SSR Journal of Engineering and Technology (SSRJET)

Homepage: https://ssrpublisher.com/ssrjet/

Volume 3, Issue 8, 2026 ISSN: 3049-0383

The Effects of Collaborative and Monograde Teaching Strategies on Academic Performance of Computer Science Students in Programming in Colleges of Education in Abia State, Nigeria

Peace A. Frank, Ogori Esau Ogori, Pius John Bassey

Department of Computer and Robotics Education, University of Uyo, Uyo, Akwa Ibom State

Received: *11.06.2026* | ***Accepted:*** *25.08.2026* | ***Published:*** *28.08.2026*

***Corresponding author:** Peace A. Frank

DOI: *10.5281/zenodo.22137842*

Abstract **Original Research Article**

The purpose of this study was to determine the effect of collaborative and monograde teaching strategies on academic performance of Computer Science Students in Programming in Colleges of Education in Abia State, Nigeria. Two specific objectives were identified, two research questions were raised and two null hypotheses were formulated to guide this study. The review of related literature was carried out under three major subheadings namely; theoretical framework, conceptual framework and empirical framework. The study adopted Quasi-experimental design and was carried out in Abia State, Nigeria. The population of the study was 103 NCE 111 Computer Education Students of 2023/2024 Session. The sample size consisted of 103 students at intact class level from two Colleges of Education in Abia State. A researcher developed Programming Skills Performance Test" (PSPT) was used for data collection. The lesson plan and the achievement test were given out to three experts in the University of Uyo for face and content validation. The instrument was tested on 20 Students from Akwa Ibom State College of Education, Afaha Nsit. The internal consistency of the instrument was obtained using Cronbach Alpha which yielded a reliability coefficient of 0.83. The experimental group was taught using collaborative teaching strategy while the control group was taught using monograde teaching strategy. Mean (gain score) was used to answer the research questions while Analysis of Covariance (ANCOVA) was used to test the null hypotheses at 0.05 level of significance. The findings of the study revealed Computer Science Education Students in Programming who taught algorithm design and debugging skills using Collaborative Teaching Strategies had an improved academic performance than those taught using Monograde Teaching Strategies in College of Education, Abia State. It was further revealed that, there was significant difference in the mean academic performance of Computer Science Education Students in programming when taught algorithm design and debugging skills using Collaborative Teaching Strategy as against those taught using Monograde Teaching Strategy in College of Education, Abia State. Based on the results of this finding, it was concluded that collaborative teaching strategies significantly improved academic performance of Computer Science Education Students in programming than using monograde teaching strategies in the College of Education, Abia State, Nigeria. Based on the conclusion, it was recommended that Colleges of Education should design courses that emphasize group work, peer-to-peer learning and collaborative problem-solving activities. This approach can help foster a supportive learning environment and improve student engagement, motivation, and performance across critical programming concepts.

Keywords: Collaborative teaching strategies, monograde teaching strategies, programming skills, academic performance, Computer Science Education students.

1.1 Background of the Study

Collaborative teaching strategy is a pedagogical approach that emphasizes learner-centered instruction, active participation, and knowledge construction through interaction among students, often facilitated by the teacher. In contrast to traditional teacher-centered approaches, collaborative teaching organizes students into small, interactive groups where they work together to solve problems, complete tasks, and create products, with each member contributing meaningfully to the learning process (Johnson and Johnson, 2017). Collaborative teaching strategy which is also known as co-teaching or team teaching is an instructional approach in which two or more educators jointly plan, deliver and assess lessons for the same group of students. This method fosters shared responsibility, integrates diverse expertise and enhances student engagement by leveraging each teacher's strengths to support differentiated and inclusive learning experiences. Collaborative teaching not only focuses on the cognitive development of learners but also seeks to enhance social skills such as communication, cooperation, and conflict resolution, making it particularly valuable in skill-based disciplines such as computer science. In programming education, for example, collaborative learning facilitates peer-to-peer explanation, debugging assistance, and shared exploration of coding solutions, thereby deepening conceptual understanding and practical proficiency. The use of collaborative teaching strategy in higher education has been associated with several positive outcomes. Research indicates that collaborative learning environments can lead to higher academic achievement, improved retention of knowledge and greater student satisfaction compared to individualistic or competitive learning structures (Slavin, 2015).

