SYLLABUS DEVELOPMENT GUIDE

AP® Computer Science A

The guide contains the following sections and 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).

The Unit Guides and the "Instructional Approaches" section of the *AP® Computer Science A Course and Exam Description* (CED) may be useful in providing evidence for satisfying these curricular requirements.

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, 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.

Curricular Requirements

CR1	Students and teachers have access to a college-level computer science textbook in print or electronic format	*See page:* 1
CR2	The course provides opportunities to develop student understanding of the required content outlined in each of the units described in the AP Course and Exam Description (CED).	*See page:* 4
CR3	The course provides opportunities to develop student understanding of the big ideas.	*See page:* 7
CR4	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 1: Program Design and Algorithm Development.	*See page:* 9
CR5	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 2: Code Logic.	*See page:* 10
CR6	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 3: Code Implementation.	*See page:* 11
CR7	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 4: Code Testing.	*See page:* 12
CR8	The course provides opportunities for students to develop the skills related to Computational Thinking Practice 5: Documentation.	*See page:* 13
CR9	This course provides students with hands-on lab experiences to practice programming through designing and implementing computer-based solutions to problems.	*See page:* 14

Curricular Requirement 1

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

Required Evidence

- ☐ The syllabus must list the title and author of a college-level computer science textbook.

Samples of Evidence

1. The following textbooks are used in the course:

   J. Bergin, M. Stehlik, J. Roberts, R. Pattis, *Karel J Robot: A Gentle Introduction to the Art of Object-Oriented Programming in Java*, Dreamsongs Press, 2013 edition, ISBN: 978-0970579515.

   H. Schildt, *Beginner's Guide*, McGraw-Hill Education, 6 Edition, 2014, ISBN: 978-0071809252.

2. Online text: *Think Java* by Allen B Downey & Chris Mayfield.

3. Textbook: Cay Horstmann's *Java Concepts*, 3rd Edition (2016).

Curricular Requirement 2

The course provides opportunities to develop student understanding of the required content outlined in each of the units 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

1. The course includes lessons on:
   - Declaration of three primitive types: `int`, `double`, and `boolean`, arithmetic operations on numeric types, and Boolean expressions using Boolean types **(Unit 1, VAR)**
   - How to instantiate objects for user-defined and built-in classes, store or update data in objects, and invoke methods to extract and manipulate data. **(Unit 2, VAR)**
   - Use of Boolean expressions in `if` statements to control the statement executions. **(Unit 3, CON)**
   - Repetition structures using the `while` and `for` constructs. This includes `for` loop initialization, continuation condition, and update that will eventually terminate, `while` loop structures and flow of control, nesting of loops, selection statements and flow control. **(Unit 4, CON)**
   - Writing class definitions, creating objects as instances of the class using constructors, manipulating data (fields), using behaviors (methods) for the object, and class (static) variables and methods. **(Unit 5, MOD)**
   - Array (both 1D and 2D) to store a sequence of values of the same type instead of declaring individual variables for each value, declaring arrays and accessing array elements using array index, traversing arrays without bounds errors using loops, and using the enhanced `for` loop for accessing array elements. **(Unit 6, Unit 8, VAR, CON)**
   - The `ArrayList` class and instantiating objects of this type, and basic operations using the Quick Reference for commonly used methods **(Unit 7, VAR)**
   - Creating complete classes from a given class hierarchy, extending classes using inheritance, overriding methods, using inherited methods, data abstraction, and polymorphism **(Unit 9, MOD)**
   - On implementing recursion, how repeated function calls create iteration, and stopping criteria in recursion, recursive binary searching, and iterative sorting. **(Unit 10, CON)**

