CollegeBoard AP

COMPUTER SCIENCE A

Course Description

EFFECTIVE FALL 2014

AP Course Descriptions are updated regularly. Please visit AP Central® (apcentral.collegeboard.org) to determine whether a more recent Course Description PDF is available.

The College Board

The College Board is a mission-driven not-for-profit organization that connects students to college success and opportunity. Founded in 1900, the College Board was created to expand access to higher education. Today, the membership association is made up of more than 5,900 of the world's leading educational institutions and is dedicated to promoting excellence and equity in education. Each year, the College Board helps more than seven million students prepare for a successful transition to college through programs and services in college readiness and college success — including the SAT® and the Advanced Placement Program®. The organization also serves the education community through research and advocacy on behalf of students, educators, and schools.

For further information, visit **www.collegeboard.org**.

AP Equity and Access Policy

The College Board strongly encourages educators to make equitable access a guiding principle for their AP programs by giving all willing and academically prepared students the opportunity to participate in AP. We encourage the elimination of barriers that restrict access to AP for students from ethnic, racial, and socioeconomic groups that have been traditionally underserved. Schools should make every effort to ensure their AP classes reflect the diversity of their student population. The College Board also believes that all students should have access to academically challenging course work before they enroll in AP classes, which can prepare them for AP success. It is only through a commitment to equitable preparation and access that true equity and excellence can be achieved.

AP Course Descriptions

AP Course Descriptions are updated regularly. Please visit AP Central® (apcentral.collegeboard.org) to determine whether a more recent Course Description PDF is available.

© 2014 The College Board. College Board, ACCUPLACER, Advanced Placement Program, AP, AP Central, SAT, SpringBoard, and the acorn logo are registered trademarks of the College Board. PSAT/NMSQT is a registered trademark of the College Board and National Merit Scholarship Corporation. All other products and services may be trademarks of their respective owners. (Visit the College Board on the Web: www.collegeboard.org.)

Contents

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

About the AP® Program

AP® enables students to pursue college-level studies while still in high school. Through more than 30 courses, each culminating in a rigorous exam, AP provides willing and academically prepared students with the opportunity to earn college credit, advanced placement, or both. Taking AP courses also demonstrates to college admission officers that students have sought out the most rigorous course work available to them.

Each AP course is modeled upon a comparable college course, and college and university faculty play a vital role in ensuring that AP courses align with college-level standards. Talented and dedicated AP teachers help AP students in classrooms around the world develop and apply the content knowledge and skills they will need in college.

Each AP course concludes with a college-level assessment developed and scored by college and university faculty as well as experienced AP teachers. AP Exams are an essential part of the AP experience, enabling students to demonstrate their mastery of college-level course work. More than 90 percent of four-year colleges and universities in the United States grant students credit, placement, or both on the basis of successful AP Exam scores. Universities in more than 60 countries recognize AP Exam scores in the admission process and/or award credit and placement for qualifying scores. Visit www.collegeboard.org/ap/creditpolicy to view AP credit and placement policies at more than 1,000 colleges and universities.

Performing well on an AP Exam means more than just the successful completion of a course; it is a pathway to success in college. Research consistently shows that students who score a 3 or higher on AP Exams typically experience greater academic success in college and are more likely to graduate on time than otherwise comparable non-AP peers. Additional AP studies are available at www.collegeboard.org/apresearchsummaries.

Offering AP Courses and Enrolling Students

This course description details the essential information required to understand the objectives and expectations of an AP course. The AP Program unequivocally supports the principle that each school develops and implements its own curriculum that will enable students to develop the content knowledge and skills described here.

Schools wishing to offer AP courses must participate in the AP Course Audit, a process through which AP teachers' syllabi are reviewed by college faculty. The AP Course Audit was created at the request of College Board members who sought a means for the College Board to provide teachers and administrators with clear guidelines on curricular and resource requirements for AP courses and to help colleges and universities validate courses marked "AP" on students' transcripts. This process ensures that AP teachers' syllabi meet or exceed the curricular and resource expectations that college and secondary school faculty have established for college-level courses. For more information on the AP Course Audit, visit www.collegeboard.org/apcourseaudit.

