SYLLABUS DEVELOPMENT GUIDE

AP® Computer Science A

The guide contains the following information:

Curricular Requirements

The curricular requirements are the core elements of the course. A syllabus must provide explicit evidence of each requirement based on the required evidence statement(s).

Required Evidence

These statements describe the type of evidence and level of detail required in the syllabus to demonstrate how the curricular requirement is met in the course.

Note: Curricular requirements may have more than one required evidence statement. Each statement must be addressed to fulfill the requirement.

Samples of Evidence

For each curricular requirement, two to three separate samples of evidence are provided. These samples provide either verbatim evidence or clear descriptions of what acceptable evidence could look like in a syllabus. In some samples, the specific language that addresses the required evidence is highlighted in **bold** text.

Curricular Requirements

CR1	Students and teachers have access to a college-level computer science textbook or resource in print or electronic format.	*See page:* 3
CR2	The course provides opportunities to develop student understanding of the required content outlined in each unit described in the AP Course and Exam Description (CED).	*See page:* 4
CR3	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 1: Design Code, as outlined in the AP Course and Exam Description (CED).	*See page:* 6
CR4	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 2: Develop Code, as outlined in the AP Course and Exam Description (CED).	*See page:* 7
CR5	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 3: Analyze Code, as outlined in the AP Course and Exam Description (CED).	*See page:* 8
CR6	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 4: Document code and computing systems, as outlined in the AP Course and Exam Description (CED).	*See page:* 9
CR7	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 5: Use computers responsibly, as outlined in the AP Course and Exam Description (CED).	*See page:* 10
CR8	The course provides students with hands-on lab experience to practice programming through designing and implementing computer-based solutions to problems.	*See page:* 11

Curricular Requirement 1

Students and teachers have access to a college-level computer science textbook or resource in print or electronic format.

Required Evidence

- ☐ The teacher must select or provide a college-level computer science textbook or resource.

Samples of Evidence

Sample 1

The teacher selects a pre-approved college-level computer science textbook or resource.

Sample 2

The teacher provides the title and author of a college-level computer science textbook or resource.

Curricular Requirement 2

The course provides opportunities to develop student understanding of the required content outlined in each unit described in the AP Course and Exam Description (CED).

Required Evidence

- ☐ The syllabus must include an outline of course content by unit title using any organizational approach to demonstrate the inclusion of required course content.

Note: If the syllabus demonstrates a different approach than the units outlined in the *AP Computer Science A Course and Exam Description* (CED), the syllabus must indicate where the required content of each unit in the CED will be taught.

Samples of Evidence

Sample 1

The course includes the required content organized into the following units based on the AP Course and Exam Description:

Unit 1: Using Objects and Methods

Unit 2: Selection and Iteration

Unit 3: Class Creation

Unit 4: Data Collections

Sample 2

The course follows the given alternative outline not provided in the AP Course and Exam Description:

1. Primitive types, variables, sequential flow (CED Unit 1)
2. Strings (CED Unit 2)
3. Iteration and `ArrayLists` (CED Unit 2 and 4)
4. Selection (CED Unit 2)
5. Methods and Objects (CED Unit 1)
6. Writing Classes (CED Unit 3)
7. File processing (CED Unit 4)
8. Arrays (1D and 2D) (CED Unit 4)
9. Recursion (CED Unit 4)

Sample 3

Course Outline:

Unit	CED Topics
1: Algorithms, Using Classes & Objects, Variables, Input & Output	1.1–1.10, 1.12–1.14
2: Conditionals, `Math` class & `String` class	2.1–2.6, 1.11, 1.15
3: Loops	2.7–2.12
4: Methods & Designing Classes	All of Unit 3
5: 1D & 2D Arrays	4.3–4.5, 4.11–4.13
6: File Reading & `ArrayList`	4.1–4.2, 4.6–4.10
7: Searching, Sorting & Recursion	4.14–4.17

Curricular Requirement 3

The course provides opportunities for students to develop the skills related to Computational Thinking Practice 1: Design Code, as outlined in the AP Course and Exam Description (CED).

Required Evidence

- ☐ The syllabus must include a brief description of an activity or assignment in which students design code to demonstrate a skill from Computational Thinking Practice 1.

Note: The description must explicitly label which skill(s) it addresses. For reference, Skill 1.A and Skill 1.B can be found on the Computational Thinking Practices: Skills page in the CED.

