AP Computer Science A

The AP Computer Science A course is an introductory computer science course. A large part of the course involves developing the skills to write programs or parts of programs that correctly solve specific problems. The course also emphasizes the design issues that make programs understandable, adaptable, and when appropriate, reusable. At the same time, the development of useful computer programs and classes is used as a context for introducing other important concepts in computer science, including the development and analysis of algorithms, the development and use of fundamental data structures, and the study of standard algorithms and typical applications. In addition, an understanding of the basic hardware and software components of computer systems and the responsible use of these systems are integral parts of the course.

The goals of the AP Computer Science A course are comparable to those in the introductory sequence of courses for computer science majors offered in college and university computer science departments. Students completing the AP Computer Science A course will be able to:

- Design and implement computer-based solutions to problems in a variety of application areas.
- Use and implement commonly-used algorithms and data structures.
- Develop and select appropriate algorithms and data structures to solve problems.
- Code fluently in an object-oriented paradigm using the programming language Java. Students will be familiar with and will be able to use standard Java library classes from the AP Java subset.
- Read and understand a large program consisting of several classes and interacting objects. Students will be able to read and understand a description of the design and development process leading to a program such as the AP Computer Science Labs.
- Identify the major hardware and software components of a computer system, their relationship to one another, and the roles of these components within the system.
- Recognize the ethical and social implications of computer use.

This course spans 32 weeks, requiring a minimum of 10 hours per week to read lessons and complete 54 programming assignments, 23 quizzes, six written assignments, six Discussion-Based Assessments, and 12 exams. The AP-approved IMACS curriculum is used to develop fundamental programming knowledge and skills, but enhanced instruction is provided to more thoroughly explore computer science concepts.

Resources:

- Institute of Mathematics and Computer Science (IMACS): ***Computer Science: Java Programming*** (www.eimacs.com).
- Institute of Mathematics and Computer Science (IMACS): ***Be Prepared for the AP Computer Science Exam*** (www.eimacs.com).
- Litvin & Litvin, ***Java Methods A & AB***, Andover, MA. Skylight Publishing 2006.
- College Board, ***AP Computer Science Labs: Magpie, Picture and Elevens,*** and the ***Student Manual***.
- The BlueJ Integrated Development Environment (bluej.org).

In the following detailed syllabus, the **Topic Outline** column identifies the concepts from the AP Computer Science Topic Outline, the **Assignments** column indicates assignments that address the concepts, and **the IMACS Lessons** column identifies specific IMACS lessons that address the concepts. The combination of IMACS lessons and specific assignments permits coverage of all concepts in the **Topic Outline**.

---*Semester 1*---

Module	Scope and Sequence	Topic Outline	Assignments	IMACS Lessons
1	**Getting Started**	**Program Implementation** ▪ Console output (System.out.print/println)-IIB3	▪ Download and install Java ▪ Download and install BlueJ ▪ Print student information card	
2	**Variables and Expressions**	**Program Implementation** ▪ Variable Declarations: IIB2b ▪ Control – Sequential: IIB4b **Program Analysis**	▪ Calculations with int variables ▪ Calculations with double variables ▪ Calculate test grade averages ▪ Convert currencies (dollar, pesos, yen, and Euro)	**Variables and Arithmetic Expressions** ▪ Integers 1, 2, 3, 4 ▪ Doubles 1, 2

		▪ Identify and correct errors: IIIB2 ▪ Limitations of finite representations (e.g., integer bounds, imprecision of floating-point representations, and round-off error): IIIF2 **Standard Data Structures** ▪ Simple data types (int and double): IVA		▪ Arithmetic Expressions 1, 2, 3, 4, 5 ▪ Declaring and Assigning Values to Variables ▪ Casting 1, 2 ▪ Pitfalls and Surprises 1, 2 ▪ Programming Shortcuts 1, 2, 3 **Test #1 and #2**
3	**Strings and User Input**	**Program Implementation** ▪ Primitive types vs. objects: IIB1 ▪ Variable Declarations: IIB2b ▪ Java library classes (String): IIC **Program Analysis** ▪ Identify and correct errors: IIIB2	▪ Using Pseudocode ▪ Reading Source Code ▪ Interpreting a secret message written in ASCII ▪ Using escape characters to create ASCII art ▪ Exploring the Java API ▪ Decoding cell phone text message phrases ▪ Modify text message and currency converter to accept user input with Scanner class methods ▪ Create a receipt for a purchase based on user input ▪ Discussion-based Assessment (Modules 1–3) ▪ Challenge Exam (Module 1–3)	**Variables and Arithmetic Expressions** ▪ Strings 1, 2 ▪ Concatenation 1, 2 ▪ String Methods 1, 2, 3 ▪ Displaying Messages ▪ Converting between numbers and Strings **Test #3**