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

How AP Courses and Exams Are Developed

AP courses and exams are designed by committees of college faculty and expert AP teachers who ensure that each AP subject reflects and assesses college-level expectations. AP Development Committees define the scope and expectations of the course, articulating through a curriculum framework what students should know and be able to do upon completion of the AP course. Their work is informed by data collected from a range of colleges and universities to ensure that AP coursework reflects current scholarship and advances in the discipline. To find a list of each subject's current AP Development Committee members, please visit apcentral .collegeboard.org/developmentcommittees.

The AP Development Committees are also responsible for drawing clear and well-articulated connections between the AP course and AP Exam — work that includes designing and approving exam specifications and exam questions. The AP Exam development process is a multi-year endeavor; all AP Exams undergo extensive review, revision, piloting, and analysis to ensure that questions are high quality and fair, and that there is an appropriate spread of difficulty across the questions.

Throughout AP course and exam development, the College Board gathers feedback from various stakeholders in both secondary schools and higher education institutions. This feedback is carefully considered to ensure that AP courses and exams are able to provide students with a college-level learning experience and the opportunity to demonstrate their qualifications for advanced placement upon college entrance.

How AP Exams Are Scored

The exam scoring process, like the course and exam development process, relies on the expertise of both AP teachers and college faculty. While multiple-choice questions are scored by machine, the free-response questions are scored by thousands of college faculty and expert AP teachers at the annual AP Reading. AP Exam Readers are thoroughly trained, and their work is monitored throughout the Reading for fairness and consistency. In each subject, a highly respected college faculty member fills the role of Chief Reader, who, with the help of AP Readers in leadership positions, maintains the accuracy of the scoring standards. Scores on the free-response questions are weighted and combined with the weighted results of the computer-scored multiple-choice questions. These composite, weighted raw scores are converted into the reported AP Exam scores of 5, 4, 3, 2, and 1.

The score-setting process is both precise and labor intensive, involving numerous psychometric analyses of the results of a specific AP Exam in a specific year and of the particular group of students who took that exam.

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

Additionally, to ensure alignment with college-level standards, part of the score-setting process involves comparing the performance of AP students with the performance of students enrolled in comparable courses in colleges throughout the United States. In general, the AP composite score points are set so that the lowest raw score needed to earn an AP Exam score of 5 is equivalent to the average score among college students earning grades of A in the college course. Similarly, AP Exam scores of 4 are equivalent to college grades of A–, B+, and B. AP Exam scores of 3 are equivalent to college grades of B–, C+, and C.

AP Score	Qualification
5	Extremely well qualified
4	Well qualified
3	Qualified
2	Possibly qualified
1	No recommendation

Additional Resources

Visit apcentral.collegeboard.org for more information about the AP Program.

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

AP Computer Science A Course Description

Important Revisions to This Course Description

- New *AP Computer Science A Labs* can be found on the AP Computer Science home page, http://apcentral.collegeboard.com/apc/public/courses/teachers_corner/222163.html
- Lab Requirements and the new *AP Computer Science Labs*.
- Sample Search and Sort algorithms (Appendix C)

INTRODUCTION

Computer science embraces problem solving, hardware, algorithms, and perspectives that help people utilize computers to address real-world problems in contemporary life. As the study of computer science is evolving, the careful design of the AP Computer Science A course and exam continues to strive to engage a diverse student population, including female and underrepresented students, with the rigorous and rewarding concepts of computer science. Students who take the AP Computer Science A course and exam are well prepared to continue their study of computer science and its integration into a wide array of computing and STEM-related fields.