Samples of Evidence

Sample 1

Students **create a program** of their choice that meets the following requirements:

- **Use of at least two classes** in addition to the driver is required. **(Skill 1.A)**
- One class must have at least three private variables with at least one `String` and one `int` or `double`. All necessary accessor and modifier methods must also be included.
- The second class must have a data structure (1D array, 2D array or `ArrayList`) of objects of the first class and must have **a method for accumulating a sum of a numerical value in the objects. (Skill 1.B)**

Sample 2

Decision Lab Project – Students work independently or collaboratively to **design then create a Java program** that prompts the user through a decision-making process. The program should present the user with a series of at least 5 questions that result in an answer or a list of recommendations. The program design should utilize selection in the form of `if`, `if-else`, or `if-else-if` statements. Some examples of project topics are the type of college to attend, the school clubs to join, or the next book to read. **(Skill 1.A)**

Sample 3

Students complete a lab in which they find a comma-separated values (CSV) file and create a program to **answer a question they have by processing the data file**. Students will include their question and the answer their program yielded in a comment inserted in their code. **(Skill 1.B)**

Curricular Requirement 4

The course provides opportunities for students to develop the skills related to Computational Thinking Practice 2: Develop Code, as outlined in the AP Course and Exam Description (CED).

Required Evidence

- ☐ The syllabus must include a brief description of an activity or assignment in which students develop code to demonstrate a skill from Computational Thinking Practice 2.

Note: The description must explicitly label which skill(s) it addresses. For reference, Skill 2.A, Skill 2.B, and Skill 2.C can be found on the Computational Thinking Practices: Skills page in the CED.

Samples of Evidence

Sample 1

Students **write a program that tells the user to wear a sweater, t-shirt, or coat based on a temperature** entered by the user. **(Skill 2.A)**

Sample 2

STEM Lab Project – Students work independently or collaboratively to create an equation/algorithm solver program. Using an equation/algorithm from a math or science course, the Java program will receive user input, **implement a method that returns the equation's/algorithm's solution**, and calls the method to output the solution. **(Skill 2.A and Skill 2.C)**

Sample 3

Students are given a data file containing demographics for various college majors over time, and a program that reads from the file, stores the data in an `ArrayList`, and prints the contents. Students modify this program **to include methods (Skill 2.C)** that **process the `ArrayList` (Skill 2.B)** to:

- **Calculate the total number** of bachelor's degrees obtained. **(Skill 2.A)**
- **Calculate the highest mean salary** (including the year and major) and the lowest mean salary that is not 0 (including the year and major). **(Skill 2.A)**

Curricular Requirement 5

The course provides opportunities for students to develop the skills related to Computational Thinking Practice 3: Analyze Code, as outlined in the AP Course and Exam Description (CED).

Required Evidence

- ☐ The syllabus must include a brief description of an activity or assignment in which students analyze code to demonstrate a skill from Computational Thinking Practice 3.

Note: The description must explicitly label which skill(s) it addresses. For reference, Skill 3.A, Skill 3.B, Skill 3.C, and Skill 3.D can be found on the Computational Thinking Practices: Skills page in the CED.

Samples of Evidence

Sample 1

Students are given a program that contains errors. (Based on what point in the year this activity is used, the teacher should determine if only syntax errors, or if run-time error or logic errors should be included in the code). Students should use a red or other bright colored pen to **make corrections for all the errors found**. Then students should compile their corrected version and test it. This activity is done several times throughout the year. Common student errors are incorporated into the given programs for students to analyze in each unit. This activity is also used to review earlier topics. **(Skill 3.D)**

Sample 2

2D Traversal Lab Project – Students are given a paper copy of Java code that displays the contents of a 2D array while traversing the array in row-major order. Working independently, students

a) **determine the output of the provided program**.
b) modify the program so that it displays the contents of the 2D array while traversing in column-major order.
c) test their modifications creating a program that outputs a 2D array in column-major order.

(Skill 3.A and Skill 3.B)

Sample 3

Students are given short, syntactically correct code fragments that add and remove elements from an `ArrayList`. Students work in pairs to **determine the contents of the `ArrayList` after each code fragment** completes or when an `IndexOutOfBoundsException` occurs. **(Skill 3.B)**

Curricular Requirement 6

The course provides opportunities for students to develop the skills related to Computational Thinking Practice 4: Document code and computing systems, as outlined in the AP Course and Exam Description (CED).

Required Evidence

- ☐ The syllabus must include a brief description of an activity or assignment in which students document code to demonstrate a skill from Computational Thinking Practice 4.

Note: The description must explicitly label which skill(s) it addresses. For reference, Skill 4.A and 4.B can be found on the Computational Thinking Practices: Skills page in the CED.

Note: An activity or assignment that only includes writing comments is not sufficient to meet this requirement. The activity or assignment must specify how comments or descriptions address Skill 4.A or Skill 4.B.

Samples of Evidence

Sample 1

Recursion Tracing Problems Activity – Students are given a paper copy of Java recursive methods with a call to each method. Students create on the paper copy a call stack trace diagram for each recursive method call. Additionally, students **add a statement describing the behavior of each method to arrive at its result**. **(Skill 4.A)**

Sample 2

Students are given a method description from which to brainstorm test cases that can cause the method to fail to produce the desired postcondition. **Class discussion will guide refinement of the methods** to identify concrete test cases for method implementation or **identification of preconditions that students must add as documentation**. **(Skill 4.B)**

Sample 3

Students are required to create methods for standard algorithms (swapping values, finding a sum of n integers, find the average of n integers, etc.). Students must include documentation **comments** for all methods **describing** the purpose **(Skill 4.A)**, `@param` tag for each parameter, and `@return` tag when the method is not void. The `@param` tags are required to **include any restrictions on the values of the parameters that are necessary to ensure the method works as intended**. **(Skill 4.B)**

Curricular Requirement 7

The course provides opportunities for students to develop the skills related to Computational Thinking Practice 5: Use computers responsibly, as outlined in the AP Course and Exam Description (CED).

