Work In Progress: Impact of Collaborative Learning Strategies on Anxiety Reduction in Introductory Programming Courses

Dr. Joseph Ekong, Western New England University

Dr. Joseph Ekong is an Assistant Professor in the Department of Industrial Engineering and Engineering Management at Western New England University. He received his Ph.D. in Industrial and Systems Engineering from Auburn University, and M.Sc. in Industrial and Systems Engineering from Georgia Institute of Technology. His research and teaching interests include advanced manufacturing systems, robotics and automation systems, data analytics, and engineering education.

Dr. Arnab A. Purkayastha, Western New England University

Arnab A Purkayastha is an Assistant Professor in the Electrical and Computer Engineering department at Western New England University, Massachusetts. He received his PhD in the year 2021 from the University of North Carolina at Charlotte. His research interests and activities lie in the recent advances in High Performance Computing and Machine Learning fields, including system level integration both at the cloud and edge.

Dr. Gladys Ekong

Dr. Gladys Ekong is an Associate Professor in the Department of Pharmaceutical and Administrative Sciences, College of Pharmacy and Health Sciences, Western New England University (WNE). Her research work focuses on behavioral and sociocultural aspects of chronic disease management and prevention. She has published in various peer-reviewed journals on Motivational Interviewing, provider-patient communication, medication use, and behavioral interventions. Her teaching interests include research methods & data analytics, pharmacy practice management and healthcare communications.

Work In Progress: Impact of Collaborative Learning Strategies on Anxiety Reduction in Introductory Programming Courses

Abstract

This work in progress study investigates the effectiveness of a teaching intervention focused on collaborative learning strategies, specifically, write-pair-share (WPS), and vertical non-permanent surfaces (VNPS), in reducing students' anxiety with learning introductory programming courses. An introductory programming course is an important course for first-year engineering students. The ability to write programs to solve real-world problems is a vital skill for engineers. First-year students without prior programming experience may encounter challenges in introductory programming courses, which may lead to increased anxiety and academic hurdles. This study was implemented among students in a first-year programming course at a private university. Students enrolled in the introductory programming course were given modeling problems during the semester to assess their progress in developing programming solutions. In each modeling problem, students were required to submit a written solution to the problem, detailing their solution approach, and a coded solution to the problem using MATLAB as the programming language. The validated survey, "Attitudes Toward Mathematics Inventory (ATMI)" was revised to focus on programming rather than mathematics. Students completed the revised version of the ATMI survey at the beginning of the semester and at the end of the semester. Study data was analyzed using descriptive statistics and t-tests to determine significant improvement in students' anxiety and confidence in their programming skills. Findings from this study suggest that effective teaching methods may improve students' anxiety, confidence, and engagement in similar programming courses.

Keywords: First-Year Program, Introductory Programming, Collaborative Learning, Educational interventions, Vertical Non-Permanent Surfaces, Learning Anxiety

Introduction

Introductory programming is an essential aspect of an engineer's education. Engineers are usually tasks with solving complex and complicated real-world problems. To successfully solve such problems, an in-depth understanding of how to develop and utilize mathematical and computational models to solve problems is vital. Studies have shown that explicitly teaching first-year engineering students how to develop models to solve problems has several benefits [1, 2].

Even though programming is a crucial aspect of the engineering curriculum, many engineering students find introductory programming to be a difficult course to take. Several factors have been identified as contributing to students' struggle with learning how to program. Such factors include self-efficacy, mental models, and previous experience with programming [2]. Previous experience with programming has been identified as a major factor influencing students' performance in introductory programming courses. Students with previous programming exposure tend to be more confident about taking introductory programming courses, while students with no previous programming experience are more likely to deal with learning anxiety issues which may impact their performance in the course.

It has also been pointed out that students who do not do well in introductory programming classes tend to drop out of programming focused majors, with studies suggesting the dropout and failure

rates in introductory programming courses could be as high as 30 percent [2, 3]. Unfortunately, many students taking an introductory programming course do not have prior programming experience before college. For such students, their learning pace is likely to be slower than for students with prior programming experience or exposure. As such it becomes vital to ensure that the learning process and environment promotes students' ability to gain self-efficacy while taking introductory programming courses. Teaching and learning methods that foster classroom environment and culture that promote students' engagement has been credited with increasing retention in computer science programs [4, 5]. An example of such teaching and learning methods is collaborative learning.

