Computer Science Students CAN write: A year-long writing project

by Melody Lam, 2019 CTI Fellow
William Amos Hough High School

This curriculum unit is recommended for Computer Science courses, 9-12

Keywords: computer science, mathematics, writing

Teaching Standards: See Appendix 1 for teaching standards addressed in this unit.

Synopsis: It is a common myth that computer programmers lack communication skills. This curriculum unit is designed to "bust the myth" by allowing students to develop crucial communication skills through a student-led project. Students will propose a project that piques their interest, write a project proposal, and seek feedback from their peers and other members of their community. Finally, as a culminating assignment, students will present their progress and work on their project though a short presentation and short, 750 word paper.

I plan to teach this unit during the coming year to grade 9-12

Computer Science Students CAN write: A year-long writing project

by Melody Lam

Introduction

Rationale

The purpose of the unit is to allow students who take a computer science course the opportunity to create a project that reflects their own interests as well as for them to practice an often overlooked soft skill: communication. With the growing interest in computer science in both education and in today's industry, more students are taking computer science courses. However, there is a marked lack of various communication opportunities, especially in the written form, in these computer science courses. Oftentimes there is a reluctance to introduce writing prompts or opportunities for writing because it is simply not needed, or it would detract from the time for acquiring concepts that are relevant to the subject at hand.

CTE courses are designed "to help equip students with the 21st century skills needed for a global economy"[1]. Looking at the NC CTE objectives for computer programming courses and the AP Course and Exam Descriptions (CED), besides one particular class (AP Computer Science Principles), there is at most minimal focus on communication skills. In fact, the NC standards for the computer programming courses do not mention communication at all.[2] In AP Computer Science Principles, one of the "Big Ideas" that drive the course is "Communication", and there are numerous opportunities for students to communicate, from pair programming paradigms to creating a computer program and discussing the program through a short 750 word technical paper to be submitted to CollegeBoard as part of the exam. In Python Programming 1, replacing Computer Programming 1 from the 2019-2020 year, communication is relegated to the objective of commenting the user code so that other programmers can understand the purpose and aims of the code. In AP Computer Science A, a "Computational Thinking Practice" is "Documentation" but is relegated to describing the function of computer code and why it would not work, using technical vocabulary.[3] In any case, there is not a standard objective of being able to communicate effectively across the three computer science courses. As AP Computer Science Principles is mainly a freshman-level class, students will have practiced the soft-skill of communication but do not have the opportunity to reinforce the skill throughout the 3 courses.

[1] "Charlotte Mecklenburg Schools Career & Technical Education."
[2] "NORTH CAROLINA CAREER AND TECHNICAL EDUCATION STANDARDS - Business, Finance, and Information Technology Education - Computer Programming 1."
[3] "AP® Computer Science A."

Demographics

William Amos Hough High School is a large suburban high school of over 2500 students located in the small town of Cornelius, North Carolina just north of Charlotte. We opened our doors in 2010 to serve the northern part of the Charlotte-Mecklenburg School District. Eighty-four percent of our graduates go on to either two- or four-year colleges while 16% join the military. Twenty-six percent of our students are minorities and 18% are free or reduced lunch students. We offer a comprehensive college preparatory program in the arts and sciences. Classes are taught at the Standard and Honors levels and we offer 26 Advanced Placement courses in conjunction with the College Board. Hough also offers a variety of CTE classes, from engineering to marketing to computer science. Students that are interested in computer science take classes in the Software Development pathway. We offer 3 choices this year: AP Computer Science Principles, AP Computer Science A, and Python Programming 1. (Hough did not offer Microsoft Software Development Fundamentals due to a lack of enrollment.)

I am currently teaching all 3 computer science courses this year. Hough started offering AP Computer Science Principles in the 2017-2018 year and AP Computer Science A in the 2018-2019 year. The previous course in the pathway, Computer Programming 1, had already been offered and has been replaced with Python Programming 1 starting this year. AP Computer Science Principles classes average 30 students, Python Programming 1 classes average 15 students, and AP Computer Science A average 12 students during the 2019-2020 school year. The majority of students that take these courses are male. There is a sizable amount of students for which the AP Computer Science Principles class is their first AP, and about half of the students in AP Computer Science Principles are freshmen. AP Computer Science A is only offered at the sophomore-senior level. While there is a defined sequence in the Software Development pathway, students can take courses off-sequence.

