The AP Computer Science (AP CS) course is an introductory course in computer science. Because the design and implementation of computer programs to solve problems involve skills that are fundamental to the study of computer science, a large part of the AP CS course is built around the development of computer programs that correctly solve a given problem. These programs should be understandable, adaptable, and, when appropriate, reusable. At the same time, the design and implementation of computer programs is used as a context for introducing other important aspects of computer science, including the development and analysis of algorithms, the development and use of fundamental data structures, the study of standard algorithms and typical applications, and the use of logic and formal methods. In addition, the responsible use of these systems is an integral part of the course. The topic outline summarizes the content of the AP CS curriculum. In this section, we provide more details about the topics in the outline.

Topic Outline

Following is an outline of the major topics covered by the AP Computer Science Exams. This outline is intended to define the scope of the course but not necessarily the sequence.

I. Object-Oriented Program Design

The overall goal for designing a piece of software (a computer program) is to correctly solve the given problem. At the same time, this goal should encompass specifying and designing a program that is understandable, can be adapted to changing circumstances, and has the potential to be reused in whole or in part. The design process needs to be based on a thorough understanding of the problem to be solved.

Scope

A. Program design

1. Read and understand a problem description, purpose, and goals.
2. Apply data abstraction and encapsulation.
3. Read and understand class specifications and relationships among the classes (“is-a,” “has-a” relationships).
4. Understand and implement a given class hierarchy.
5. Identify reusable components from existing code using classes and class libraries.

B. Class design

1. Design and implement a class.
2. Choose appropriate data representation and algorithms.
3. Apply functional decomposition.
4. Extend a given class using inheritance.

II. Program Implementation

The overall goals of program implementation parallel those of program design. Classes that fill common needs should be built so that they can be reused easily in other programs. Object-oriented design is an important part of program implementation.

A. Implementation techniques
1. Methodology
a. Object-oriented development
b. Top-down development
c. Encapsulation and information hiding
d. Procedural abstraction
B. Programming constructs
1. Primitive types vs. objects
2. Declaration
a. Constant declarations
b. Variable declarations
c. Class declarations
d. Interface declarations
e. Method declarations
f. Parameter declarations
3. Console output
(System.out.print/println)
4. Control
a. Methods
b. Sequential
c. Conditional
d. Iteration
e. Understand and evaluate recursive methods recursive solutions
C. Java library classes (included in the A-level AP Java Subset)

III. Program Analysis
The analysis of programs includes examining and testing programs to determine whether they correctly meet their specifications. It also includes the analysis of programs or algorithms in order to understand their time and space requirements when applied to different data sets.
A. Testing
1. Test classes and libraries in isolation.
2. Identify boundary cases and generate appropriate test data.
3. Perform integration testing.
B. Debugging
1. Categorize errors: compile-time, run-time, logic.
2. Identify and correct errors.
3. Employ techniques such as using a debugger, adding extra output statements, or hand-tracing code.
C. Understand and modify existing code

D. Extend existing code using inheritance
E. Understand error handling
1. Understand runtime exceptions.
F. Reason about programs
1. Pre- and post-conditions
2. Assertions
G. Analysis of algorithms
1. Informal comparisons of running times
2. Exact calculation of statement execution counts
H. Numerical representations and limits
1. Representations of numbers in different bases
2. Limitations of finite representations (e.g., integer bounds, imprecision of floating-point representations,and round-off error)

IV. Standard Data Structures
Data structures are used to represent information within a program. Abstraction is an important theme in the development and application of data structures.
A. Simple data types (int, boolean, double)
B. Classes
C. One-dimensional arrays

V. Standard Algorithms
Standard algorithms serve as examples of good solutions to standard problems. Many are intertwined with standard data structures. These algorithms provide examples for analysis of program efficiency.
A. Operations on A-level data structures previously listed
1. Traversals
2. Insertions
3. Deletions
B. Searching
1. Sequential
2. Binary
C. Sorting
1. Introduction to sorting

VI. Computing in Context
An awareness of the ethical and social implications of computing systems is necessary for the study of computer science. These topics need not be covered in detail but should be considered throughout the course.
A. System reliability
B. Privacy
C. Legal issues and intellectual property
D. Social and ethical ramifications of computer use