4	**Condition Statements**	**Program Implementation** ▪ Control – Conditional: IIB4c **Program Analysis** ▪ Representations of numbers in different bases: IIIF1 **Standard Data Structures** ▪ Simple data types (boolean): IVA	▪ Practice base conversions (binary, octal, decimal, and hexadecimal) ▪ Calculate basal metabolic heart rate ▪ Calculate body mass index ▪ Calculate total daily energy expenditure	**Variables and Arithmetic Expressions** ▪ Integers 5 ▪ Booleans ▪ Relational Operators 1, 2 ▪ Comparing Strings 1, 2, 3 ▪ Logical Operators 1, 2, 3, 4, 5 Program Control ▪ Conditional Statements 1, 2, 3, 4 ▪ Blocks 1, 2 **Practice Test #4 and #6**
5	**Loops**	**Program Implementation** ▪ Control – Iteration: IIB4d **Program Analysis** ▪ Identify boundary cases and generate appropriate data: IIIA2 ▪ Employ techniques such as using a debugger, adding extra output statements, or hand tracing code: IIIB3 ▪ Identify and correct errors: IIIB2 ▪ Limitations of finite representations (e.g., integer	▪ Simulate tossing a fair or biased coin ▪ Predict percentage of males and females in a population ▪ Debug a guess my number game ▪ Read text files ▪ Calculate percentage of male/female, male/male, female/female family combinations read from a sample text file ▪ Calculate outcomes of three-ball lottery combinations ▪ Predict odds of an item using the Monte Carlo Method	**Program Control** ▪ Iteration ▪ While Loops 1, 2, 3 ▪ For loops 1, 2, 3 **Practice Test #7 and #8**

		bounds, imprecision of floating-point representations, and round-off error): IIIF2	▪ Generate random passwords with different character sets ▪ Discussion: Computer science careers based on Univ. of Washington videos	
6	**Arrays**	**Program Implementation** ▪ Control – Iteration: IIB4d **Program Analysis** ▪ Understand and modify existing code: IIIA1 ▪ Identify and correct errors: IIIB2 ▪ Limitations of finite representations (e.g., integer bounds, imprecision of floating-point representations, and round-off error): IIIF2 **Standard Data Structures** ▪ Arrays (one dimension): IVE	▪ Calculate average temperature and total rainfall and choose output in metric or English units ▪ Format output using the printf method ▪ Calculate and display average, maximum, and minimum hurricane speed, pressure, and category based on data read in from a text file ▪ Challenge Exam (Module 4–6) ▪ Discussion-based assessment (Modules 4–6)	**Variables and Arithmetic Expressions** ▪ Arrays 1, 2, 3, 4, 5, 6, 7, 8 **Program Control** ▪ For-each loops 1, 2, 3, 4 **Test #5 and #9**
7	**Methods**	**Program Implementation** ▪ Top-down development: IIA1 ▪ Procedural abstraction: IIA5 ▪ Method declarations: IIB2c	▪ Calculate x, y coordinates on the circumference of a circle ▪ Calculate the surface gravity on each planet	**Methods** ▪ Static Methods 1, 2, 3 ▪ Defining New Static Methods 1, 2, 3, 4, 5, 6, 7, 8