Furthermore, collaborative teaching strategies are highly effective in fostering essential 21st-century skills, such as communication, collaboration, critical thinking, and creativity, which are vital for success in professional environments that rely on teamwork and interdisciplinary approaches. In the realm of computer science education, the increasing complexity of projects necessitates not only strong technical programming skills but also the ability to effectively engage in collaborative problem-solving, coordination, and project management. This is particularly relevant in online and distributed contexts, where remote teamwork and asynchronous communication introduce real-world challenges (Neill *et al.*, 2016). By creating classroom environments that simulate these circumstances through team-based projects, computer-supported collaborative learning platforms, and structured facilitation, educators provide a realistic learning experience that connects theoretical knowledge with industry practices. For example, when students work together to design and implement a Java application, they participate in requirement analysis, modularization, code integration, and coordination—activities that closely resemble industry workflows while enhancing both individual skills and group collaboration (Gomes de Oliveira Neto and Dobslaw, 2024).

Moreso, while collaborative teaching strategies emphasize teamwork and shared learning experiences, they are implemented within broader instructional settings that influence their effectiveness. One of the most common instructional settings in basic education is the monograde classroom, where teaching and learning occur among students of the same grade level under a standardized curriculum. Understanding the characteristics of the monograde teaching strategy is important because it provides the organizational context within which collaborative instructional techniques can be effectively planned, implemented, and evaluated.

Monograde teaching occurs where a single teacher instructs students of the same grade level using a uniform curriculum and remains a foundational model in primary education, prized for its structured environment and targeted instructional pacing (Risonar and Digamon, 2023). Monograde teaching strategy is defined as a classroom arrangement where a single teacher instructs learners of the same grade level, following a common curriculum and instructional plan, allowing lessons to be delivered at a uniform pace and depth across all students. Risonar and Digamon (2023)

emphasized grade-specific learning objectives, structured classroom management, and a cohesive learning environment tailored to the developmental stage of the learners. Studies show that monograde classrooms can yield comparable or even superior academic outcomes relative to multigrade settings. Furthermore, surveys of primary school teachers in Pakistan indicate that educators overwhelmingly perceive monograde teaching as positive for student learning, underscoring its perceived effectiveness in fostering academic achievement and classroom manageability (Ashfaq *et al*., 2018).

Programming skills are increasingly recognized as essential competencies in the 21st-century knowledge economy, equipping learners with the ability to design, implement and evaluate computational solutions to real-world problems. Beyond syntax proficiency, programming develops computational thinking a problem-solving approach involving decomposition, abstraction, algorithmic design and debugging that supports learning across multiple disciplines (Marín-Marín *et al.,* 2021; Qian and Choi, 2023). Studies have shown that integrating programming into formal education enhances students' logical reasoning, creativity and perseverance while fostering collaborative problem-solving abilities vital in modern workplaces (Melro *et al*., 2023; Mills *et al.*, 2024). Furthermore, programming experience improves students' adaptability to emerging technologies and contributes to digital literacy, a core skill for employability in technology-driven sectors (Shen *et al.,* 2022). The rise of block-based environments such as Scratch and web-based platforms like Code.org has lowered entry barriers, making programming accessible to learners of varying backgrounds and ages while boosting engagement and motivation (Choi and Choi, 2024). As industries increasingly rely on automation, data processing and artificial intelligence, the cultivation of programming skills in educational settings ensures that learners are not only consumers of technology but also capable creators, innovators and problem solvers in an interconnected world. The following variables represent programming skills: algorithm design, debugging skills, syntax accuracy, control structures and problem-solving skills.

Algorithm design is a core aspect of programming skills performance, as it involves creating step-by-step procedures to solve computational problems efficiently. Well-designed algorithms enhance program logic, reduce execution time and optimize resource usage, thereby improving overall software quality (Lamprou and Repenning, 2020). Mastery of algorithm design equips programmers with the ability to translate problem statements into structured solutions before coding, minimizing syntax and logical errors (Brusilovsky *et al*., 2021). In educational contexts, strong algorithm design skills are linked to improved computational thinking and problem-solving abilities, which are critical for tackling complex programming tasks (Omar *et al.,* 2021). Conversely, poor algorithm design can lead to inefficient, error-prone, and difficult-to-maintain code. Thus, cultivating algorithm design competence through practice, feedback, and problem-based learning is vital for enhancing programming performance and preparing students for real-world software development challenges. Algorithm design will only lead to functional programs if debugging skills are applied.