In Principles, the language of choice is Snap!, a teaching language that utilizes blocks to arrange a program. Along with basic programming and computational thinking skills, the course covers the workings of the Internet as well as global impacts of computing. Python Programming 1 and AP Computer Science A are more traditional CS courses, where the focus is on developing proficiency in a programming language. In Python Programming, the language is Python, a simple language predominantly oriented towards data science applications. In AP Computer Science A, the language is Java, an object oriented language that is widely used in CS 101 courses around the nation as well as in the industry in embedded applications such as ATM machines and Point-Of-Sale machines.

Unit Goals

The general goals of the unit are as follows:

- Develop the soft skill of communication through the written and verbal forms.
- Develop student self-reliance and sufficiency through a student-directed project
- Create an artifact (computer program and written description of the program) that can be used in a portfolio for resume building and/or a graduation project (if the school has a graduation project).

The unit aims to satisfy the AP Computer Science Principles Big Idea of communication by allowing students to develop a year-long project of their own: Students communicate through a final presentation, writing formal reports, and collaborating with others on their project. As there are no explicit standards in either AP Computer Science A or Python Programming concerning communication, I will refer to the *Computer Science Curricula 2013* learning outcomes in the Professional Communication area.[4] While the document is oriented towards undergraduate programs in computer science, the learning objectives are appropriate for students in K-12 with support in place from the teacher.

This unit is designed for high schools; however, with modifications, this project can be done at the middle school and elementary school levels.

[4] ACM Computing Curricula Task Force, *Computer Science Curricula 2013*.

Content Research

A pervasive myth in Computer Science education, especially when preparing students for the workforce, is that Computer Science is simply just programming computers to perform a task. The focus is getting students proficient in programming tools and techniques. However, when looking at what the tech industry expects out of new hires (besides programming skills), they frequently list communication as a skill. [5] In fact, most recent graduates from a Computer Science or Software Engineering program are notoriously weak in this area.[6] However, it seems like progress is being made to improve a graduate's communication-related soft skills. For example, universities such as Eastern Washington University are introducing courses in technical writing for students in the computer science major.[7]

In the postsecondary area, most faculty do not focus on communication (or other soft skills) because of the lack of time or a fear that teaching communication skills may reduce time spent on essential technical content.[8] Not only that, by adding a communication/writing component to a course, instructors may find that it adds more work to grading and assessment of students.[9] This is in contrast to the ISTE/ACM standards for computing courses at the undergraduate level, which emphasise at least 1 hour (if not more) of "social issues" and "professionalism" content in an undergraduate course. [10,11] While there are articles from professors and CS faculty at the postsecondary level detailing their efforts to incorporate technical writing into a CS major track[12], there is little research or study done with writing in the CS curriculum at the secondary (high school) level. Most journal articles are simply detailing the implementation of writing into a CS course with some small commentary on the effects on the students, but do not have a long-term followup.[13] For example, at the University of Scranton, Professors Jackowitz, Plishka, and Sidbury introduced a writing component in the form of reading an article and testing over the article, as well as requiring the students to write a specification sheet on their assignments.Later on, the students have a senior project to turn in with an extensive writing component. At the conclusion of the study, the researchers determined

[5] "10 Skills You Didn't Know Could Land You an IT Job."
[6] Anderson et al., "CS/SE Instructors Can Improve Student Writing without Reducing Class Time Devoted to Technical Content."
[7] Kaufman, "Technical Writing and Computer Programming."
[8] Anderson et al., "CS/SE Instructors Can Improve Student Writing without Reducing Class Time Devoted to Technical Content."
[9] Anewalt, "EXPERIENCES TEACHING WRITING IN A COMPUTER SCIENCE COURSE FOR THE FIRST TIME."
[10] Tucker, "Computing Curricula 1991."
[11] ACM Computing Curricula Task Force, *Computer Science Curricula 2013*.
[12] Jackowitz, Plishka, and Sidbury, "Teaching Writing and Research Skills in the Computer Science Curriculum."
[13] Jackowitz, Plishka, and Sidbury.

that introducing writing assignments early has a long-term benefit to students. However; it causes additional strain by students and by the instructors of the course.[14]