The AP Computer Science A curriculum provides resources, such as application-related labs, that connect with students with diverse interests, particularly female and underrepresented student populations. The course is engaging and underscores the importance of communicating solutions appropriately and in ways that are relevant to current societal needs. Thus, a well-designed, modern AP Computer Science A course can help address traditional issues of equity, access, and broadening participation in computing while providing a strong and engaging introduction to fundamental areas of the discipline.

The AP Computer Science A course introduces students to computer science with fundamental topics that include problem solving, design strategies and methodologies, organization of data (data structures), approaches to processing data (algorithms), analysis of potential solutions, and the ethical and social implications of computing. The course emphasizes both object-oriented and imperative problem solving and design. These techniques represent proven approaches for developing solutions that can scale up from small, simple problems to large, complex problems. For a listing of the topics addressed, see the Computer Science A topic outline on pages 8–10.

The AP Computer Science A course curriculum is compatible with many CS1 courses in colleges and universities. Some colleges and universities may organize their curricula in alternative ways, so that the topics of the AP Computer Science A course are spread over several college courses, with other topics from computer science interspersed.

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

Colleges and universities offer a wide range of introductory computer science courses, so the outline of topics described here may not match any course exactly. The AP Computer Science A course is compatible with recommendations of the Association for Computing Machinery (ACM) and the Computer Society of the Institute of Electrical and Electronics Engineers (IEEE-CS) in several fundamental areas. ACM/IEEE-CS Knowledge Units represented in the AP Computer Science A course include fundamental data structures and algorithms, machine-level representation of data, object-oriented programming, basic type systems, algorithms and design, fundamental programming concepts, fundamental data structures, development methods, and social context. The AP Computer Science A course is compatible with the curriculum outlined in the Topics in Computer Science course of the Computer Science Teachers Association (CSTA).

The AP Computer Science A course can be offered by any secondary school that has faculty who possess the necessary expertise and have access to appropriate computing facilities. The course represents college-level achievement for which many colleges and universities grant advanced placement and credit. Placement and credit are granted by institutions in accordance with their own policies, not by the College Board or the AP Program.

THE COURSE

The AP Computer Science A course is an introductory course in computer science. The major theme of the course is problem solving. The topic outline on pages 8–10 summarizes the content required in the AP Computer Science A course.

Goals

The goals of the AP Computer Science A course are comparable to those in the introductory course for computer science majors offered in many college and university computer science departments. It is not expected that all students in the AP Computer Science A course will major in computer science at the university level. The AP Computer Science A course is intended to serve both as an introductory course for computer science majors and as a course for people who will major in other disciplines and want to be informed citizens in today's technological society.

The following goals apply to the AP Computer Science A course. Students should be able to:

- design, implement, and analyze solutions to problems.
- use and implement commonly used algorithms.
- use standard data structures.
- develop and select appropriate algorithms and data structures to solve new problems.
- write solutions fluently in an object-oriented paradigm.
- write, run, test, and debug solutions in the Java programming language, utilizing standard Java library classes and interfaces from the AP Java subset.

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

- read and understand programs consisting of several classes and interacting objects.
- read and understand a description of the design and development process leading to such a program. (Examples of such solutions can be found in the *AP Computer Science Labs.*)
- understand the ethical and social implications of computer use.

Computer Language

Because the discipline of computer science emphasizes problem solving, study of the discipline requires a mechanism to express potential solutions precisely and concisely. Since any natural language (e.g., English) allows inconsistencies and ambiguities, solutions in computer science require a communication medium more formal than a natural language. For this reason, the AP Computer Science A course requires that potential solutions of problems be written in the Java programming language. In addition to precision of expression, Java supports important elements of problem solving, including object-orientation, abstraction, and encapsulation. The use of Java also allows students to test potential solutions to problems by running programs.

Because the Java programming language is extensive with far more features than could be covered in a single introductory course, the AP Computer Science A Exam covers a subset of Java. The AP Java subset can be found in Appendix A.

Resources

Students should have access to a computer system that represents relatively recent technology. A school should ensure that each student has access to a computer for at least three hours a week; additional time is desirable. Student and instructor access to computers is important during class time, but additional time is essential for students to develop solutions to problems individually.