Debugging skills in programming is the ability to systematically identify, analyze and correct errors in code is are essential for developing reliable, functional and efficient software. Effective debugging involves understanding program logic, interpreting error messages, using diagnostic tools and applying testing strategies to locate and resolve faults (Fitzgerald *et al*., 2018). Research shows that novice programmers often find debugging challenging due to limited problem decomposition abilities and difficulty tracing program execution, which can hinder learning and productivity (Prather *et al.,* 2017). Targeted instruction that integrates debugging activities into programming courses has been found to improve students' code comprehension, problem-solving skills, and confidence (Baker *et al.,* 2020). Moreover, in professional contexts, strong debugging skills reduce development time, enhance software quality, and lower maintenance costs (Sharma and Kumar, 2021). As debugging is both a cognitive and practical process, fostering it through structured practice,

error analysis, and reflective thinking is critical for preparing learners to meet academic and industry demands in software development. Accurate debugging skills is only possible if syntax accuracy is correctly implemented.

Statement of the Problem

Programming is a core competency in Computer Science Education, yet students in Colleges of Education and other tertiary institutions continue to struggle with essential programming skills such as algorithmic implementation, method creation, object-oriented programming concepts, control structures, and syntactic accuracy (Bennedsen and Caspersen, 2019; Luxton-Reilly *et al.*, 2021). This recurring weakness not only hinders students' academic progress but also affects their preparedness for real-world computing tasks and software development careers (OECD, 2021; Gomes de Oliveira Neto and Dobslaw, 2024). Despite the sustained use of traditional monograde teaching strategies, where students of similar proficiency levels receive uniform instruction, many learners still fail to attain the expected level of programming competence (Risonar and Digamon, 2023). Moreover, although monograde instruction provides structured pacing and facilitates classroom management, it often offers limited opportunities for peer-assisted learning, collaborative problem-solving and the exchange of diverse perspectives that are essential for mastering complex programming concepts (Ashfaq *et al*., 2018; Johnson and Johnson, 2022).

Emerging evidence suggests that collaborative teaching strategies enhance students' engagement, deepen conceptual understanding, improve problem-solving abilities, and strengthen programming skills through active interaction, peer support, and shared learning experiences (Gomes de Oliveira Neto and Dobslaw, 2024; Hmelo-Silver, 2021). Collaborative learning has also been associated with improved critical thinking, communication, and knowledge retention, particularly in computing and STEM education (UNESCO, 2023; OECD, 2025). However, there is insufficient empirical evidence directly comparing collaborative and monograde teaching strategies with respect to programming skills development among Computer Science Education students in Colleges of Education in Nigeria, particularly in Abia State. This paucity of comparative studies leaves educators and curriculum planners without sufficient evidence to determine whether the continued use of monograde instruction or the adoption of collaborative teaching strategies would more effectively improve students' programming competence (NCCE, 2024).

Furthermore, the persistent low performance in programming courses among Computer Science students, coupled with the need to produce graduates who possess practical programming competencies required for the digital economy, underscores the necessity for evidence-based instructional innovations (Onifade *et al.*, 2025; World Economic Forum, 2023). Therefore, this study seeks to determine the effect of collaborative and monograde teaching strategies on Computer Science Education students' programming skills performance in Colleges of Education in Abia State, Nigeria.

Purpose of the Study

The main purpose of this study is to determine the effects of collaborative and monograde teaching strategies on academic performance of Computer Science Students in Programming in Colleges of Education in Abia State, Nigeria. Specifically, the study seeks to:

1. Determine the difference in mean academic performance of Computer Science Education Students in programming when taught algorithm design using collaborative teaching strategy as against those taught using monograde teaching strategy in Colleges of Education, Abia State.
2. Ascertain the difference in mean academic performance of Computer Science Education Students in programming when taught debugging skills using collaborative teaching strategy as against those taught using monograde teaching strategy in Colleges of Education, Abia State.

Research Questions

The following research questions were raised to guide the study.

1. What is the difference in the mean academic performance of Computer Science Education Students in programming when taught algorithm design using collaborative teaching strategy as against those taught using monograde teaching strategy in Colleges of Education, Abia State?
2. What is the difference in the mean academic performance of Computer Science Education Students in programming when taught debugging skills using collaborative teaching strategy as against those taught using monograde teaching strategy in Colleges of Education, Abia State?