Collaborative learning is a teaching and learning method that encourages learners to learn by working together to perform a task in groups small enough to guarantee everyone participates in the learning process [6 - 9]. Collaborative learning has been shown to be effective in engaging students in the learning process and helping learners develop self-efficacy and reduce learning anxiety issues during the learning process. It has also been demonstrated that collaborative learning promotes a sense of belonging and community for undergraduate engineering students and also leads to students' persistence in the learning process [10, 11].

If properly designed and implemented in the teaching and learning process for introductory programming courses, collaborative learning strategies could offer the opportunity of leveraging the strengths and experiences of students with prior programming experience to benefit students being introduced to programming for the first time, and should help with ramping up the overall knowledge of the class. The teaching methods that were applied in this study are write-pair-share (WPS), and vertical non-permanent surfaces (VNPS).

WPS is a teaching technique commonly used in teaching students how to write and communicate their ideas. In this approach, students are encouraged to formulate their thoughts in writing and then engage in oral interaction with a peer. VNPS on the other hand is a teaching technique that involves students leaving their seats and participating in a group setting while standing at a vertical non-permanent surface like a whiteboard to accomplish a task. An added advantage of the VNPS approach is that it provides students the opportunity of seeing the work done by other groups, thereby gaining insights into ideas they may decide to adopt. It has been suggested that the use of vertical non-permanent surfaces for group tasks promotes greater thinking, classroom participation, discussion, persistence, and knowledge mobility [12].

The overarching purpose of this study is to investigate the effectiveness of incorporating collaborative teaching and learning strategies on students' engagement and anxiety while taking introductory programming courses. Hence, the following research questions were investigated:

Research question #1: Do collaborative learning strategies like WPS and VNPS impact students' confidence and reduce anxiety with learning introductory programming?

Research Question #2: For students without prior exposure to computer programming, how effective was the intervention in reducing students' anxiety and improving student confidence with introductory programming?

Methods

Setting and Participants

This study involves students enrolled in an introductory programming course for engineers at a private university. The course is required for all engineering students in the five engineering departments (biomedical, civil and environmental, electrical and computer, industrial, and mechanical engineering) in the College of Engineering. MATLAB was used as the programming language for the course. The study consent form was shared with the students during the recruitment event and students were asked to voluntarily enroll in the research study. The teaching intervention was offered to all the students; however, the research findings focus on consenters only. The research protocol indicated that the data assessor will not disclose the students who consent to the study until the end of the semester after the final grades are released. The research protocol was approved by the Institutional Review Board (IRB) at the study setting.

Study Protocol and Measures

The introductory programming course focuses on teaching engineering students the fundamental concepts involved in developing effective computer programs to solve engineering problems. Learning objectives for the course include:

1. Demonstrate an ability to translate an engineering problem into a set of logical steps necessary to arrive at a feasible solution.
2. Demonstrate an ability to utilize knowledge of mathematics and computer programming concepts such as MATLAB built-in functions, sequences, selection and repetition structures, and user-defined functions when solving computational problems
3. Demonstrate understanding of how to use proper techniques in presenting engineering information in graphical form
4. Demonstrate the ability to use MATLAB to solve problems consisting of non-numerical data.

Prior to implementing the teaching interventions, the students completed a pre-survey to assess their programming knowledge, skills, and confidence (Appendix A). Students involved in this study were taught by two instructors that incorporated the two collaborated learning strategies (Write Pair Share and Vertical Non-Permanent Surfaces) in their teaching approach to reduce students' anxiety in introductory programming courses. Teaching Assistants (TAs) were available to provide support during collaborative learning activities. In addition, self-paced learning was supported using on-demand recorded lecture videos, and video solutions to problem sets, that enable students to review the lectures after class sessions at their learning pace. This support was provided to ameliorate learning anxiety that may arise from difficulties with the course content and teaching pace. Both instructors used similar assigned work and structure for the course, however there were slight variations in class activities and presentations.