The computer system must allow students to create, edit, compile quickly, and execute Java programs comparable in size to those found in the *AP Computer Science Labs*. It is highly desirable that these computers provide student access to the Internet. It is essential that each computer science teacher has Internet access.

A school must ensure that each student has a college-level text for individual use inside and outside of the classroom. Schools are encouraged to provide copies of the lab materials for individual use.

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

Prerequisites

The assumed prerequisites for entering the AP Computer Science A course include knowledge of basic English and algebra. A student in the AP Computer Science A course should be comfortable with functions and the concepts found in the uses of function notation, such as $f(x) = x + 2$ and $f(x) = g(h(x))$. It is important that students and their advisers understand that any significant computer science course builds upon a foundation of mathematical reasoning that should be acquired before attempting such a course.

Teaching the Course

Teachers are expected to present a course comparable to that taught as the first course on the college level. The course emphasizes problem solving, procedural and data abstraction, object-oriented programming and design methodology, algorithms, and data structures. Teachers should be aware that the field of computer science is constantly evolving. Teachers should endeavor to keep current with changes in problem solving methodologies, the Java programming language, and pedagogy involving active learning. Some resources that may assist teachers in professional development are AP Computer Science workshops and summer institutes, the AP Teacher Community and AP Central. For more information on workshops, and released materials, teachers should visit the College Board's online home for AP Teachers (www.collegeboard.org).

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

TOPIC OUTLINE

Following is an outline of the major topics considered for the AP Computer Science A Exam. This outline is intended to define the scope of the course, but not 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, and can be adapted to changing circumstances. The design process needs to be based on a thorough understanding of the problem to be solved

A. Program and Class Design
   1. Problem analysis
   2. Data abstraction and encapsulation
   3. Class specifications, interface specifications, relationships ("is-a," "has-a"), and extension using inheritance
   4. Code reuse
   5. Data representation and algorithms
   6. Functional decomposition

II. Program Implementation

Part of the problem-solving process is the statement of solutions in a precise form that invites review and analysis. The implementation of solutions in the Java programming language reinforces concepts, allows potential solutions to be tested, and encourages discussion of solutions and alternatives.

A. Implementation techniques
   1. Top-down
   2. Bottom-up
   3. Object-oriented
   4. Encapsulation and information hiding
   5. Procedural abstraction

B. Programming constructs
   1. Primitive types vs. reference types
   2. Declaration
      a. Constants
      b. Variables
      c. Methods and parameters
      d. Classes
      e. Interfaces
   3. Text output using `System.out.print` and `System.out.println`
   4. Control
      a. Method call
      b. Sequential execution
      c. Conditional execution
      d. Iteration
      e. Recursion

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

5. Expression evaluation
    a. Numeric expressions
    b. String expressions
    c. Boolean expressions, short-circuit evaluation, De Morgan's law

C. Java library classes and interfaces included in the 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. Development of appropriate test cases, including boundary cases
    2. Unit testing
    3. Integration testing
B. Debugging
    1. Error categories: compile-time, run-time, logic
    2. Error identification and correction
    3. Techniques such as using a debugger, adding extra output statements, or hand-tracing code.
C. Runtime exceptions
D. Program correctness
    1. Pre- and post-conditions
    2. Assertions
E. Algorithm Analysis
    1. Statement execution counts
    2. Informal running time comparison
F. Numerical representations of integers
    1. Representations of non-negative integers in different bases
    2. Implications of finite integer bounds

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. Primitive data types (int, boolean, double)
B. Strings
C. Classes
D. Lists
E. Arrays (1-dimensional and 2-dimensional)

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

V. Standard Operations and 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 data structures
    1. Traversals
    2. Insertions
    3. Deletions
B. Searching
    1. Sequential
    2. Binary
C. Sorting
    1. Selection
    2. Insertion
    3. Mergesort

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

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

COMMENTARY ON THE TOPIC OUTLINE

The topic outline summarizes the content of the AP Computer Science A curriculum. In this section, we provide more details about the topics in the outline.