Research Hypotheses

The following research hypotheses were formulated to guide the study:

$H_{O1:}$ There is no significant difference in the mean academic performance of Computer Science Education Students in programming when taught algorithm design using collaborative teaching strategy as against those taught using monograde teaching strategy in Colleges of Education, Abia State.

$H_{O2:}$ There is no significant difference in the mean academic performance of Computer Science Education Students in programming when taught debugging skills using collaborative teaching strategy as against those taught using monograde teaching strategy in Colleges of Education, Abia State.

Research Methods

The study adopted a quasi-experimental pre-test post-test non-equivalent group design, using intact classes because random assignment of students was not possible without disrupting the academic programmes of the participating institutions. The study was conducted in two Colleges of Education in Abia State: Abia State College of Education (Technical), Arochukwu, and College of Education, Akwete. The population comprised 103 NCE III Computer Education students, with 54 students from Arochukwu and 49 from Akwete. The entire population was used as the sample, with the former serving as the experimental group and the latter as the control group. Data were collected using a researcher-developed 50-item Programming Skills Performance Test (PSPT) covering algorithm design and debugging skills. The instrument was validated through face and content validation by three experts, while reliability testing was conducted with 20 NCE III Computer Education students outside the study area, yielding reliability coefficients of 0.80 and 0.83. During the experiment, the experimental group was taught using the collaborative teaching strategy, while the control group received the monograde teaching strategy. Both groups were taught by their course lecturers, who were briefed and guided by the researcher, and the treatment lasted for seven weeks. A pre-test was administered to both groups before the treatment, followed by a post-test after the seven-week intervention. The course lecturers administered the instruments, while the researcher retrieved, scored, and organized the data for analysis. Mean gain scores were used to answer the research questions, while Analysis of Covariance (ANCOVA) was employed to test the null hypotheses at the 0.05 level of significance. The decision rule stated that the group with the higher mean gain performed better, while a null hypothesis was rejected when $p < 0.05$ and retained when $p > 0.05$. Ethical considerations, including informed consent, voluntary participation, confidentiality, avoidance of plagiarism, and use of genuine data, were also observed.

Results and Discussion

Research Question 1: What is the difference in the mean academic performance of Computer Science Education Students in programming when taught algorithm design using collaborative teaching strategy as against those taught using monograde teaching strategy in College of Education, Abia State?

Table 1: The summary of Mean Scores of Academic Performance of Computer Education Students in Programming when taught Algorithm Design using Collaborative Teaching Strategy as against those taught using Monograde Teaching Strategy.

Groups	N	Pretest $\bar{X}_1$	Posttest $\bar{X}_2$	Mean Gain $\bar{X}$	Mean Gain Difference
Collaborative Teaching	54	5.13	14.38	9.25	4.32
Monograde Teaching	49	5.15	10.08	4.93	

Source: Field Survey (2025)

The data presented in Table 1 showed the comparison of academic performance among Computer Education Students in programming, using two teaching strategies which indicated that for students who were taught Algorithm Design using collaborative teaching strategy, the mean increases from 5.13 in pre-test to 14.38 in post-test, giving mean gain of 9.25. Furthermore, for students who were taught Algorithm Design using monograde teaching strategy, the mean increases from 5.15 in pre- test to 10.08 in post-test, giving mean gain of 4.93. The difference in the mean gain for students who were taught Algorithm Design using collaborative teaching strategy was greater than the mean gain for students who were taught Algorithm Design using monograde teaching strategy by 4.32. It is therefore concluded that, students who were taught Algorithm Design using collaborative teaching strategies performed better than those taught using monograde teaching strategy in College of Education, Abia State.

Research Question 2: What is the difference in the mean academic performance of Computer Science Education Students in programming when taught debugging skills using collaborative teaching strategy as against those taught using monograde teaching strategy in College of Education, Abia State?

Table 2: The summary of Mean Scores of Academic Performance of Computer Education Students in Programming when taught Debugging Skills using Collaborative Teaching Strategy as against those taught using Monograde Teaching Strategy.