The WPS activities were introduced early in the course and served to prepare students for the vertical non-permanent surfaces to be introduced later in the course. For the WPS activities, a programming problem is presented to the students to work on in class. All the students were provided with 12 inches by 15 inches dry erase white board and markers to develop their solution ideas individually. Students then discussed their solution approach in pairs and shared their ideas

with the class. During these activities, the instructors and teaching assistants were available to support students in need of guidance. The goals for the WPS activities were: 1) to enhance students' engagement with the class at the early stages and also encourage students to become comfortable with documenting their solution ideas and discussing these ideas, and 2) to help the instructors and TAs to identify students who may be struggling and provide support early in the semester. In addition, it is expected that these activities would ease learning anxieties for students at the early phase of learning programming and enhance their confidence by being engaged and able to perform the initial basic tasks on their own.

The Vertical Non-Permanent Surfaces activities were introduced after the students were comfortable with the basic programming concepts. The problem sets used for the VNPS activities are more complex and reflect real-world situations. Students are tasked with working on these problems in groups of three. Students first develop their solution approach individually. They then meet in their group to discuss their individual solution and agree on a solution approach to adopt as a group. To ensure all the members of the group are engaged in the learning process, each member is assigned a specific task. One member is to verbally explain the solution steps, while another member writes the MATLAB code required to execute the program. The last member will then report out the group's solution and explain it to the instructor or TA. Students' tasks will be rotated across different VNPS activities to ensure all students play various roles.

During the initial VNPS activities, the groups were created randomly using a software for random group allocation. However, as the semester progressed, the grouping approach was altered to create groups based on a brief history of their performance in the course using their grades from quizzes and exams. The objective of this grouping approach was to pair students with different performance levels within a group to enhance the possibility of students learning from each other.

Data Collection & Analysis

The validated survey, "Attitudes Toward Mathematics Inventory (ATMI)" [13] was revised to focus on programming rather than mathematics. The revised survey (Appendix A) was applied to evaluate students' learning anxiety and engagement with the introductory programming course. The survey was completed at the beginning of the semester and at the end of the semester to assess the change in students' attitudes in the following domains: enjoyment of introductory programming, motivation, self-confidence, and value of programming based on the intervention strategies applied in the teaching and learning process. The research data was analyzed using descriptive statistics to summarize study variables and t-test statistics are used to identify changes in target outcomes after the intervention (post-intervention). For the secondary research objective, a t-test was applied to compare average scores on the overall ATMI score and Confidence scores for 1) programming skills and 2) solving open-ended programming problems among students with prior exposure to computer programming ("Yes" or "No"). A statistical significance of $p \leq 0.05$ was applied to all inferential statistics.

Results

A total of 73 students consented and 67 students completed the survey leading to a response rate of 92%. Most of the participants were first-year students (n=57, 85%). The class was comprised of students from various engineering majors; Mechanical engineering (n=26, 38.8%), Civil/Environmental engineering (n=17, 25.4%), Electrical/computer engineering (n=11, 16.4%), Biomedical engineering (n=7, 10.4%), and Industrial engineering ((n=1, 1.3%). Self-reports on prior exposure to computer programming revealed that over half of the students did not have any programming experience (n=36, 53.7%). Students reported an average score of 7.8 (SD=1.9) on the rating for the overall usefulness of the various teaching methods. The highest level of usefulness was reported for lecture sessions (mean = 7.8, SD=1.7) and Teaching Assistant (TA) tutoring sessions (mean = 7.1, SD=2.4). Table 1 reports the findings on the participants' characteristics and perceived usefulness of the teaching methods.

Table 1: Participants Characteristics and Assessment of Teaching Methods (N=67)*

Student Characteristics	**N (%)**
Year of Study	
- Freshman	57 (85.1%)
- Sophomore	9 (13.4%)
- Junior	1 (1.5%)
Engineering major	
- Civil and Environmental	17 (25.4%)
- Electrical and Computer	11 (16.4%)
- Mechanical	26 (38.8%)
- Industrial	1 (1.3%)
- Biomedical	7 (10.4%)
- Undecided	5 (7.5%)
Participated in a Programming class/training in the past?	
- Yes	31 (46.3%)
- No	36 (53.7%)
Exposure to a programming language (multi-select question)	
- Python	6 (9.0%)
- Java	11 (16.4%)
- HTML	8 (11.9%)
- C^{++}	9 (13.4%)
- Visual Basic	1 (1.5%)
- Arduino	47 (70.1%)
- MATLAB	7 (10.4%)
- None	11 (16.4%)
- Other	2 (3.0%)
Assessment of Teaching Methods on Programming Skills	**Mean (SD)[#]**
On a scale of 1 to 10 (with 1 being not helpful and 10 being very helpful), how helpful were the various in-class activities (in-class	7.8 (1.9)