At other colleges, such as Seton Hill University, writing is incorporated into a CS course in the form of a writing intensive course, where students design and write a programming language of their own.[15] Tircuit, in his paper, details the process of adapting a writing intensive program for an existing course. Unlike in the Scranton example, his course focuses less on the research aspect (simply asking for sufficient research to make the argument) and more on writing for industry professionals. In Tircuit's case, most students go on to the industry rather than pursuing a graduate level degree. The process of writing the paper is very much aligned with an English class's writing process, where students prewrite, draft, and then submit the paper. One of the things that Tircuit details is objectives of the assignment, which are to "learn to write well enough to write a proposal at your future place of employment" and "[express] your ideas clearly and with strong rationale for your language choices".[16] While Tircuit was pleasantly surprised by some of the students going above and beyond. He also shared some reservations with peer review, citing "bad experiences". As a result he focuses on peer collaboration such as discussing anonymized responses with the whole group. I believe that for a high school student, peer collaboration is more viable as they can see many examples of good and bad papers.

On a national level, the Computer Science Teacher Association has listed various objectives to promote writing in computer science classes at all levels in their K-12 Computer Science Standards.[17] For example, in the K-2 level, we see that students are expected to develop a plan for a program through various mediums such as storyboards or graphic organizers. At the HS level, students are expected to "document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs." However, it seems like while there are these standards, courses at the state level are slow to adopt them. Looking at the NC CTE objectives for computer programming courses and the AP Course and Exam Descriptions (CED), besides one particular class (AP Computer Science Principles), there is at most minimal focus on communication skills. In fact, the NC standards for the computer programming courses do not mention communication at all.[18] Considering that communication is an essential soft skill that industry expects from their programmers, it seems remiss that it is not included in the standards of the curriculum. However, considering the approximately 90 days (block schedule) allotted for teaching courses in North Carolina, it seems that the focus is on acquiring technical and content skills versus developing soft skills in communication and writing. However, Linda Pesante at Carnegie Mellon warns that "isolated attempts to teach

[14] Jackowitz, Plishka, and Sidbury.
[15] Tircuit, "Teaching Writing in Computer Science."
[16] Tircuit.
[17] "CSTA Computer Science Standards Revised 2017.Pdf."
[18] "NORTH CAROLINA CAREER AND TECHNICAL EDUCATION STANDARDS - Business, Finance, and Information Technology Education - Computer Programming 1."

writing hinder the transfer of learning to other courses..and to the workplace."[19] Therefore, the unit aims to be a general framework for writing in computer science so that it can be applied across all computer science courses and that teachers may choose to utilize this across the years amongst common students.

In the Writing in the Mathematics seminar, we started by describing what mathematical writing really is. The focus is to have students write about mathematics for the purposes of reasoning and communication. For this unit, I decided to focus on Exploratory, Informative, and Mathematically Creative writing for the students. The Exploratory portion would be the specification sheet or "spec sheet", where the students describe the project they would like to embark on. The Informative portion would come during the formal reporting, and the mathematically creative portions would come through the project itself. [20] While the seminar is oriented towards elementary school writing, because computer science is taught at a relatively basic level in the K-12 education sphere, a lot of the elementary school writing techniques can be applied to computer science.

Pesante details some ways to approach the writing situation in a computer science class. In essence, according to Pesante, writing and software development (programming) are rather the same. [21]A lot of the postsecondary courses that do utilize writing also argue that writing and programming are much the same as well. Therefore, computer science teachers can relate the writing process by explaining to students that the process is much like the programming process. [22] In addition, students need to identify the reader (the teacher, acting as if they did not know what the student was doing) and the writer (themselves) and see that the communication is effective if both parties are on the same page. [23] Therefore, in my curriculum unit, I identify the teacher as a "boss" or a "project manager" in a software development company, who may not know (or want) the exact technical details of the project and relies on the software developer (the student) for expository information on the project.