		▪ Parameter declarations: IB2c ▪ Control – Methods: IIB4a ▪ Java library classes (Math): IIC **Program Analysis** ▪ Limitations of finite representations (e.g., integer bounds, imprecision of floating-point representations, and round-off error): IIIF2	▪ Calculate individual weight on each planet ▪ Approximate the value of pi by simulating throwing darts	▪ The `main` Method ▪ Comments ▪ Multiple Variable Declarations **Test #10**
8	**Introduction to OOP and Classes**	**Object-Oriented Program Design** ▪ Read and understand problem description, purpose, and goals: IA1 ▪ Apply data abstraction and encapsulation: IA2 ▪ Design and implement a class: IA3 ▪ Apply functional decomposition: IA6 **Program Implementation** ▪ Object-oriented development: IIA3 ▪ Encapsulation and information hiding: IIA4	▪ Identify examples of real-world classes, objects, methods, and attributes ▪ Evaluate OOP, procedural, and non-procedural styles ▪ Establish an implementation class for an object including instance variables and methods ▪ Create a project with multiple objects stored using an array ▪ Create program documentation using Javadocs ▪ Calculate net carbon dioxide footprint based on levels of recycling ▪ Discussion: Computer Modeling ▪ Challenge Exam (Module 7–8)	**Object-oriented Programming Concepts** ▪ OOP and Java ▪ Classes and Instances 1, 2, 3 **Simple Objects** ▪ Person Class 1, 2, 3, 4, 5, 6, 7, 8 ▪ Point Class 1, 2, 3, 4, 5, 6 ▪ Public Classes and the Java Compiler ▪ The Java Compiler and the Virtual Machine ▪ Errors, Exceptions, and Garbage Collection ▪ Arrays of Objects 1, 2, 3

		▪ Primitive types vs. objects: IIB1 ▪ Class declarations: IIB2d ▪ Java class libraries (ArrayList): IIC **Program Analysis** ▪ Categorize errors: compile-time, runtime, logic: IIIB1 **Standard Data Structures** ▪ Classes: IVC ▪ Lists IVD ▪ Arrays IVE		▪ ArrayLists 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12 **Java Basics** ▪ Overloading Methods 1, 2 **Test #12 and #13**
9	**Computer Systems and History**	**Standard Data Structures** ▪ Arrays (2-d): IVE **Computing in Context** ▪ System reliability VIA	▪ Distinguish between analog and digital computing ▪ Conduct a family computer hardware and software inventory ▪ Investigate computer pioneers and devices of early computer history ▪ Explore a timeline of the generations of computers and correlate to historical/cultural events ▪ Calculate projectile trajectory table based on launch angle and speed	**Computing in Context** ▪ Hardware ▪ Systems and System Software **Variables and Arithmetic Expressions** ▪ Arrays 6, 7, 8 ▪ Test 5

10	**Semester Exam**	▪	▪ Discussion-based assessment (Modules 7–10) ▪ Semester Exam	▪ **Review IMACS practice tests.**

---Semester 2---

11	**Computing in Context**	**Computing in Context** ▪ System reliability: VIA ▪ Privacy: VIB ▪ Legal issues and intellectual property: VIC ▪ Social and ethical ramifications of computer use: VID	▪ Discuss a family identify theft prevention plan ▪ Conduct a computer security audit ▪ Discuss online safety and intellectual property issues ▪ AP Computer Science Labs introduction	
12	**Recursion**	**Program Implementation** ▪ Understand and evaluate recursive methods: IIB4e ▪ **AP Computer Science Lab Student Guide: Magpie**	▪ Discuss examples of real-world recursion ▪ Create Mondrian art using recursion ▪ Visualize the recursive leap of faith (the Towers of Hanoi and Martin and the Dragon) ▪ Translate piecewise functions into recursive methods ▪ Calculate Fibonacci numbers recursively	**Java Basics** ▪ Recursive Methods 1, 2, 3, 4

