *2023 IEEE Frontiers in Education Conference (FIE)*. 1–8. doi:10.1109/FIE58773.2023.10342898

[28] Gavin Witsken, Igor Crk, and Eren Gultepe. 2025. LLMs in the Classroom: Outcomes and Perceptions of Questions Written with the Aid of AI. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Philadelphia, Pennsylvania, USA), Vol. 39. 27698–27705. doi:10.1609/aaai.v39i26.34984