Dugan also gives some advice in his paper about writing in computer science courses. Dugan's courses were co-taught with a writing professor. In his paper, he splits the writing into 3 general categories: learning, academic communication, and industrial communication. In the case of this curriculum unit, we are focusing on industrial communication, in what Dugan calls "team" and "project management" categories.[24] I focus on industrial communication because students can utilize these communication skills in not only the coursework but also in the

[19] Pesante, “Integrating Writing into Computer Science Courses.”
[20] Colonnese, “Instructional Guidelines for Elementary Mathematical Writing.”
[21] Pesante, “Integrating Writing into Computer Science Courses.”
[22] Pesante.
[23] Pesante.
[24] Dugan and Polanski, “WRITING FOR COMPUTER SCIENCE: A TAXONOMY OF WRITING TASKS AND GENERAL ADVICE.”

workforce. While there may be some research and peer evaluation segments in this curriculum, the main writing portions would fit the former categories. He also gives some advice on teaching structure and grammar "as needed"; K-12 teachers may feel uncomfortable with this. However, I feel that some structure can be taught to the students; for example, giving students sentence stems to guide how they should write certain sections, or possibly referring them to an English teacher for more assistance. While co-teaching is definitely an option, because of the varying skill levels and ages of the students in an elective class this may be almost (or completely) impossible to accomplish.

In addition, the curriculum unit utilizes a similar format to the CTI seminars. In our seminars, we discuss the subject of the seminar (for example, Writing in Mathematics) in 2 hour sessions with a seminar leader (a professor at UNCC or at Johnson C. Smith) and seminar members. The culminating project is a curriculum unit that unites the ideas presented in the seminar to the coursework at hand. During the writing of the curriculum project, seminar members write 2 drafts and get feedback from the seminar leader on the curriculum. Students in the curriculum unit will get feedback at regular intervals concerning the writing and progress of their project. Without guidance and feedback, students may be unfocused in their writing and fail to communicate effectively. Also, in *Computer Science Students CAN Write* (CSSCW), there is a focus on efficient communication versus correct grammar and sentence structure. While there is an expectation that students use the tools at hand (such as Google Docs, Grammerly, etc) to be as correct as possible with their spelling and grammar, we value seeing the students think about the process of the project and their critical thinking over English proficiency skills. Anewalt in her paper suggests something similar to the conferences between seminar leader and teachers where instead of grading a draft, there is a "cold conference" where students talk to the instructor in appointments where there is a discussion of the paper in a limited time frame.[25] However, unlike in a university/college setting there are not on-campus resources for writing help besides the English teachers. I feel that a good working relationship with the English teachers on campus will help to address the necessary writing support and mitigate students' differing English skills somewhat.

[25] Anewalt, "EXPERIENCES TEACHING WRITING IN A COMPUTER SCIENCE COURSE FOR THE FIRST TIME."

General Teaching Methods

To implement this curriculum, an important aspect is an environment that fosters collaboration amongst students and the teacher. Along with this, there needs to be time set aside for students to work on their projects and written portions.

This project has several components, some of which will be looped/repeated. The components are as follows:

- **Component 1:** Student project/proposals
- **Component 2:** Progress Checks
- **Component 3:** Developing the product/project
- **Component 4:** Presenting the finished product/project.
- **Component 5:** Reflection Piece

Component 1: Student project and proposals

In Component 1, students develop a potential project to pursue. This should be done after the first few weeks of the course. This gives time for the students to develop basic skills in programming, get acquainted with the course material and how it will be taught in the class, and get acquainted with the teacher and the students in the course. For the teacher, this allows them to get acquainted with the students and previous knowledge/skill sets that students may come into the course with.

The students should attempt to independently develop an idea, but if students are having difficulty developing an idea to work on, have a class parking lot or idea board where students can list ideas that they would like to pursue. The ideas need not be unique; in fact, students will probably propose an idea that has already been done. Teachers should make sure the ideas are feasible within the context of a year. Often times, students who have heard of things like "machine learning" or "quantum computing" will suggest these ideas; however, it will probably not end well for the student at the end of the year. Students should also try to be somewhat focused in their idea -- for example, "games" could be refined to a particular game like "pong", "tic-tac-toe" or "checkers", amongst other games.

After students develop an idea, they will write a proposal for their idea. In their initial proposal, students should detail as much as possible what their idea is and how they would accomplish it, including a sample timeline. The students should know that this is a non-binding proposal and that proposals often change direction and timing, just like in real life. A sample worksheet is in Appendix 2. For the teacher, this can be an assessment of a student's initial writing skills. The teacher should also anticipate possibly modeling the proposal for the students, so they have an exemplar proposal. Suggested sentence stems are in Appendix 3.