I. Object-Oriented Program Design

Computer science involves the study of complex systems, of which computer software is often a part. To understand the development of computer software, we need tools that can make sense of that complexity. Object-oriented design and programming form an approach that enables us to do that, based on the idea that a piece of software, just like a computer itself, is composed of many interacting parts.

The term *design*, as used here, refers to the design of implementations that meet particular specifications, as opposed to the design of specifications that meet particular requirements or of the requirements themselves. Students need not start by designing a whole program but may study programs already developed, then write or modify parts of a program to add to or change its functionality. By the end of the course students will be able to work from a specification to develop a design for a program or part of a program.

In an object-oriented approach, the fundamental unit of an executing program is an object, an entity that has state (data) and behavior (operations that access or change its state and that may interact with other objects through well defined interfaces). The AP Computer Science A curriculum uses the common class-based approach to object-oriented programming, in which objects are defined by classes: a class specifies the components and operations of an object, and each object is an instance of a class.

A. Program and Class Design

A fundamental part of the development of an object-oriented program is the design of its classes. Students should understand inheritance ("is-a") relationships and composition ("has-a") relationships among the different classes that comprise a program. They should also be able to implement a class inheritance hierarchy when given the specifications for the classes involved — which classes are subclasses of other classes.

Students should be able to design a class — write the class declaration including the instance variables and the method signatures (whose bodies would comprise the implementation of this design) — when they are given a description of the type of entity the class represents. Such a description would include the information that must be encapsulated by the class and the operations that can be applied to an instance of the class to access or modify that information. The design of a class includes decisions on appropriate data structures for representing its information and on algorithms for operations on that data. The decomposition of operations into subsidiary operations — functional decomposition — is part of the design process.

Given a design for a class, either their own or one provided, students should then be able to implement the class. They should also be able to extend a given class using inheritance, thereby creating a subclass with modified or additional functionality.

An *interface* is a specification for a set of operations that a class must implement. In Java, there is a specific construct, the interface, which can be used for this purpose, so that the specification of the methods applicable to a class can be separated from implementations of that specification. Students should be able to write interfaces, write classes that implement interfaces and write programs that use interfaces.

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

B. Design as an Exam Topic

The AP Computer Science A Exam may include questions that ask about the design as well as the implementation of classes or a simple hierarchy of classes.

A design question provides students with a description of the type of information that an object should encapsulate and of operations on that information that an object should provide. Students are required to provide part or all of an interface or class declaration to define such objects.

A design question may require a student to develop a solution that requires the following:

- appropriate use of inheritance from another class using the keyword `extends`
- appropriate implementation of an interface using the keyword `implements`
- declaration of constructors with appropriate parameters
- declaration of constructors and methods with
  - meaningful names
  - appropriate parameters
  - appropriate return types
- appropriate choice of data representation
- appropriate designation of methods as client-accessible (public) or internal (private)
- designation of all instance variables as private

A design question might only require that a student specify the appropriate constructor and method headers (access modifier, return type, method identifier, and parameter list) and not require that the body of the constructors or methods be implemented. A question focusing on a simple class hierarchy might also require implementation of the bodies of some or all of the classes' constructors and methods.

The AP Java Subset is restricted to private instance variables to encourage the exclusive use of methods to access and modify objects, a practice that minimizes dependence on particular choices of representation and thus makes such choices easy to change.

II. Program Implementation

There are topics not included in the course outline that will be part of any introductory course. For example, input and output must be part of a course on computer programming. However, in a modern object-oriented approach to programming, there are many ways to handle input and output. Consequently, the AP Computer Science A course does not prescribe any particular approach and will not test the details of input and output (except for basic text output using `System.out.print` and `System.out.println`), so that teachers may use an approach that fits their own style and whatever textbook and other materials they use.

