|  |  |
| --- | --- |
|  | Concept Assignment 1  PLTW Computer Science CSP Core Training |

# Problem Solving, Collaboration, and Creativity

|  |  |
| --- | --- |
|  | Learning Objectives |

LO1.1 While examining resources in the PLTW Computer Science Principles (CSP) course and on the AP CSP course website, the teacher will:

* Inspect the components of the AP CSP Framework.
* Recognize the alignment of PLTW CSP to that framework.
* Join the AP CS Principles Teacher Community.

LO1.2 While considering the book, *Stuck in the Shallow End*, the teacher will:

* Discuss with the group the problems of lack of equity and access to computer science.
* Investigate resources available through National Center for Women and Information Technology (NCWIT).
* Develop a set of strategies for overcoming those challenges.

LO1.3 While exploring the Scratch programming environment and the content of Lesson 1.1, the teacher will:

* Experiment with the creative tools in Scratch.
* Discover the wide variety of projects created by members of the Scratch community.
* Share an original project in Scratch or a remixed project with substantial changes with the group.

LO1.4 While working through the APBs of Lesson 1.2, the teacher will:

* Practice pair programming with a partner.
* Use the Agile Scrum framework to develop a mobile app.
* Submit a finished app along with documentation of the development process.

|  |  |
| --- | --- |
|  | Introduction |

This course is designed to follow the College Board’s framework for AP Computer Science Principles. The AP CSP framework consists of seven Big Ideas:

1. Creativity
2. Abstraction
3. Data and Information
4. Algorithms
5. Programming
6. The Internet
7. Global Impact

Along with the Big Ideas, six Computational Thinking Practices have been defined:

1. Connecting Computing
2. Creating Computational Artifacts
3. Abstracting
4. Analyzing Problems and Artifacts
5. Communicating
6. Collaborating

The Big Ideas are the universal themes about computer science that direct the content of this course. The Computational Thinking Practices are the skills and techniques that students will learn to employ throughout the course.

A major goal of this curriculum is to increase access to and strive for more equity in computer science. The content is designed to show students that computer science is for people of all backgrounds and impacts all areas of their lives.

The Project Lead The Way approach to learning is centered around problem-solving. This course uses the PLTW APB method in each lesson. A series of **A**ctivities are used to build skills and knowledge for students, leading to culminating **P**rojects and pro**B**lems where they are put to use. Projects contain high-ceiling tasks with a known end point designed to allow the students to employ what they’ve learned through the activities. Problems use real-world, open-ended tasks that allow students to choose both the tools and methods with which to develop a solution.

In PLTW Computer Science Principles, students learn to use the Agile Scrum software development framework to solve computational problems. This iterative and flexible methodology is the preferred choice among the leaders and innovators in the world of computer science.

The ability to effectively collaborate and work well with partners and in a team is an important skill that’s taught explicitly in the CSP curriculum. Techniques taught for effective collaboration include listening actively, using engaging body language, reflecting, giving feedback, and using professional greetings.

Throughout the course, CSP uses pair programming as a way to encourage the development of these skills and learn to work effectively with a partner. Several tools for collaboration are also introduced, including:

* Scratch in Lesson 1.1. With Scratch, students can remix each other’s work.
* GitHub in Lesson 1.3. In GitHub, developers can track and merge their improvements to software.
* National Center for Biotechnical Information (NCBI) in Lesson 3.2. NCBI allows biologists to share data about DNA and protein and allows people to make discoveries by combining many different scientists’ work.

Creativity is an essential part of computer science and is threaded throughout the course. The ability of computing to enable and enhance opportunities for creative expression is an essential theme in Computer Science Principles.

|  |  |
| --- | --- |
|  | AP CSP Enduring Understandings (EU) and Learning Objectives (LO) |

Problem-solving, collaboration, and creativity are threaded throughout the entire course. This lesson focuses on how they’re taught and used in Lessons 1.1 and 1.2 in the CSP curriculum. Specific concepts that are addressed include:

* Computing facilitates creative expression. EU1.3 through LO1.3.1 [P2].
* Computers execute algorithms using variables and procedures. EU4.1 through LO4.1.1 [P2], EU 5.2 through LO5.2.1 [P3].
* Programming is often a collaborative process. EU1.2 through LO1.2.4 [P6], EU5.1 through LO5.1.3 [P6].
* Teams use programming to create solutions that meet peoples’ needs. EU1.2 through LO1.2.2 [P2], and EU 5.1 through LO5.1.2 [P2] and LO5.1.3 [P6].
* Computers represent numbers, text, sound, and color images with 0s and 1s. EU2.1 through LO2.1.1 [P3] and LO2.1.2 [P5].
* Computers execute algorithms using variables and procedures. EU4.1 through LO4.1.1 [P2] and LO4.1.2 [P5].