practice problems, think-pair share, vertical non-permanent surfaces, TA sessions) in helping you to learn programming during the course. For each of the following, indicate how you would rate the effectiveness of each of the following components of the course in helping you learn computer programming. Use a scale of 1 to 10 (with 1 being least effective and 10 being most effective) - Lecture sessions. - Vertical non-permanent surfaces. - Video support materials - TA tutoring sessions. - Write-pair-share activities.	 7.8 (1.7) 6.9 (1.9) 6.1 (2.4) 7.1 (2.4) 6.5 (2.6)

- *Missing data observed in some variables. # Post-intervention assessment (N=58)

Findings on Research Question #1: Do collaborative learning strategies like WPS and VNPS impact students' confidence and reduce anxiety with learning introductory programming?

At post-intervention, a statistically significant improvement was observed in the Confidence score for programming skills ($p<0.001$) and Confidence score in solving open-ended programming problems ($p<0.001$). The ATMI score did not improve significantly from baseline to post-intervention ($p=0.39$). Table 2 shows the findings in the ATMI summary score and Confidence scores.

Table 2: Changes in ATMI Score and Confidence

Variables	**Baseline, N=67 Mean (SD)**	**Post, N=58 Mean (SD)**	**p-value**
ATMI score	3.3 (0.6)	3.44 (0.6)	0.39
Confidence with programming skills	3.8 (2.1)	6.4 (1.6)	<0.001*
Confidence in solving open-ended programming problems	3.6 (2.3)	6.2 (1.9.2)	<0.001*

*Statistical significance = $p < 0.05$, T-test statistics

Findings on Research Question #2: For students without prior exposure to computer programming, how effective was the intervention in reducing students' anxiety and improving student confidence with introductory programming?

In the subgroup analysis of students with prior exposure to computer programming ("Yes" or "No"), students with prior exposure to computer skills were significantly less anxious at the beginning of the semester - ATMI score ($p=0.04$) and more confident in their programming skills ($p=0.01$). However, the participants' perception of their abilities to solve open-ended programming problems did not differ significantly at baseline ($p=0.58$). At post-intervention, the scores for anxiety (ATMI overall score) and confidence with programming skills for both cohorts of students ("Yes" or "No") did not differ significantly. Table 3 reflects the scores for the subgroup analysis.

Table 3: Score Distribution for students with Prior exposure to Computer programming

	Baseline Assessment (n=67)			Post-intervention Assessment (n=58)		
Prior Exposure to Computer Programming	Yes (31, 46.3%) Mean (SD)	No (36, 53.7%) Mean (SD)	p-value	Yes (28, 48.3%) Mean (SD)	No (30, 51.7%) Mean (SD)	p-value
ATMI score	3.5 (0.5)	3.2(0.5)	0.04*	3.5 (0.5)	3.3 (.6)	0.39
Confidence with programming skills	4.5(2.2)	3.2 (1.8)	0.01*	4.0(2.3)	3.4(2.2)	0.26
Confidence in solving open-ended programming problems	6.5(1.5)	6.3(1.6)	0.58	6.2(1.8)	6.1(2.1)	0.93

* Statistical significance = $p < 0.05$, T-test statistics

Discussions & Conclusions

Participants and Assessment of Intervention Methods

The study assessed the effectiveness of a structured teaching intervention to improve student confidence and anxiety with computer programming. The teaching methods integrated into the intervention were Write-Pair-Share (WPS) activities, Vertical Non-Permanent Surfaces (VNPS), and video support materials. Lecture sessions and TA sessions were designed to improve problem-solving skills and improve confidence in programming skills.

Overall, the summary rating of the various teaching methods was high (mean= 7.8, SD=1.9). The level of usefulness was reported for lecture sessions (mean = 7.8, SD=1.7), Teaching Assistant (TA) tutoring sessions (mean = 7.1, SD=2.4), VNPS (mean = 6.9, SD=1.9), WPS ((mean = 6.5, SD=2.6), and Video Support (mean = 6.1, SD=2.4). In this study, the VNPS activities were applied after the students were comfortable with the basic programming concepts. Students worked in groups to solve the VNPS problem sets. The VNPS activities were complex and relevant to real-world scenarios. Improvement in programming skills and confidence were the primary focus during the VNPS activities as studies suggests that VNPS activities promote dynamic classroom practices [14].