Students are expected to demonstrate an understanding of the concept of recursion and to determine the results of executing simple recursive methods. Unlike the other control structures, students will not have to write recursive methods.

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

JAVA LIBRARY CLASSES AND INTERFACES

An important aspect of modern programming is the existence of extensive libraries that supply many common classes and methods. An essential programming skill is the ability to appropriately use available libraries. The AP Computer Science A curriculum specifies a subset of classes and interfaces from the Java libraries with which students should be familiar.

In addition, students should recognize possibilities for reusing components of their own code or of other examples of code, such as the *AP Computer Science Labs*, in different programs.

III. Program Analysis

There are many techniques for finding and correcting errors, for "debugging" a program or segment of a program. These include hand-tracing code, adding extra output statements to trace the execution of a program, or using a debugger to provide information about the program as it runs and when it crashes. Students should be encouraged to experiment with available debugging facilities. However, these will not be tested since they vary from system to system.

Students should be able to read and modify code for a program. They should also be able to extend existing code by taking a given class declaration and declaring a new class using inheritance to add or change the given class' functionality. The *AP Computer Science Labs* contain examples of using inheritance to create new classes.

Students in the AP Computer Science A course should understand runtime exceptions as they are used by the Java virtual machine to report program errors. The `throw` statement is not part of the subset but might be useful for teaching students about error checking.

Students also need to be familiar with the concepts of preconditions, postconditions, and assertions and correctly interpret them when presented as pseudocode. Teachers may find the `assert` statement of the Java language to be a convenient way to have students test the satisfaction of any of these, but `assert` is not part of the AP Java Subset and is not tested.

Students should be able to make informal comparisons of running times of different pieces of code: for example, by determining the number of times a statement is executed or the number of loop iterations needed for a computation.

Many programs involve numerical computations and therefore are limited by the finite representations of integers in a computer. Students should understand the representation of non-negative integers in different bases, particularly decimal, binary, hexadecimal, and octal. They should also understand the consequences of the finite bounds of integer representation.

IV. Standard Data Structures

Students should understand the standard data structures and their use. Students need to be able to use the primitive types: `int`, `double`, and `boolean`. The other primitive types in Java are not part of the AP Java subset but may be useful in the AP Computer Science A course.

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

Students are responsible for understanding the Java `String` class and the methods of the `String` class that are listed in the Java Quick Reference (see Appendix B).

Students should be comfortable working with 1-dimensional and 2-dimensional arrays. They need to understand that 2-dimensional arrays are stored as arrays of arrays. For the purposes of the AP Computer Science A Exam, students should assume that 2-dimensional arrays are rectangular (not ragged) and the elements are indexed in row-major order.

Students should be able to use lists of data and be able to use Java arrays and the `ArrayList` class to implement such lists. They should be able to use either of these list types in a program and should be able to select the most appropriate one for a given application. The methods for the `List` interface (and its implementation by the `ArrayList` class) for which students are responsible are specified in the Java Quick Reference (see Appendix B).

V. Standard Operations and Algorithms

Standard operations for arrays include traversals, insertions, and deletions. Students should know the two standard searches, sequential search and binary search, and the relative running time of each. There are three sorting algorithms that are required for the AP Computer Science A course: selection, insertion, and merge sort. Standard implementations of these algorithms can be found in Appendix C. Questions based on these implementations may appear on the exam.

VI. Computing in Context

Given the tremendous impact computers and computing have on almost every aspect of society, it is important that intelligent and responsible attitudes about the use of computers be developed as early as possible. The applications of computing that are studied in the AP Computer Science A course provide opportunities to discuss the impact of computing. Typical issues include the:

- impact of applications using databases, particularly over the Internet, on an individual's right to privacy;
- economic and legal impact of viruses and other malicious attacks on computer systems;
- need for fault-tolerant and highly reliable systems for life-critical applications and the resulting need for software engineering standards; and
- intellectual property rights and fair use of intellectual property.