|  |  |
| --- | --- |
|  | Part 1: AP CS Principles Framework |

1. Review the AP CS Principles Framework, examining the Big Ideas and the Computational Thinking Practices.
2. Locate the AP CS Principles resources on the AP Central website. Review the structure of the AP CS Principles exam and its various components (Performance Tasks and End of Course Exam).
3. Familiarize yourself with the PLTW CSP Alignment Excel document in the CSP course shell.
4. Sign up for an account on the AP Teacher Community and join the AP CS Principles Teacher Community group.

|  |
| --- |
| Submission Item |
| 1. Capture a screenshot of the main page of the AP CS Principles Teacher community and submit it as a deliverable. |

|  |  |
| --- | --- |
|  | Part 2: Equity In and Access to CS |

Inspiring diverse students to enter computing careers and to leverage computing in all career fields is a primary goal of the course. Throughout Core Training, teachers should share effective practices for broadening participation in computing. The majority of students at most schools are in groups that are underrepresented in the workforce in computer science and information technology. A primary aim of PLTW Computer Science Principles (CSP) is to broaden participation in computing, both in total number of students entering the labor pipeline as well as demographic representation. Broadening participation in computing is needed to ensure equity in jobs as well as equal opportunity to participate in a democracy. Computing education for all is a civil rights issue. Recruiting and retaining diverse students requires intention, reflection, and effort.

Most computer science classes will include students with various levels of background knowledge. Students in different life circumstances have different opportunities to obtain skills outside of school curricula, and creating a class atmosphere in which all students feel welcome and comfortable is crucial. Because students’ experiences with computing varies widely among homes and students, preparatory privilege can produce highly variable achievement levels from prior experience. A common challenge teachers face is how to structure classroom routines and activities so that all students are engaged.

*Stuck in the Shallow End*, by Jane Margolis et al. (2010), makes the case that computer science education is a civil rights issue. Although the book is nominally about race, the issues it raises apply to many demographics of computer science education. The authors’ observations apply in all schools, no matter how homogeneous.

1. Review the following excerpt from pages 81–85 and review your writing about the classroom dynamics in which Ms. Carter’s classroom was dominated by Dan and Ken. Discuss how the teacher could prevent or manage the dynamic described.

Nestled in the hills overlooking the Pacific Ocean, in a predominantly white and wealthy community, sits Canyon Charter High School. … At the time of our study, approximately 43% of the students were white, 24% were African American, 24% were Latino/a, and 8% were Asian American/Pacific Islander. … While there is no gender imbalance in the overall enrollment at Canyon, males made up, on average, 84% of the AP computer science student classroom. … In this chapter, we discuss how, in a school setting as diverse as Canyon, advanced computer science remains insular, dominated primarily by white (male) students. ... During our visits, we found the AP computer science classroom discussion to be routinely dominated by students like Dan and Ken, who came into the course with a firm grasp on much of the subject matter. This happened in the following way, as captured by a summary of our classroom field notes:

On this particular day, when Ms. Carter wrote a problem on the board for the class to work through, three to four of the most tech-savvy white male students shouted out possible methods for solving the problem, argued with each other, then got up from their seats and stood in front of the board to discuss the problem, blocking the view from the rest of [the] class. This classroom discussion then turned into a long private debate among these students and Ms. Carter. The rest of the silent students sat at their desks, listening to the techie students argue while looking at their backs.

To be sure, the eagerness with which the techie students engaged in problem solving at the board was rewarding for Carter, and thus she further encouraged the discussion. She would often enter the fray and ask probing questions, helping to fuel the debate and getting the students at the board to think about alternative ways of solving the problem. Without Carter even realizing it, the end result was that some students engaged in excited argument about a topic they were quite familiar with, while others sat in silence, all but excluded from the conversation. Though the physical blocking of the whiteboard was not necessarily a common occurrence, the reduction of classroom discussion into a private dialogue among the class techies was frequent in our observations, and the consequence was the same. (Chapter 4)

1. Recruiting and retaining diverse students requires intention, reflection, and effort. A network of passionate mentors working together is far more powerful than any one of us working alone.

The following is a list of materials and resources from the National Council for Women in Information Technology (NCWIT) that can help to address the challenges of reaching underrepresented groups:

* [10 Tips to Engage Underrepresented Students](https://www.ncwit.org/resources/top-10-ways-engage-underrepresented-students-computing)
* [10 Tips to Recruit Women into CS Classes](https://www.ncwit.org/resources/top-10-ways-recruiting-high-school-women-your-computing-classes)
* Tips for recruiting and retaining students with [disabilities](https://www.ncwit.org/resources/how-do-you-recruit-or-retain-women-through-inclusive-pedagogy/equal-access-inclusive)
* [Best practices](https://www.ncwit.org/sites/default/files/resources/practices_guide_final.pdf): hands-on, collaborative, problem-based, exploratory activities with approachable staff, strong leadership, and girls-only opportunities
* Use [discussion](https://www.ncwit.org/resources/how-do-you-recruit-or-retain-women-through-inclusive-pedagogy/conversational-classroom).
* Offer an [entry level course exclusively for students without prior experience](https://www.ncwit.org/resources/how-do-you-recruit-or-retain-women-through-inclusive-pedagogy/designing-diversity-case).
* [Rotate participation](https://www.ncwit.org/resources/how-do-you-recruit-or-retain-women-through-inclusive-pedagogy/framing-supportive-classroom) and avoid letting prior knowledge determine participation.
* Give feedback to encourage a [growth mindset](https://www.ncwit.org/resources/ncwit-tips-8-ways-give-students-more-effective-feedback-using-growth-mindset-0).
* Encourage, [encourage](https://www.ncwit.org/resources/how-can-encouragement-increase-persistence-computing/encouragement-works-academic-settings), encourage.
* Beware of invoking [stereotype threat](https://www.ncwit.org/resources/stereotypes-and-stereotype-threat-affect-computing-students) with [well-intentioned but biased messages](https://www.ncwit.org/sites/default/files/resources/howdostereotypethreatsaffectretention.pdf).
* Make the [classroom space](https://www.ncwit.org/resources/how-does-physical-environment-affect-women%E2%80%99s-entry-and-persistence-computing/how-does) colorful, tidy, and inviting.
* Use [targeted recruitment](https://www.ncwit.org/resources/what-are-important-components-targeted-recruiting) of peer groups; consider demographics of where you recruit.
* Use [talking points](https://www.ncwit.org/sites/default/files/resources/tp_youngwomen.pdf) that are known to be effective.
* Engage in [collaborative work](https://www.ncwit.org/resources/how-do-you-retain-women-through-collaborative-learning).

|  |
| --- |
| Submission Item |
| 1. Choose a few resources that you think will be particularly useful and explain how you’ll use them to recruit female and minority students into your class. |

|  |  |
| --- | --- |
|  | Part 3: Creating with Scratch (Lesson 1.1) |

1. Spend some time browsing through the Scratch projects available in the Scratch community. Run the projects to see how they work and look inside them to understand how the developer put the project together.
2. With a partner, plan a new Scratch project or a remix of an existing project you found in the community. Work together to build the project or make changes to the project you remixed. Be ready to share your work with the group when you are finished.

|  |
| --- |
| Submission Item |
| 1. Submit the URL of the Scratch project you completed with your partner. Make sure to set the project to Public. |

|  |  |
| --- | --- |
|  | Part 4: Pair Programming, Agile Scrum, and App Inventor (Lesson 1.2) |

Pair programming is a development workflow that leverages collaboration between two developers to produce a product that will be of higher quality than they could achieve separately. In pair programming, each developer takes on the role of either Driver or Navigator. The Driver operates the keyboard and mouse while the Navigator guides the team through the task at hand. These roles are switched frequently (every 10 to 20 minutes). It’s important to make sure the work is being shared with the team and that it doesn’t become a “worker” and “watcher” experience.

1. Using pair programming, work through Activities 1.2.2 and 1.2.3, familiarizing yourself with MIT App Inventor and the pair programming process.
2. The Agile Scrum design methodology is the design process used throughout this course. Use the resources in Problem 1.1.7 to become familiar with this methodology.
3. As directed by your instructor, explore the concepts involved in Agile Scrum through the scrum4lego game.
4. Problem 1.2.6 is the culminating activity in Lesson 1.2 where you get to employ all the tools and skills you’ve learned so far and develop an app in App Inventor. Working with a partner, use the Agile Scrum methodology you practiced yesterday to plan and develop an app of your own design. Document your planning, including project backlog, sprint planning, sprint review, and backlog grooming. Include screenshots of your finished app and code blocks in your documentation. Work through at least two sprints while pair programming with your partner.

|  |
| --- |
| Submission Item |
| 1. Submit documentation for your App Inventor project for Problem 1.2.6: User Stories, Product Backlog, Sprint Task List, and screen captures of your Blocks and Design view. |

|  |  |
| --- | --- |
|  | Part 5: Considering Classroom Implications |

1. Having students work in pairs can cause challenges in the classroom. Discuss strategies for pairing students and for managing pair-programming dynamics.
2. How will you work with a student who is stuck?
3. Creativity is an important part of CS and these projects. However, students can easily get distracted by non-CS elements and spend a disproportionate amount of time on those elements in the name of creativity. For example, students may spend a long time designing backgrounds or costumes for characters in Scratch and run out of time to actually build the project in the code. Discuss how you can deal with situations like this.
4. Tablets present some logistical issues. Participate in group discussion about issues related to tablet storage, distribution and collection, security, and battery charging. Record some notes about how you might handle these issues.

|  |
| --- |
| Submission Item |
| 1. Write a reflection about the things you’ve learned today. Consider highlighting new things you’ve learned, items you need to consider for implementing this in your classroom, and ideas and suggestions you’ve heard from others. Use the questions in Part 5 of the assignment as prompts, but don’t feel limited or constrained by just those questions. |