			▪ Collaboratively decode a secret message ▪ AP Computer Science Lab: Magpie ▪ Challenge Exam Module 12	
13	**Inheritance and Polymorphism**	**Object-Oriented Program Design** ▪ Read and understand class specifications and relationships among the classes ("is-a," "has-a" relationships): IA3 ▪ Understand and implement a given class hierarchy: IA3 ▪ Identify reusable components from existing code using classes and class libraries: IA4 ▪ Extend a given class using inheritance: IA3 **Program Implementation** ▪ Java library classes (Object): IIC **Program Analysis** ▪ Test classes and libraries in isolation: IIIA1	▪ Extend a box class to create a cube class ▪ Extend a triangle class to create equilateral and isosceles classes ▪ Create a class hierarchy to represent simple terrains in a graphics game ▪ AP Computer Science Lab: Magpie ▪ Discussion-based assessment (Modules 11–13)	**Inheritance and Polymorphism** ▪ Extending Classes 1, 2, 3, 4 ▪ Class Hierarchies 1, 2, 3, 4, 5, 6, 7 ▪ Polymorphism 1, 2, 3, 4, 5, 6 ▪ Overriding Methods 1, 2, 3, 4, 5, 6, 7 ▪ Test #15

		▪ Perform integration testing: IIIA3 **Standard Data Structures** ▪ Classes: IVC ▪ **AP Computer Science Lab Student Guide: Magpie**		
14	**Classes Revisited**	**Object-Oriented Program Design** ▪ Apply functional decomposition: IA6 **Program Implementation** ▪ Constant declarations: IIB2a ▪ Java library classes (Integer and Double): IIC ▪ **AP Computer Science Lab Student Guide: Picture**	▪ Calculate prime numbers ▪ Perform a frequency analysis on a passage of text ▪ Encode and decode a secret message using a Caesar shift ▪ AP Computer Science Lab: Picture ▪ Challenge Exam Modules 13–14	**OOP - Class Definitions Revisited** ▪ Multiple Constructors 1, 2 ▪ Overloaded Instance Methods ▪ `Integer` and `Double` ▪ `public` and `private` 1, 2 ▪ Class Methods 1, 2, 3 Class Variables and Constants 1, 2, 3 ▪ `final` Block Variables ▪ Object aliasing 1, 2, 3 ▪ Using `this` 1, 2, 3, 4 ▪ Test 16

15	**Abstraction and Interfaces**	**Program Implementation** ▪ Interface declarations: IIB2e ▪ Java library classes (List, Comparable): IIC **Standard Data Structures** ▪ Classes: IVC ▪ Lists: IVD **AP Computer Science Lab Student Guide: Picture**	▪ Create abstract classes for homework assignments in different subjects ▪ Implement an interface to process homework assignments in different subjects ▪ Implement Comparable <T> to process homework assignments in different subjects ▪ Use an interface and abstract classes to process an inventory of products ▪ AP Computer Science Lab: Picture ▪ Challenge Exam Module 15	**OOP - Abstractions** ▪ Abstract Classes 1, 2, 3 ▪ Interfaces 1, 2, 3, 4, 5 ▪ The `List <E>` Interface 1, 2 ▪ The `Comparable <T>` Interface 1, 2, 3, 4, 5, 6, 7, 8 ▪ Test 17 **Data Structures** ▪ ArrayLists
16	**Algorithms**	**Object-Oriented Program Design** ▪ Choose appropriate data representation and algorithms: IA5 **Standard Data Structures** ▪ Lists: IVD ▪ Arrays: IVE **Standard Algorithms** ▪ Traversals: VA1 ▪ Insertions: VA2 ▪ Deletions: VA3	▪ Traverse a set of election results, calculate the total votes, and print an updated report ▪ Traverse a set of election results, replace vote counts, calculate the new results, and print an updated report ▪ Traverse a set of election results, insert write-in candidates, calculate the new results, and print an updated report ▪ Traverse a set of election results, delete incorrect data, calculate the	**Algorithms** ▪ Algorithms 1, 2, 3 ▪ Traversals ▪ Replacements ▪ Insertions 1, 2, 3 ▪ Deletions