In the overall student population, the teaching intervention was effective at improving the student's confidence in their programming skills ($p<0.001$) and solving open-ended programming problems ($p<0.001$). However, the ATMI score did not improve significantly post-intervention. These findings support the utility of collaborative learning methods on students' confidence in computer programming. The VNPS and WPS activities were included in the lecture sessions for students to apply the lessons to real-world problems. Findings suggest this approach served to improve problem-solving skills for open-ended programming problems and improved confidence in programming skills.

In the subgroup analysis of students with or without computer programming experience (“Yes” or “No”), students with prior exposure to computer programming were significantly less anxious at the beginning of the semester - ATMI score ($p=0.04$) and more confident in their programming

skills (p=0.01). However, after the intervention, both the ATMI score and levels of confidence were not significantly different between the two groups. This suggests that both groups of students did not differ in their levels of confidence or anxiety with computer programming after the intervention. This finding is significant in supporting decision-making to adopt collaborative learning strategies in computational courses to support students with less preparedness for these courses. It also provides students with greater preparedness for such courses the opportunity to utilize their knowledge to benefit the class in general, thereby enhancing their engagement in the learning process. In addition, it highlights the benefit of integrating computer programming into the curriculum early in students' educational program, as it helps students to confidently take on computationally challenging course as they progress in the educational journey [15].

Limitations

Study limitations that need to be considered include external generalizability of the study findings, social desirability bias, and a potential source of bias that may be related to the variability in participants' responses since two faculty taught the course. The participants were first-year students in the Northeastern region of the US. Student characteristics and exposure to MATLAB programming may be different in other regions which may impact the external generalizability of the study findings. Secondly, social desirability bias is a potential limitation for self-report studies – a tendency for participants to over-report desirable qualities. This limitation was addressed in the consent form where participants were informed that their responses would be anonymous to the faculty members. It is also possible that students may have been involved with learning outside of the classroom, for example, being involved in an engineering club where programming activities are carried out. Finally, two faculty delivered the intervention and the potential for slight variations in implementing the collaborative strategies may occur. However, both faculty members followed the study protocol and applied similar case scenarios to ensure intervention integrity.

Conclusions

Study findings suggests that adopting collaborative learning strategies in introductory programming courses could help students overcome anxiety issues associated with learning complex and challenging concepts. Collaborative learning strategies are especially helpful for students who may not have had prior exposure to programming before starting their college programs. In addition, early exposure to programming is helpful for students as they transition into college. As such, programs that expose students to programming at the high school level are vital for engineering education. Also, it is critical to design introductory programming courses in a way that recognizes the fact that a significant number of first-year students lack prior exposure to programming and the need to address such deficiencies. Future studies may investigate the effectiveness of teaching interventions which focuses on other domains of the ATMI survey such as enjoyment of introductory programming, motivation, and value of programming.

References

1. Carberry, A. R., & McKenna, A. F. (2014). Exploring student conceptions of modeling and modeling uses in engineering design. *Journal of Engineering Education*, *103*(1), 77-91.

2. Wiedenbeck, S., Labelle, D., & Kain, V. N. (2004, April). Factors affecting course outcomes in introductory programming. In *PPIG* (p. 11).

3. Guzdial, M., & Soloway, E. (2002). Teaching the Nintendo generation to program. *Communications of the ACM*, *45*(4), 17-21.

4. Margolis, J., & Fisher, A. (2002). *Unlocking the clubhouse: Women in computing*. MIT press.

5. Zawojewski, J. S., Diefes-Dux, H. A., & Bowman, K. J. (2008). *Models and modeling in engineering education: Designing experiences for all students*. Brill.

6. Laal, M., & Ghodsi, S. M. (2012). Benefits of collaborative learning. *Procedia-social and behavioral sciences*, *31*, 486-490.

7. Ekong, J., Chauhan, V., Osedeme, J., & Niknam, S. (2022, August). A framework for Industry 4.0 workforce training through project-based and experiential learning approaches. In *2022 ASEE Annual Conference & Exposition*.