Groups	N	Pretest $\bar{X}_1$	Posttest $\bar{X}_2$	Mean Gain $\bar{X}$	Mean Gain Difference
Collaborative Teaching	54	5.36	14.02	8.66	3.15
Monograde Teaching	49	5.06	10.57	5.51	

Source: Field Survey (2025)

The data presented in Table 2 showed the comparison of academic performance among Computer Education Students in programming, using two teaching strategies which indicated that for students who were taught Debugging Skills using collaborative teaching strategy, the

mean increases from 5.36 in pre-test to 14.02 in post-test, giving mean gain of 8.66. Furthermore, for students who were taught Debugging Skills using monograde teaching strategy, the mean increases from 5.06 in pre- test to 10.57 in post-test, giving mean gain of 5.51? The difference in the mean gain for students who were taught Debugging Skills using collaborative teaching strategy exceeds the mean gain for students who were taught Debugging Skills using monograde teaching strategy by 3.15. It is therefore concluded that, students who were taught Debugging Skills using collaborative teaching strategy performed better than those taught using monograde teaching strategy in College of Education, Abia State.

Research Hypothesis 1**:** There is no significant difference in the mean academic performance of Computer Science Education Students in programming when taught algorithm design using collaborative teaching strategy as against those taught using monograde teaching strategy.

Table 3: Summary of ANCOVA Analysis Test of Significance for the Effect of Collaborative and Monograde Teaching Strategies on Students Academic Performance in Programming when taught Algorithm Design using Collaborative Teaching Strategy as against those taught using Monograde Teaching Strategy.

Source	Type 111 Sum of Squares	df	Mean Square	t-value	p-value	Decision
Corrected Model	538.66[a]	2	269.33	25.79	.00	S
Intercept	4057.29	1	4057.29	388.50	.00	S
PreTest1	38.13	1	38.13	3.65	.06	NS
Strategies	501.51	1	501.51	48.02	.00	S
Error	1096.55	105	10.44			
Total	17891.00	108				
Corrected Total	1635.21	107				

S = Significant, NS = Not Significant.

Source: Field Survey (2025).

The ANCOVA analysis presented in Table 3 revealed significant findings regarding the effect of Collaborative and Monograde teaching strategies on students' academic performance in programming when taught algorithm design using collaborative teaching strategy against those taught using monograde teaching strategy. The overall model displayed a substantial effect, with a t-value of 25.79 and a p-value of .00, indicating a statistically significant difference in academic performance attributable to the teaching strategies employed. Furthermore, the specific effect of the teaching strategies yielded a t-value of 48.02 with a p-value of .00, confirming that students who were taught using the Collaborative teaching strategy significantly outperformed those who were taught using Monograde teaching strategy when taught algorithm design in programming. This result indicates a meaningful difference in the mean academic performance scores of Computer Science Education students, suggesting that

teaching strategies significantly influence student academic performance; thus, the null hypothesis of no significant difference was rejected. In contrast, the pre-test variable (PreTest1) produced a t-value of 3.65 and a p-value of .06, indicating a trend toward significance but failing to meet the conventional alpha threshold of 0.05. Consequently, this suggests that there is no significant difference in the mean academic performance of Computer Science Education students based on Students pre-test scores in College of Education, Abia State. Therefore, the null hypothesis of significant difference was retained.

Research Hypothesis 2: There is no significant difference in the mean academic performance of Computer Science Education Students in programming when taught debugging skills using collaborative teaching strategy as against those taught using monograde teaching strategy.

Table 4: Summary of ANCOVA Analysis Test of Significance for the Effect of Collaborative and Monograde Teaching Strategies on Students Academic Performance in Programming when taught Debugging Skills using Collaborative Teaching Strategy as against those taught using Monograde Teaching Strategy.

Source	Type 111 Sum of Squares	df	Mean Square	t	Sig.	Decision
Corrected Model	352.84[a]	2	176.42	20.34	.00	S
Intercept	4660.70	1	4660.70	537.29	.00	S
PreTest1	31.18	1	31.18	3.59	.06	NS
Strategies	312.65	1	312.65	36.04	.00	S
Error	910.82	100	9.11			
Total	17667.00	103				
Corrected Total	1263.66	102				

S = Significant, NS = Not Significant.