2. The course follows the given alternative outline not provided in the AP Course and Exam Description:
    1. Introduction to OOP (CED Unit 1)
        a. How to write a Java program
        b. Simple I/O
    2. Data (CED Unit 1)
        a. Primitive Variables
        b. Basic Strings
        c. Expression and Assignment Statements
        d. Casting
    3. Designing and Using Classes (CED Units 2, 5)
        a. Writing Classes
        b. Documentation
        c. Methods
        d. Java API and Libraries (Math, String, Integer, Double, Graphics)
    4. Control Structures (CED Unit 3)
        a. Boolean
        b. Logical and Relational Operators
        c. If, if-else and nested ifs
    5. Iterations (CED Unit 4)
        a. For loops
        b. While loops
        c. Recursion
        d. Standard Algorithms
    6. Advanced Strings (CED Units 2, 3, 4)
        a. String Methods
        b. Iterating over a string
        c. Standard Algorithms
    7. Arrays (CED Units 6, 8)
        a. 1D
        b. 2D
        c. Enhanced for Loops
        d. Standard Algorithms
    8. `ArrayLists` (CED Unit 7)
        a. Methods
        b. Enhanced for Loops
        c. Standard Algorithms
    9. Searching and Sorting Data Structures (CED Units 7, 10)
        a. Linear Search
        b. Binary Search
        c. Selection Sort
        d. Insertion Sort
        e. Merge Sort

10. Inheritance (CED Unit 9)
    a. Writing Superclasses and Subclasses
    b. Overriding
    c. Hierarchies
    d. Polymorphism

Notes: The following topics are taught throughout the course as appropriate:

Exception (CED Units 2, 6, 8)

Ethical and Social Implications (CED Units 5. 7)

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

    Unit 1: Primitive Types

    Unit 2: Using Objects

    Unit 3: Boolean Expressions and if Statements

    Unit 4: Iteration

    Unit 5: Writing Classes

    Unit 6: Array

    Unit 7: `ArrayList`

    Unit 8: 2D Array

    Unit 9: Inheritance

    Unit 10: Recursion

Curricular Requirement 3

The course provides opportunities to develop student understanding of the big ideas, as outlined in the AP Course and Exam Description (CED).

Required Evidence

- ☐ The syllabus must include four student activities, each describing how it relates to one of the four big ideas. All of the big ideas must be represented.
- ☐ Each activity must be labeled with the related big idea(s).

Samples of Evidence

1. The following lessons are provided:
   - How to solve a problem by breaking it into smaller problems that can be assembled together as a solution for the original problem. **(MOD)**
   - How to store data in computer memory using variable, variable names for memory locations, declaration of primitive type variables, arithmetic operation on numeric data types, and data structures for storing and manipulating data. **(VAR)**
   - Selection and repetition structures, and how problem solving can use nested repetition and selection controls. **(CON)**
   - Discussion on the protection of privacy, defending against crimes on the internet, the consequence of spreading and using computer viruses for causing harm to users, and copywrite law violation involved in using or copying a program without permission. **(IOC)**

2. Students will spend a minimum of 20 hours of instructional time engaged in hands-on lab experiences through the use of the provided College Board labs as suggested in the unit guides in the AP Course and Exam Description. The labs will be used after the following units:
   - Unit 4: Consumer Review Lab **(CON)**—During the open-ended project, students will demonstrate their understanding of control structures.
   - Unit 7: Data Lab **(VAR, IOC)**—During the open-ended project, students will demonstrate their understanding of the use of data structures. Also, students will consider the ethical and social implications of storing and analyzing data.
   - Unit 8: Steganography Lab **(VAR)**—During the open-ended project, students will demonstrate their understanding of the use of 2D arrays.
   - Unit 9: Celebrity Lab **(MOD)**—During the open-ended project, students will demonstrate their understanding of inheritance and creating classes.

3. Big Idea #1, MODULARITY (MOD): Students will design and write a program to determine if a square 2D array is a Magic Square. A Magic Square is defined as a 2D array with n rows and n columns. Each element is a unique number from 1 to n2. The sum of each row, each column and each diagonal is the same value. The students will create a class for Magic Square and break down each check for the square into appropriate methods.

   Big Idea #2, VARIABLES (VAR): Students will be assigned to create a program of their choice that meets the following requirements:
   - Use of at least two classes in addition to the driver is required.
   - One class must have at least three private variables with at least one String and one int/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 the following:

- A method for accumulating a sum of a numerical value in the object
- The ability to find either a min or max of a numerical value in the object
- The ability to locate a specific object and return the index of that object
- One other array method related to the objects

Big Idea #3, CONTROL (CON): Students will write a program to simulate playing rock, paper, scissors against the computer. The computer will randomly choose rock, paper, or scissors. The player will input his/her choice. Based on the choices, the program will output the winner and then ask if the user wants to play again. If so, the game is played again. If not, the overall results for the games played will be output.