8. Al Mezrakchi, R., & Al-Ramthan, A. (2022, March). The Impact of collaborative learning strategies on Engineering Students' Ability to Problem Solve and Apply Theories to Practical Applications. In *2022 ASEE Gulf Southwest Annual Conference*.

9. Murray, L., Ekong, J., Niknam, S., & Rust, M. (2022, August). A Framework for Implementing Design for Additive Manufacturing Methods in First-Year Engineering Curriculum: Investigating the effects of specialized training on engineering design and student self-efficacy. In *2022 ASEE Annual Conference & Exposition*.

10. Tinto, V. (2012). *Leaving college: Rethinking the causes and cures of student attrition*. University of Chicago press.

11. Sauve, R., Evans, C., & Schneider-Bentley, L. (2023, June). Work in Progress: PEER LED COLLABORATIVE COURSES DEVELOP A SENSE OF BELONGING AND COMMUNITY FOR ALL UNDERGRADUATE ENGINEERING STUDENTS. In *2023 ASEE Annual Conference & Exposition*.

12. Liljedahl, P. (2016). Building thinking classrooms: Conditions for problem-solving. *Posing and solving mathematical problems: Advances and new perspectives*, 361-386.

13. Lim, S. Y., & Chapman, E. (2013). Development of a short form of the attitudes toward mathematics inventory. *Educational studies in mathematics*, *82*, 145-164.

14. Mikes, M. (2021). Teacher Perceptions of the Impact of Vertical Non-Permanent Surfaces in Mathematics Classrooms. *Lincoln Memorial University*.

15. McCoy, L. P., & Burton, J. K. (1988). The relationship of computer programming and mathematics in secondary students. *Computers in the Schools*, *4*(3-4), 159-166.

Appendix A: Attitudes Toward Mathematics Inventory (ATMI) - Modified version

Instructions: The following questions consist of statements about your attitude toward programming. There are no correct or incorrect responses. Read each item carefully. Q1: Think about how you feel about each question and select the option that most closely corresponds to how the statement best describes your feelings.

	Strongly Disagree (1)	Disagree (2)	Neither Agree nor Disagree (3)	Agree (4)	Strongly Agree (5)
I have usually enjoyed studying computer programming or a programming class in school (1)	○	○	○	○	○
I like to solve new problems in computer programming or programming in general. (2)	○	○	○	○	○
I really like computer programming or programming in general. (3)	○	○	○	○	○
I am happier in a computer programming or programming class than in any other class. (4)	○	○	○	○	○
Computer programming is a very interesting subject. (5)	○	○	○	○	○

Q2: Please, think about how you feel about each question and select the option that most closely corresponds to how the statement best describes your feelings.

	Strongly Disagree (1)	Disagree (2)	Neither Agree nor Disagree (3)	Agree (4)	Strongly Agree (5)
I am confident that I could learn advanced computer programming. (1)	○	○	○	○	○
I am willing to take more than the required amount of computer programming. (2)	○	○	○	○	○
I plan to take as much computer programming as I can during my education. (3)	○	○	○	○	○
The challenge of computer programming appeals to me. (4)	○	○	○	○	○

Q3: Select the option that most closely corresponds to how the statement best describes your feelings.

	Strongly Disagree (1)	Disagree (2)	Neither Agree nor Disagree (3)	Agree (4)	Strongly Agree (5)
Studying computer programming makes me feel nervous (1)	○	○	○	○	○
I am always under a terrible strain in computer programming class. (2)	○	○	○	○	○
It makes me nervous to even think about having to do a computer programming problem. (3)	○	○	○	○	○
I am always confused in my computer programming class. (4)	○	○	○	○	○
I feel a sense of insecurity when attempting computer programming. (5)	○	○	○	○	○

Q4: Identify the option that most closely corresponds to your opinion about computer programming.

	Strongly Disagree (1)	Disagree (2)	Neither Agree nor Disagree (3)	Agree (4)	Strongly Agree (5)
Computer programming is a very worthwhile and necessary subject. (1)	○	○	○	○	○
Computer programming is important in everyday life. (2)	○	○	○	○	○
Computer programming is one of the most important subjects for people to study. (3)	○	○	○	○	○
College computer programming lessons would be very helpful no matter what I decide to study in future. (4)	○	○	○	○	○
A strong computer programming background could help me in my professional life. (5)	○	○	○	○	○

End of Block: Section 2