		AP Computer Science Lab Student Guide: Picture	new results, and print an updated report ▪ Process a set of student grades using traversal, insertion, replacement, and deletion methods ▪ AP Computer Science Lab: Picture ▪ Discussion-based assessment (Modules 14–16)	
17	**Sorting**	**Object-Oriented Program Design** ▪ Choose appropriate data representation and algorithms: IA5 **Program Implementation** ▪ Understand and evaluate recursive methods: IIB4e **Standard Algorithms** ▪ Selection Sort: VC1 ▪ Insertion Sort: VC2 ▪ Mergesort: VC3 ▪ **AP Computer Science Lab Student Guide: Elevens**	▪ Use an insertion sort to arrange a movie list in ascending or descending order by title, release year, or studio ▪ Use a selection sort to arrange a movie list in ascending or descending order by title, release year, or studio ▪ Use a merge sort to arrange a movie list in ascending or descending order by title, release year, or studio ▪ Use insertion, selection, and merge sorts to arrange products in an inventory by name, quantity, identification number, and price ▪ AP Computer Science Lab: Elevens	**Algorithms** ▪ Insertion Sort 1, 2, 3, 4, 5, 6 ▪ Selection Sort 1, 2, 3, 4, 5, 6, 7 ▪ Merge Sort 1, 2, 3, 4, 5, 6 ▪ Test 19

18	**Searching**	**Standard Algorithms** ▪ Sequential Search: VB1 ▪ Binary Search: VB2 ▪ **AP Computer Science Lab Student Guide: Elevens**	▪ Conduct a sequential search for specific titles, release year, or artist in a collection of music CDs ▪ Conduct a binary search for specific titles, release year, or artist in a collection of music CDs ▪ Use binary and sequential searches to locate specific individuals in a contact list by name, relationship, birth month, phone number, or email address ▪ AP Computer Science Lab: Elevens ▪ Challenge Exam Module 16, 17, and 18	**Searching** ▪ Sequential Search 1, 2 ▪ Binary Search 1, 2, 3, 4, 5 ▪ Test 18
19	**Program Analysis**	**Program Analysis** ▪ Identify boundary cases and generate appropriate test data: IIIA1 ▪ Perform integration testing: IIIA3 ▪ Understand runtime exceptions: IIIC ▪ Pre- and post-conditions: IIID1 ▪ Assertions: IIID2 ▪ Exact calculation of statement execution counts: IIIE1	▪ Handle exceptions appropriately when processing a set of student grades ▪ AP Computer Science Lab: Elevens	**Algorithms/Program Analysis** ▪ Introduction ▪ Assertions and Exceptions 1, 2, 3, 4

		▪ **AP Computer Science Lab Student Guide: Elevens**		
20	**AP Exam Review**		**Exam Review** ▪ Review for AP Computer Science A Exam ▪ Course Reflection ▪ Discussion-Based Assessment (Modules 17–20) ▪ Segment 2 exam	**IMACS Be Prepared for the AP Computer Science Exam** ▪ Exam Format, Grading, and Tips ▪ Java Features, Part 1 ▪ Java Features, Part 2 ▪ Program Design and OOP Concepts ▪ Algorithms ▪ Past Free Response Questions ▪ Practice Exams

APCS Scoring Component	Description	Modules within Course
[C1]	The course teaches students to design and implement computer-based solutions to problems	Modules 2–9, 12–19
[C2a]	The course teaches students to use and implement commonly-used algorithms.	5, 6, 8, 9, 12, 16, 17, 18
[C2b]	The course teaches students to use and implement commonly-used data structures.	6, 8, 9, 14, 15, 16, 17, AP Computer Science Labs (12-19)

[C3]	The course teaches students to select appropriate algorithms and data structures to solve problems.	12, 16, 17, 18
[C4]	The course teaches students to code fluently in an object-oriented paradigm using the programming language Java.	8, 13, 14, 15, AP Computer Science Labs (12-19)
[C5]	The course teaches students to use standard Java library classes from the AP Java subset delineated in Appendix A of the AP Computer Science A Course Description.	1–9, 12–19
[C6]	The course includes a structured lab component comprised of a minimum of 20 hours of hands-on lab experiences.	1–9, 12–19, AP Computer Science Labs 12-19
[C7]	The course teaches students to recognize the ethical and social implications of computer use.	5, 8, 9, 11, 14