Source: Field Survey (2025)

The ANCOVA analysis presented in Table 4 revealed significant findings regarding the effect of Collaborative and Monograde teaching strategies on students' academic performance in programming when taught debugging skills using collaborative teaching strategy against those taught using monograde teaching strategy. The overall model displayed a substantial effect, with a t-value of 20.34 and a p-value of .00, indicating a statistically significant difference in academic performance attributable to the teaching strategies employed. Furthermore, the specific effect of the teaching strategies yielded a t-value of 36.04 with a p-value of .00, confirming that students who were taught using the Collaborative teaching strategy significantly outperformed those who were taught using Monograde teaching strategy when taught debugging skills in programming. This result indicates a meaningful difference in the mean academic performance scores of Computer Science Education students, suggesting that teaching strategies significantly influence student academic performance; thus, the null

hypothesis of no significant difference was rejected. In contrast, the pre-test variable (PreTest1) produced a t-value of 3.59 and a p-value of .06, indicating a trend toward significance but failing to meet the conventional alpha threshold of 0.05. Consequently, this suggests that there is no significant difference in the mean academic performance of Computer Science Education students based on Students pre-test scores in College of Education, Abia State. Therefore, the null hypothesis of significant difference was retained.

Discussion of Findings

The finding of the study revealed that the two modes of teaching strategies in programming when taught Algorithm Design using Collaborative Teaching Strategy as against those taught using Monograde Teaching Strategy differ in their effects on academic performance. The finding regarding Research Question 1 showed that the post-test mean score of students in the experimental group that taught Algorithm Design using collaborative teaching strategy perform better than the post-test mean score of students in the control group taught Algorithm Design using monograde teaching strategy. The difference in the mean gain for students who taught Algorithm Design using collaborative teaching strategy was greater than the mean gain for students who taught Algorithm Design using monograde teaching strategies. This result clearly showed that teaching strategies using collaborative teaching strategy is more effective in enhancing students' academic performance in Algorithm Design. Moreover, findings show the rejection of Null Hypothesis 1 on the basis that the analysis of covariance (ANCOVA) sig-value (p-value) on Table 6 was below 0.05. This means that there is a significant effect between collaborative and monograde teaching strategies on students' academic performance in programming when taught algorithm design using collaborative teaching strategy against those taught using monograde teaching strategy. This could be as a result of the fact that algorithm design is a core aspect of programming skills performance, as it involves creating step-by-step procedures to solve computational problems efficiently. This is because well-designed algorithms enhance program logic, reduce execution time and optimize resource usage, thereby improving overall software quality. The result of this finding is in consonance with the findings of Patel and Singh (2021) who found that the effects of early exposure to algorithm optimization principles on programming performance, particularly within the contexts of competitive programming and real-world software projects. This finding underscored that early engagement with algorithmic principles significantly enhances both theoretical understanding and practical application, leading to better outcomes in programming challenges and development tasks. The result of the study also agreed with the assertion of Brusilovsky *et al.* (2021) who assert that students with well-developed algorithm design skills demonstrated significantly fewer syntax errors and produced more efficient code, thereby highlighting the critical importance of integrating algorithm design training into computer science curriculum. The finding is in agreement with Soliman and Guetl, (2020) who found that students exposed to a variety of algorithm design strategies exhibited significantly greater adaptability in addressing novel problems, highlighting the need for educational approaches that foster flexible thinking and problem-solving skills in the face of technological change. However, this finding revealed that student taught algorithm design using collaborative teaching strategy perform better than students taught algorithm design using monograde teaching strategy.

The finding of the study revealed that the two modes of teaching strategies in programming when taught Debugging Skills using Collaborative Teaching Strategy as against those taught using Monograde Teaching Strategy differ in their effects on academic performance. The finding regarding Research Question 2 showed that the post-test mean score of students in the experimental group that taught Debugging Skills using collaborative teaching strategy perform better than the post-test mean score of students in the control group taught Debugging Skills using monograde teaching strategy. The difference in the mean gain for students who taught Debugging Skills using collaborative teaching strategy was greater than the mean gain