Finally, it may be that a student picks a project for which you do not have experience guiding a student in. Students should learn some self-sufficiency and reliance. Teachers may need to teach students about how to find reputable sources for information that they may need, and express a desire to learn more -- it is a good opportunity to have students collaborate and learn from each other!

Component 2: Progress Checks & Component 3: Developing the product/project

Students should work on the project throughout the year. The progress checks can be done at any interval that you wish. Here are some general tips about progress checks and developing the project.

Progress checks can be done as a small, written mini report. In these mini-reports, students report on their progress on the project and where they will go next. (See Appendix 1 for a sample progress report in this style). Another way to do a progress check is to do a group chat/discussion, much like how the Scrum development style is used in the industry. [26] In the Scrum software development style, the software development is split into *sprints* of a certain duration, where developers work on the project, and *scrums*, where the team comes back together to discuss the progress of the project and where to go next. While students will be working on this project on their own, feedback from other students will be valuable. Students will also practice verbal communication and questioning through the group discussions.

One-on-one progress checks between teacher and student are also valuable ways for students to gain experience in communication. A suggested way to conference with the student is to use their project proposals and progress checks to gain a better understanding of the student's project. Because it is very likely that the teacher does not know about the project or how the project should work, it helps the student gain more confidence in their project as well as learn how to communicate effectively.

Additionally, there may be some modeling required of the teacher, especially during the proposal and first few formal reporting sessions. However, by setting a hard word limit on the written portions as well as giving students guiding questions, I anticipate a lot of stress for the students will be mitigated. By also giving students guiding questions, you can give them a clear direction on how to approach the project as well as create a sense of accomplishment for students that are not confident in their writing. The one on one sessions can be done during a class period or two -- have some guiding questions during the meeting as well to help focus both parties on the task at hand.

[26] "Scrum (Software Development)."

Component 4: Presenting the finished product/project.

Along with developing the project, the student should be working on their presentation. The presentation can be much like the AP Computer Science Principles Create Task, where the students create a short 1 minute video of their program running as well as write a short, 750 word paper on their project, detailing the purpose of their program and highlighting an algorithm and abstraction present in the project. I recommend the students instead create a "computational artifact" for their program and a short 750 word paper detailing their development process written in the technical writing style.

The "computational artifact" can be something like a collage of screenshots of their program running, or perhaps a user manual in that style; however, encourage creativity by discussing what a "computational artifact" entails (something created by the computer) and have students discuss what and what is not a computational artifact. Examples of more creative artifacts include an advertisement or commercial for their project, or a public service announcement for projects that involve assisting a group of people or providing a service for the community.

The short, 750 word paper is to make sure that students stay on the task at hand (describing the project and the development process) and to make sure that students stay succinct and descriptive about their project. In this paper, students should use the progress checks as a guide for writing the portion about their development process. Encourage their writing by giving them guiding questions for their paper, and by giving them sentence stems to help them write the more difficult parts. See Appendix 3 for example guiding questions.

As part of the focus on communication, students should present the finished product/project. In AP Computer Science Principles, the culminating activity can be the submission of the Create task and a presentation of their project after the due date for the Create task. In other CS courses, a presentation as part of the grading process would be valuable for students to practice their communication skills. The CTE Advanced Studies course has a very good template & rubric that you can adapt to the projects at hand.

Component 5: Reflection Piece

A reflection piece allows students to look back at how they improved in their communication skills as well as allow them to look forward at what they can do to improve their project. It will also allow the teacher to see what the students have learned and how much they have grown with the project.

In the reflection piece, students should reflect back on a "glow" where they felt like they accomplished the most, as well as a "grow" where students detail something that they could've been better at, or maybe look forward to if they had more time. The reflection piece should be no more than 2-3 paragraphs.

Appendix 1: Teaching Standards

These teaching standards come from the CSTA K-12 Computer Science Standards, Revised 2017. The standards are located here: https://www.csteachers.org/page/standards

2-AP-15 Seek and incorporate feedback from team members and users to refine a solution that meets user needs.

This is covered in Component 3: Developing the Product/Project in the one-on-one meetings and "scrum" style discussions. While students may not be working in formal teams, they should seek out feedback from other students.