Big Idea #4, IMPACT OF COMPUTING (IOC): Students will form groups of 3–4 people. Each student will find and read an article about a data breach in the last year. Students will summarize the article they read and include the following information:

- Name and link of article
- Date of article
- Explain the data breach discussed in the article
- Identify the number of people affected and who was affected by the breach

Each student will share the summary with the other members of the group. Then as a group, students will determine what the articles have in common and do some research on ways computer science can be used to help avoid these breaches in the future.

Curricular Requirement 4

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

Required Evidence

- ☐ The syllabus must include a brief description of an instructional approach (e.g., activity or assignment) describing how students will engage with one skill (skill 1.A, 1.B, or 1.C) in Computational Thinking Practice 1.
- ☐ Instructional approaches must explicitly label which skill(s) they address.

Samples of Evidence

1. Students will solve a problem by writing an algorithm and converting it to a Java program, and practice procedural decomposition for problem solving. **(Skill 1.A)**

2. Parsons Puzzles

   Create methods for standard algorithms (swapping values, finding a sum of n integers, find the average of n integers, etc.). Remove the lines of code you want the students to determine in this exercise. Provide the lines of code that are necessary to complete the methods on strips of paper (one line of code per slip). Extra lines of incorrect code can be included among the slips. Have the students work in pairs to complete the code "puzzle." This process can be used early on to help students become familiar with proper code and syntax. This is also a nice way to help students at the beginning of each new topic to gain confidence. As students become more skilled, the "puzzle pieces" can be taken away and students can write out the missing lines on their own. **(CTP 1, Skill 1.B)**

3. The syllabus includes: Students will complete an assignment to find library methods for certain problem tasks and write methods for other tasks. **(P1, 1.C)**

Curricular Requirement 5

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

Required Evidence

- ☐ The syllabus must include a brief description of an instructional approach (e.g., activity or assignment) describing how students will engage with one skill (skill 2.A, 2.B, 2.C, or 2.D) in Computational Thinking Practice 2.
- ☐ Instructional approaches must explicitly label which skill(s) they address.

Samples of Evidence

1. Students are provided lessons on the order of execution of programs, what conditions get checked during program execution, how to impose program execution controls, and the use of flow diagrams to understand program logic in program code without method calls. **(Skill 2.B)**

2. Bell-ringer or exit slip activity **(CTP 2)**. Provide students code for a method. Then give the student an example of calling that method (with parameter values if appropriate) to determine what the method will return or output. Some examples may be very straightforward, while others may not work the way students (or even the programmer) may expect. These more difficult examples will help students read code more carefully and possibly be more careful when they write their own code.

   Sample code **(Skill 2.C)**:

   ```
   public void removeNum(int num)
   {
       for(int k = 0; k < scores.size(); k++)
       {
           if(scores.get(k) < num)
               scores.remove(k);
       }
       System.out.println(scores);
   }
   ```

   Given the method above is in a class with a private instance variable, `ArrayList` <Integer>scores and scores contains the values: {75, 45, 34, 56, 50, 98, 49}

   What will be printed if the following method call is executed in the class?

   removeNum(50);

3. The course includes multiple assignments to analyze programs:
   - Operator precedence **(Skill 2.A)**
   - Big-O for sort and search algorithms, including best/worse/average cases **(Skill 2.D)**

Curricular Requirement 6

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

Required Evidence

- ☐ The syllabus must include a brief description of an instructional approach (e.g., activity or assignment) describing how students will engage with one skill (skill 3.A, 3.B, 3.C, 3.D, or 3.E) in Computational Thinking Practice 3.
- ☐ Instructional approaches must explicitly label which skill(s) they address.

Samples of Evidence

1. The syllabus must provide lessons on writing programs that:
    - build user-defined classes **(Skill 3.B)**
    - use object instantiation and method access using the dot operator **(Skill 3.A)**
    - involve nested iteration and selection constructs **(Skill 3.C)**
    - require the use of 1D array, and `ArrayList` objects, and array iterations **(Skill 3.D)**
    - use 2D arrays and array traversal by row and column majors, and diagonally **(Skill 3.E)**

2. Students will be assigned to create a program of their choice that meets the following requirements:
    - Use of at least two classes in addition to the driver is required.
    - One class must have at least three private variables with at least one String and one int/double. All necessary accessor and modifier methods must also be included.
    - The second class must have a data structure (1D array or `ArrayList`) of objects of the first class and must have the following: **(Skill 3.D)**
        - A method for accumulating a sum of a numerical value in the object
        - The ability to find either a min or max of a numerical value in the object
        - The ability to locate a specific object and return the index of that object.
        - One other array method related to the objects.