for students who taught Debugging Skills using monograde teaching strategies. This result clearly showed that teaching strategies using collaborative teaching strategy is more effective in enhancing students' academic performance in Debugging Skills. Moreover, finding showed the rejection of Null Hypothesis 2 on the basis that the analysis of covariance (ANCOVA) sig-value (p-value) on Table 7 was below 0.05. This means that there is a significant effect between collaborative and monograde teaching strategies on students' academic performance in programming when taught debugging skills using collaborative teaching strategy against those taught using monograde teaching strategy. This could be as a result of the fact that debugging skills in programming is the ability to systematically identify, analyze and correct errors in code are essential for developing reliable, functional and efficient software. This is because effective debugging involves understanding program logic, interpreting error messages, using diagnostic tools and applying testing strategies to locate and resolve faults. The finding of the study adds to the assertion by Scherer and Watters (2021) who affirmed that collaborative and peer-based debugging methods significantly improved student performance and engagement. Thinking along this direction, Tharp and Hollingsworth (2019) summitted that student who received instruction in debugging strategies significantly improved their coding performance and developed a more systematic approach to problem-solving. In the same vein, the finding of this study agrees with the finding of Guzdial and Ericson (2016) who found that strong debugging skills not only enhance novice programmers' ability to identify and resolve errors but also foster a deeper understanding of programming concepts, ultimately contributing to improved overall programming proficiency.

Conclusion

Based on the findings of the study, it was concluded that the Collaborative Teaching Strategy was more effective than the Monograde Teaching Strategy in improving the academic performance of Computer Science Education students in programming in Colleges of Education, Abia State. Specifically, students who were taught Algorithm Design using the Collaborative Teaching Strategy demonstrated improved academic performance compared with those taught using the Monograde Teaching Strategy, and the observed difference in their mean academic performance was statistically significant. Similarly, the study concluded that the use of the Collaborative Teaching Strategy significantly improved students' academic performance in Debugging Skills compared with the Monograde Teaching Strategy. Students exposed to collaborative teaching performed better than those taught through the monograde approach, with a statistically significant difference in their mean academic performance. Thus, the findings indicate that Collaborative Teaching Strategy is an effective instructional approach for enhancing students' learning and performance in key programming skills, particularly Algorithm Design and Debugging Skills.

Recommendations

Based on the findings of the study, the following recommendations were made:

1. Computer Education lecturers should adopt Collaborative Teaching Strategy when teaching programming, particularly Algorithm Design and Debugging Skills, because it has been found to improve students' academic performance compared with the Monograde Teaching Strategy.
2. Colleges of Education should organize workshops and seminars to train Computer Education lecturers on the effective application of Collaborative Teaching Strategy in programming instruction.
3. Curriculum planners and educational authorities should incorporate collaborative teaching approaches into Computer Education and programming curricula to encourage active participation and improve students' learning outcomes.
4. College management should provide adequate instructional resources and facilities that support collaborative

learning activities in programming classes, including appropriate computer laboratories, programming software, and other relevant teaching materials.

5. Computer Education lecturers should encourage students to work collaboratively during programming lessons by engaging them in group-based activities, problem-solving tasks, algorithm development, and debugging exercises to strengthen their programming skills and academic performance.

REFERENCES

Baker, R., Mladenić, D. and Takači, D. (2020). Teaching debugging strategies in introductory programming courses. *Education and Information Technologies, 25*(6), 5363–5380.

Bennedsen, J. and Caspersen, M. E. (2019). Failure rates in introductory programming: A systematic review update. *ACM Transactions on Computing Education*.

Brusilovsky, P., Sosnovsky, S. and Yudelson, M. (2021). Adaptive support for learning algorithm design. *International Journal of Artificial Intelligence in Education, 31*(2), 245–270.

Choi, W. C. and Choi, I. C. (2024). *The influence and relationship between computational thinking, learning motivation, attitude, and achievement of Code.org in K-12 programming education.*

Fitzgerald, S., Lewandowski, G., McCauley, R., Murphy, L. and Simon, B. (2018). Debugging: Finding, fixing, and understanding bugs. *ACM Inroads, 9*(1), 52–58.

Gomes de Oliveira Neto, F. and Dobslaw, F. (2024). *Collaborative learning in software engineering education: Enhancing student engagement and skill development* (1st Ed., pp. 1–180). New York, NY.

Gomes de Oliveira Neto, F. and Dobslaw, F. (2024). *Collaborative learning in software engineering education: Enhancing student engagement and skill development* (1st Ed., pp. 1–180). New York, NY.

Guzdial, M. and Ericson, B. (2016). *Introduction to Computing and Programming with Python: A Multimedia Approach*. Pearson (2nd Ed., pp. 25–37). New York, NY.

Hmelo-Silver, C. E. (2021). Collaborative learning and problem-based learning in higher education. In *The Cambridge Handbook of the Learning Sciences*.