3A-AP-16 Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue by using events to initiate instructions

While event driven programming is not a huge focus of this CU, students should be designing a project that fits their needs and choice. Not all projects need to address societal issues.

3A-AP-23 Document design decisions using text, graphics, presentations, and/or demonstrations in the development of complex programs.

Students document their progress through their project using the progress checks and by keeping track of all the changes that happen with their program. They can also "document" by demonstrating their program and presenting their program at the end of the CU.

3B-AP-17 Plan and develop programs for broad audiences using a software life cycle process.

This goes hand in hand with 3A-AP-16 in that students will be developing their project through a cycle of testing, feedback, and revising their project as needs and users dictate.

Appendix 2: Worksheets/Handouts

Worksheet 1: Project Planning Sheet

The 20% Time Project Planning Sheet

Project Goal: What will the project accomplish?	
Project Product: I want an actual product, not just a bunch of ideas. What product will lead towards the goal?	
Resources/Supplies: What will you need in order to have the product done? Think about not only the concrete but also the abstract, like knowledge.	
Timeline: What's your timeline for getting all of this done? Since we're working backwards here, let's start with what you should have **May 2019** and go on from there...	
May 2019	
March-April 2019	

January-February 2019	
November-December 2018	
September-October 2018	
Stretch goals. If you reach your first goal, what are some other, secondary goals you would like to accomplish too? (sounds like a Kickstarter here!)	
And the hardest question: why do you want to do this project? What kind of questions does your product and progress aim to answer?	

Weekly Performance and Progress Chart

Below is a mini calendar. List what you hope to accomplish on each day. Consider things like arranging for an interview, purchasing materials, ordering special materials needed, working on project for one hour, or arranging for teacher conference.

Name ______________________________ Week Beginning /Week Ending ________________

	DESCRIPTION OF PROGRESS TOWARD COMPLETION OF PROJECT
MONDAY	
TUESDAY	
WEDNESDAY	
THURSDAY	
FRIDAY	
SATURDAY/SUNDAY	

Reflection of Weekly Performance and Progress by the student:

1. List and explain problems you are experiencing in completing your tasks.

2. Do you need a conference with your teacher? ❑ Yes ❑ No

Date of Conference: _________________________

[27] "2006_Advanced_Studies_Implementation_Guide.Pdf."

Worksheet 3: Portfolio Rubric

STUDENT DOCUMENTS, continued
Portfolio – Rubric

Teacher Name: ________________________________

Student Name: ______________________________

CATEGORY	4	3	2	1	Total
Organization	Content is well organized using headings or bulleted lists to group related material.	Uses headings or bulleted lists to organize, but the overall organization of topics appears flawed.	Content is logically organized for the most part.	There was no clear or logical organizational structure, just lots of facts.	
Requirements	All requirements are met and exceeded.	All requirements are met.	One requirement was not completely met.	More than one requirement was not completely met.	
Attractiveness	Makes excellent use of font, color, graphics, effects, etc. to enhance the presentation.	Makes good use of font, color, graphics, effects, etc. to enhance to presentation.	Makes use of font, color, graphics, effects, etc. but occasionally these detract from the presentation content.	Use of font, color, graphics, effects etc. but these often distract from the presentation content.	
Originality	Project shows a large amount of original thought. Ideas are creative and inventive.	Project shows some original thought. Work shows new ideas and insights.	Uses other people's ideas (giving them credit), but there is little evidence of original thinking.	Uses other people's ideas, but does not give them credit.	
Mechanics	No misspellings or grammatical errors.	Three or fewer misspellings and/or mechanical errors.	Four misspellings and/or grammatical errors.	More than 4 errors in spelling or grammar.	
Total Points with Conversion Chart to Grade	**34 TO 44 = B+/A+**	**23 TO 33 = C+/B-**	**12 TO 22 = D/C+**	**11 or less = F**	

Appendix 3: Resources for Teachers

The following are some guiding resources that I have used in the classroom with this project. Teachers can opt to use these resources as part of their curriculum unit to help guide students on their tasks.

Sentence Stems for Component 1: Student Project/Proposals

The goal of the project is to: ________________________ (The student should dictate the purpose of the program. It should answer the question of "what does this program accomplish?")

The product/project should ___________________ *(Insert purpose here)* by _____________________ (The student should fill this in by stating how the program is going to do it.)