3. The syllabus includes a game project assignment, which will use 2D arrays and methods that will require expressions, conditionals, and iteration. **(Skill 3.E)**

Curricular Requirement 7

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

Required Evidence

- ☐ The syllabus must include a brief description of an instructional approach (e.g., activity or assignment) describing how students will engage with one skill (skill 4.A, 4.B, or 4.C) in Computational Thinking Practice 4.
- ☐ Instructional approaches must explicitly label which skill(s) they address.

Samples of Evidence

1. The syllabus must provide lessons on unit testing by creating test cases which may be written before coding and used during program execution. **(Skill 4.A)**

2. 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 enter the corrected version in an IDE and test the new version to see if the code runs and outputs the correct information. 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. **(CTP 4, Skill 4.B)**

3. The course outline includes a lab in which half of the students are asked to code a method to given specifications using a for loop and the other half are asked to code a method to the same specifications using while loop. Then a student from each group is paired together to determine if their methods produce equivalent output. **(Skill 4.C)**

Curricular Requirement 8

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

Required Evidence

- ☐ The syllabus must include a brief description of an instructional approach (e.g., activity or assignment) describing how students will engage with one skill (skill 5.A, 5.B, 5.C, or 5.D) in Computational Thinking Practice 5.
- ☐ Instructional approaches must explicitly label which skill(s) they address.

Samples of Evidence

1. Students are given completed methods and asked to determine appropriate:
   - preconditions **(Skill 5.D)**
   - postconditions **(Skill 5.A)**

2. **(Skill 5.D)** Students work on assignments (such as below) when covering data structures.

   For each method given, write the precondition(s) that must be true for the method to run without an error.

```
public static int findMax(int[] scores)
{
    int max = scores[0];
    for(int num: scores)
    {
    if (num > max)
       max = num;
    }
    return max;
}
public static double findAverage(int start, int end, ArrayList <Double>scores)
{
    double sum = 0.0;
    for(int k = start; k <= end; k++)
        sum += scores.get(k);
    return sum/scores.size();
}
```

3. Provide students with program code containing compiler, run-time, and logic errors. Students are asked to explain why each example will not produce the intended results. **(Skill 5.B)**

Curricular Requirement 9

This course provides students with hands-on lab experiences 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. In addition, the syllabus must include titles and descriptions for at least two labs. Labs must be explicitly labeled.

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

Samples of Evidence

1. Students will spend at least 20 hours working on the following programming problems:
   - Magpie lab: This lab implements a chatbot and incorporates String manipulations.
   - Tic-Tac-Toe lab: Students use 2D arrays to create a tic-tac-toe game.
   - Data lab: Students will incorporate a real-world data set into a hands-on programming assignment.

2. Students will work on labs during class 2–3 days per week, which exceeds the 20 hours required for the course. Below are some labs completed during these hours:
   - Lab 1: CoinChange—Students write a program to determine how many of each coin is necessary to give the input change with the smallest number of coins.
   - Lab 2: Lottery—Students write Hopper, Ball, and Picker classes to create a program to simulate the lottery.
   - Lab 3: DrawHouse—Students use a Java Drawing Tool to call on existing methods in order to create new methods to create a picture containing a house.
   - Lab 4: BankAccount—Students write a class to simulate different types of bank accounts and transactions.
   - Lab 5: Statistics—Students write classes using 1D arrays and `ArrayList` to calculate various statistical measures including mean, median, mode, standard deviation, minimum, and maximum.
   - Lab 6: Yahtzee—Students design and implement their own version of a Yahtzee game using various data structures.
   - Lab 7: Corporation—Students create their own business with different types of employees that all are subclasses of the employee class.

3. The course will use the labs provided by College Board as suggested in the unit guides:
   - Consumer Review Lab
   - Data Lab
   - Steganography Lab
   - Celebrity Lab