Unit 1: Program Design

Content Area: **Computer Science**
Course(s):
Time Period: **Marking Period 1**
Length: **1-2 Weeks**
Status: **Published**

Summary

Students will learn to design a piece of software, or program, and to solve a given problem. They will be able to specify and design algorithms and develop ways to convert those algorithms into a computer program. They will learn to analyze a problem and develop a process by which a solution can be reached before beginning to develop the program.

Revised Date: July 2025

Diversity and Inclusion: Students will focus on equity, inclusion, and tolerance when analyzing the comparison of various quantities regarding characteristics of people. Equality will also be highlighted through the topic of citizenship. This can be associated with treating people fairly and equally.

L.SS.11–12.1	Demonstrate command of the system and structure of the English language when writing or speaking.
9.4.12.CI.1	Demonstrate the ability to reflect, analyze, and use creative skills and ideas (e.g., 1.1.12prof.CR3a).
8.1.12.CS.1	Describe ways in which integrated systems hide underlying implementation details to simplify user experiences.
P.5	Utilize critical thinking to make sense of problems and persevere in solving them.
L.KL.11–12.2	Apply knowledge of language to understand how language functions in different contexts, to make effective choices for meaning or style, and to comprehend more fully when reading or listening.
P.6	Model integrity, ethical leadership and effective management.
8.1.12.CS.4	Develop guidelines that convey systematic troubleshooting strategies that others can use to identify and fix errors.
P.8	Use technology to enhance productivity increase collaboration and communicate effectively.
1.2.12prof.Pr	Producing
8.1.12.AP.1	Design algorithms to solve computational problems using a combination of original and existing algorithms.
9.4.12.IML.3	Analyze data using tools and models to make valid and reliable claims, or to determine optimal design solutions (e.g., S-ID.B.6a., 8.1.12.DA.5, 7.1.IH.IPRET.8).
8.1.12.AP.2	Create generalized computational solutions using collections instead of repeatedly using simple variables.

9.4.12.IML.4	Assess and critique the appropriateness and impact of existing data visualizations for an intended audience (e.g., S-ID.B.6b, HS-LS2-4).
8.1.12.AP.3	Select and combine control structures for a specific application based upon performance and readability, and identify trade-offs to justify the choice.
8.1.12.AP.4	Design and iteratively develop computational artifacts for practical intent, personal expression, or to address a societal issue.
8.1.12.AP.5	Decompose problems into smaller components through systematic analysis, using constructs such as procedures, modules, and/or objects.
8.1.12.AP.6	Create artifacts by using procedures within a program, combinations of data and procedures, or independent but interrelated programs.
8.1.12.AP.7	Collaboratively design and develop programs and artifacts for broad audiences by incorporating feedback from users.
8.1.12.AP.8	Evaluate and refine computational artifacts to make them more usable and accessible.
8.1.12.AP.9	Collaboratively document and present design decisions in the development of complex programs.
1.2.12prof.Re9	Applying criteria to evaluate products.

Essential Questions / Enduring Understandings

Essential Questions:

- How do you understand the problem and the task at hand?
- What are the key goals in designing a program?
- What is an algorithm?
- What problems are addressed in computer science?
- Are there patterns, or repeated steps within the solution?
- Given a probem, what are the steps in defining it?
- What are the goals of a given problem?
- Can a problem be broken down into smaller subtasks?
- What are the key goals in designing a program?
- Given a problems, what components from the existing code are reusable?
- Given a problem, what operations need to be performed in order to solve the problem?

Enduring Understandings:

- Descriptions of tasks should be detailed.
- Goals of a task are specific.
- Programmers must note the process in which they will plan and design a program.
- Reused programs, either in whole or in part, can simplify the process.
- Solving the problems must meet the needs of the user
- Breaking programs into smaller units can streamline the process
- Choosing appropriate data structures and algorithms will assist in completing the task.
- Taking time to analyze programs to determine whether they meet their specifications is an important part of the process.

Objectives

Student will know:

- where to find a problem's description
- where to find a problem's purpose
- where to find a problem's goals
- how to choose test data
- how to debug a program
- how to analyze a program for efficiency

Students will be skilled at:

- being able to design an algorithm
- being able to design programs using approriate code
- defending their procedure for solving a task
- participating in class discussions

Learning Plan

- Preview the essential questions and connect learning throughout the unit.
- Have students determine the purpose of a given problem.
- Have students determine the goals of a given problem.
- Introduce sub-tasks and how they are implemented into programs.
- Using a series of problems, discuss with the students the different operations that will be required in order to solve the problem.
- Discuss with students the importance of reusable code programs. Look at examples of programs and determine what code is being reused. Have students write programs that utilize reusable code.
- Discuss function decomposition.
- Introduce algorithms to the students and their importance to good programs.
- Using algorithms, have students develop programs that meet certain specifications.
- Discuss good techniques in designing a user experience.
- Have students develop programs that include a user interface that is appropriate for its given specifications.

Assessment

- Assessments
    - Formative: Daily assessments using examples from class notes and CodeHS.com, AP Classroom/Albert Checks for Understanding
    - Summative: Teacher-created assessments/projects and CodeHS Computer Science Projects, AP Classroom/Albert Unit Assessments
    - Benchmark: Check for understanding benchmark assessments on CodeHS, AP Classroom/Albert/Khan Academy Diagnostics
    - Alternative Assessments: Student-centered activities such as a doorbell coding project, game design projects, and other activities involving real world applications
        - Complete quizzes/test: Algorithms, Structure of Programs, Design of Programs
        - Be observed by the teacher during individual work on the performance tasks.
        - Conduct self-assessments and reflections
        - Conduct Peer Evaluations.

Materials

- Core instructional materials: Core Book List

  Supplemental materials: CodeHS, computers, and reference books.

Integrated Accommodations and Modifications

See Linked Document.