In ________ *(insert timeframe here)*, I will __________________ *(insert task here*).

Examples:

The goal of the project is to <u>allow users to play a game of 20 questions</u>.
The project should allow the user to play the game by <u>asking the user yes-or-no questions about what object they are thinking of. Then the program will try to guess after the 20 questions the object. The computer wins if it successfully thinks of the object.</u>
In <u>September</u>, I will <u>learn how to use programming to determine the types of questions that the program should ask.</u>

These sentence stems should help the student think about their project in a focused way.

Guiding questions for Component 4: Presenting the project/product

- What was an instance where you had difficulty with the project. How did you resolve it?
- What was an instance where you had an opportunity to improve or extend your project?
- What were some programming structures that you utilized in the project? Would the project be feasible without these structures?
- Explain a feature of your program and how it is crucial to the operation of the program.

Bibliography

"10 Skills You Didn't Know Could Land You an IT Job." Accessed September 14, 2019. https://certification.comptia.org/career-change/exploring-it/skills-for-it.

"2006_Advanced_Studies_Implementation_Guide.Pdf." Accessed November 17, 2019. https://acrhs.buncombeschools.org/UserFiles/Servers/Server_92613/File/Staff/Lowman,%20Mark/2006_Advanced_Studies_Implementation_Guide.pdf.

ACM Computing Curricula Task Force, ed. *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. ACM, Inc, 2013. https://doi.org/10.1145/2534860.

Anderson, Paul V., Sarah Heckman, Mladen Vouk, David Wright, Michael Carter, Janet E. Burge, and Gerald C. Gannod. "CS/SE Instructors Can Improve Student Writing without Reducing Class Time Devoted to Technical Content: Experimental Results." In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, 455–64. Florence, Italy: IEEE, 2015. https://doi.org/10.1109/ICSE.2015.178.

Anewalt, Karen. "EXPERIENCES TEACHING WRITING IN A COMPUTER SCIENCE COURSE FOR THE FIRST TIME," 2002, 10.

"AP® Computer Science A," n.d., 218.

CMS Career & Technical Education. "Charlotte Mecklenburg Schools Career & Technical Education." Accessed September 21, 2019. https://discovercte.com/.

Colonnese, Madelyn W. "Instructional Guidelines for Elementary Mathematical Writing," 2017, 10.

"CSTA Computer Science Standards Revised 2017.Pdf." Accessed November 18, 2019. https://www.doe.k12.de.us/cms/lib/DE01922744/Centricity/Domain/176/CSTA%20Computer%20Science%20Standards%20Revised%202017.pdf.

Dugan, Robert F, and Virginia G Polanski. "WRITING FOR COMPUTER SCIENCE: A TAXONOMY OF WRITING TASKS AND GENERAL ADVICE." *Journal of Computing Sciences in Colleges* 21, no. 6 (2006): 13.

Jackowitz, Paul M., Richard M. Plishka, and James R. Sidbury. "Teaching Writing and Research Skills in the Computer Science Curriculum." *ACM SIGCSE Bulletin* 22, no. 1 (February 1, 1990): 212–15. https://doi.org/10.1145/319059.323454.

Kaufman, J. "Technical Writing and Computer Programming." *IEEE Transactions on Professional Communication* 31, no. 4 (December 1988): 171–74. https://doi.org/10.1109/47.9219.

"NORTH CAROLINA CAREER AND TECHNICAL EDUCATION STANDARDS - Business, Finance, and Information Technology Education - Computer Programming 1." Accessed September 14, 2019. http://center.ncsu.edu/standards/NCCTE/pdf/NCCTE.BP10.pdf.

Pesante, Linda H. "Integrating Writing into Computer Science Courses." *ACM SIGCSE Bulletin* 23, no. 1 (March 1, 1991): 205–9. https://doi.org/10.1145/107005.107040.

"Scrum (Software Development)." In *Wikipedia*, November 1, 2019. https://en.wikipedia.org/w/index.php?title=Scrum_(software_development)&oldid=923990081 Page Version ID: 923990081.

Tircuit, A. "Teaching Writing in Computer Science," n.d., 8.

Tucker, Allen B. "Computing Curricula 1991." *Communications of the ACM* 34, no. 6 (June 1, 1991): 68–84. https://doi.org/10.1145/103701.103710.