Johnson, D. W. and Johnson, R. T. (2017). The use of cooperative procedures in teacher education and professional development. *Journal of Education for Teaching, 43*(3), 284–295.

Johnson, D. W. and Johnson, R. T. (2022). *Cooperative learning: The foundation for active learning*. *Active Learning in Higher Education*.

Lamprou, A. and Repenning, A. (2020). Teaching algorithmic thinking through game design: A workshop for K-12 teachers. *Education and Information Technologies, 25*(4), 2971– 2991.

Luxton-Reilly, A., Simon, Al and Bluwi, I. (2021). *Introductory programming: A systematic literature review*. *ACM Transactions on Computing Education*.

Marín-Marín, J. A., Esteve-Mon, F. and De la Torre, I. (2021). Computational thinking and learning approaches in programming education. *Frontiers in Psychology, 12*(7), 134–156.

Melro, A., Costa, N. and Ferreira, F. (2023). Programming education and problem-solving skills: A systematic review. *Education and Information Technologies, 28*(5), 5561– 5582.

Mills, K. A., Cope, J., Scholes, L. and Rowe, L. (2024). Coding and computational thinking across the curriculum: A review of educational outcomes. *Review of Educational Research, 94*(3), 395–430.

National Commission for Colleges of Education (NCCE). (2024). *Nigeria Certificate in*

Education Minimum Standards for Computer Science Programme.

Neill, C. J., DeFranco, J. F. and Sangwan, R. S. (2016). Improving collaborative learning in online software engineering education. *European Journal of Engineering Education, 42*(6), 594–609.

OECD. (2021). *The State of Higher Education 2021*. Paris: OECD Publishing.

OECD. (2025). *Education at a Glance 2025: OECD Indicators*. Paris: OECD Publishing.

Omar, M., Al-Dosari, H. and Al-Samarraie, H. (2021). The role of algorithm design in enhancing computational thinking among novice programmers. *Computers & Education, 16*(4), 104–125.

Omar, M., Ghani, I. and Ahmad, R. (2021). Algorithm design as a cognitive scaffold in programming education. *Journal of Computer Assisted Learning, 37*(6), 1604–1617.

Onifade, O. J., Gambo, O. O., Ogunleye, A. T. and Ajayi, P. O. (2025). Assessing the relevance of the undergraduate Computer Science curriculum of Nigerian universities: Insights from industry. *Education and Information Technologies*.

Patel, R. and Singh, M. (2021). Algorithm optimization techniques for improved programming performance. *Journal of Computer Science Education, 13*(2), 45–60.

Prather, J., Pettit, R., McMurry, K., Peters, A., Homer, J. and Cohen, M. (2017). Metacognitive difficulties faced by novice programmers in automated assessment tools. *ACM Transactions on Computing Education, 17*(1), 1–23.

Qian, M. and Choi, I. (2023). A meta-analysis of the effects of programming education on computational thinking development. *Computers & Education, 19*(5), 104–195.

Risonar, J. M. and Digamon, M. M. (2023). Monograde teaching practices and classroom effectiveness in primary education. *International Journal of Educational Research*.

Scherer, D. and Watters, J. (2021). Teaching methods for debugging: Impacts on student performance and collaboration in programming courses. *Computers & Education, 16*(7), 104–178.

Sharma, V. and Kumar, R. (2021). Efficient debugging practices for improving software quality. *International Journal of Computer Applications, 18*(48), 1–6.

Shen, J., Lu, Y., Hu, X. and Zhang, H. (2022). Digital literacy and programming education: Preparing students for the future workforce. *Education and Information Technologies, 27*(8), 455–473.

Slavin, R. E. (2015). *Cooperative Learning: Theory, Research, and Practice* (2nd Ed., pp. 24–56). Routledge.

Soliman, M. and Guetl, C. (2020). Enhancing creativity in programming through diverse algorithmic problem-solving approaches. *Journal of Educational Computing Research, 58*(6), 110–132.

Tharp, M. M. and Hollingsworth, J. (2019). Investigating the impact of debugging instruction on novice programmers. *Proceedings of the ACM Conference on Innovation and Technology in Computer Science Education* (2nd Ed., pp. 137–143). ACM.

UNESCO. (2023). *Global Education Monitoring Report 2023: Technology in Education*. Paris: UNESCO.