Required Evidence

- ☐ The syllabus must include a brief description of an activity or assignment in which students explain how computing impacts society, economy, **or** culture.

Note: An activity or assignment that explains one of the three impacts is sufficient to meet this requirement.

The description must explicitly label the skill. For reference, Skill 5.A can be found on the Computational Thinking Practices: Skills page in the CED.

Samples of Evidence

Sample 1

App Showcase – Students are assigned 1–2 partners per grading period to collaborate with to prepare one "app showcase" presentation. The presentation should showcase a unique mobile app **explaining how the software** (and its associated hardware if applicable) **impacts society, economy, or culture**. **(Skill 5.A)**

Sample 2

Students write three 1-page position papers on **how computing impacts a) society, b) economy, and c) culture**. **(Skill 5.A)**

Sample 3

Students are asked to find a current event article discussing a computing innovation (e.g., self-driving cars, smartphones in the classroom) to share with the class. Each student is required to submit one article over the course of the school year. All students are asked to read and annotate the submitted article, **making note of the impacts on society, economy or culture of the computing innovation**, outside of class in preparation for a class discussion. Class discussions are held 1–2 times per month depending on the class size. **(Skill 5.A)**

Curricular Requirement 8

The course provides students with hands-on lab experience to practice programming through designing and implementing computer-based solutions to problems.

Required Evidence

- ☐ The syllabus must include an explicit statement that at least 20 hours of in-class instructional time is spent in computer-based lab experiences.

  AND

- ☐ The syllabus must include titles and descriptions of at least two lab experiences. For each lab, use the label "Lab #1" or "Lab #2" to identify the experience.

Note: If the course uses labs provided by College Board, titles must be included to satisfy this curricular requirement.

Samples of Evidence

Sample 1

Students spend well over 20 hours throughout the course working on a variety of lab projects.

[Lab 1] Data lab: Students will incorporate a real-world data set into a hands-on programming assignment.

[Lab 2] Students search the web and find an actual scenario that involves choosing between one of 3–7 alternatives (such as shipping charge based on amount of purchase, fine for speeding based on amount over the speed limit, etc.), then implement that choice in a `static` method.

Sample 2

Students will complete at least 20 hours of in-class, computer-based lab experiences including the following labs:

Lab #1: Magpie (College Board) Lab
Create a chatbot program as directed in the lab guide so that the program uses selection and nested selection to "chat" with the user.

Lab #2: Create a Case Study Lab
Design, then create, a case study based on a user-made class with methods that store objects to an `ArrayList` and methods that traverse the data, performing at least three (3) algorithms outlined in Topic 4.5, Implementing Array Algorithms.

Lab #3: Word Search Creator Lab
Create a program that creates a word search puzzle that includes hidden words and includes an optional puzzle solution output option.

Sample 3

Students complete at least 3 labs per unit guaranteeing at least 20 hours of in-class instructional time is spent in computer-based lab experiences. The table indicates examples from a few different units.

Course Unit	Example Labs
1: Algorithms, Using Classes & Objects, Variables, Input & Output	TempConverter: Create a program that prompts the user for a temperature in Fahrenheit and displays the equivalent temperature in Celsius. (Lab 1)
2: Conditionals, Math class & String class	CountVowels: Create a program to count the number of vowels in a string entered by the user. The string entered may contain spaces. Both uppercase and lowercase vowels should be counted. (Lab 2)
3: Loops	Nested Loop Investigation: Students are given a class that contains several methods. Some of the methods contain implementation code that uses nested loops, while other methods need to be implemented with a nested loop to produce the specified output.
4: Methods & Designing Classes	DiceRollGame: Students work with a partner to design and implement an object-oriented dice game using a Dice class, a Player class, and a Game class.
5: 1D & 2D Arrays	Picture Lab: Students complete the activities in Activity 5 and Activity 6 in the College Board–provided lab to practice manipulating a 2D array of objects.
6: File Reading & `ArrayList`	WordCounter: Students implement a class method that counts the number of "words" in a file specified by the `String` parameter that contains a file path and name. “Words”, in the file are separated by whitespace, not necessarily a single space. Students implement a second-class method that counts the number of times a specified word occurs in a file. The file and specified word are parameters to the method. The words on each line of this file will be separated by a single space.
7: Searching, Sorting & Recursion	Slurpy: Students are given the skeleton of a class and implement three methods, two recursive and one iterative, to determine whether a given string fits the rules of being a slurpy.