Attitudes are acquired, not taught. References to responsible use of computer systems should be integrated throughout the AP Computer Science A course. Participation in the AP Computer Science A course provides an opportunity to discuss issues such as the responsible use of a system and respect for the rights and property of others. Students should learn to take responsibility for the programs they write and for the consequences of the use of their programs.

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

LAB REQUIREMENTS

Although the AP Computer Science A course draws heavily upon theory, formal logic, abstract data structures, and a conceptual understanding of algorithms, students also must gain significant experience applying the concepts to tackle a wide range of problems. As students design data structures and develop algorithms, the students should integrate ideas, test hypotheses, and explore alternative approaches. Further, activities motivated by real-world applications can provide insights about how computing can be useful in society, motivate the study of technical issues, and capture students' interest.

The AP Computer Science A course must include a minimum of 20 hours of hands-on structured-lab experiences to engage students in individual or group problem solving. Thus, each AP Computer Science A course must include a substantial laboratory component in which students design solutions to problems, express their solutions precisely (i.e., in the Java programming language), test their solutions, identify and correct errors (when mistakes occur), and compare possible solutions. Collectively, these laboratory experiences and activities should contain the following characteristics:

- Explore computing in context at a significant level, building upon existing code that provides examples of good style and appropriate use of programming language constructs.
- Contain a significant problem-solving component in which students study alternative approaches for solving a problem, solve new problems, or modify existing code to solve altered problems.
- Provide students with experience working with programs involving multiple interactive classes and may involve decomposing a program into classes and using inheritance, interfaces, and other object-oriented concepts as identified in the AP Computer Science A topic outline.

Three exemplar labs, the AP Computer Science A Labs, have been developed for teachers to use in the AP Computer Science A course. AP Computer Science A teachers will be able to access all instructional resources for each lab (Teacher and Student Guides, solutions and code files) through their AP Course Audit accounts. The first lab (Magpie) can be incorporated early in the course and involves simple string processing and conditional execution. The second lab (Picture Lab) involves 2-dimensional array manipulation in the context of image processing. The third lab (Elevens) provides an example of larger object-oriented program design. The AP Computer Science Labs include teacher and student guides along with Java code. Each teacher guide includes concepts covered, learning objectives, necessary prerequisite knowledge, guidelines on when each lab might fit naturally into a course, suggestions on the use of the materials, suggested problems and questions for use during each activity, and sample assessment exercises. The Student Guides for the *AP Computer Science A Labs* can be found on the AP Computer Science home page:

http://apcentral.collegeboard.com/apc/public/courses/teachers_corner/222163.html

Teachers may use the provided labs, develop their own labs and/or utilize laboratory exercises from textbook authors or other sources. When choosing labs, teachers must carefully evaluate the activities, objectives, and materials to be certain that the labs address the characteristics outlined above.

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

AP Computer Science A: Curricular Requirements

- The teacher has read the most recent AP Computer Science A Course Description.
- The course teaches students to design and implement computer-based solutions to problems.
- The course teaches students to use and implement commonly used algorithms and data structures.
- The course teaches students to select appropriate algorithms and data structures to solve problems.
- The course teaches students to code fluently in an object-oriented paradigm using the programming language Java.
- 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.
- The course includes a structured lab component comprised of a minimum of 20 hours of hands-on lab experiences.
- The course teaches students to recognize the ethical and social implications of computer use.

Resource Requirements

- The school ensures that each student has a college-level text for individual use inside and outside of the classroom and has access to the AP Computer Science A Labs.
- The school ensures that each student has access to a computer for at least three hours a week; three hours are the bare minimum, additional time is desirable. The computer system must contain appropriate software to create and edit programs and must allow programs comparable in size to the current AP Computer Science A Labs to compile in seconds. Internet access is strongly encouraged.

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

THE EXAM

The AP Computer Science A Exam is 3 hours long and seeks to determine how well students have mastered the concepts and techniques contained in the course outline. The exam consists of two sections: a multiple-choice section (40 questions in 1 hour and 15 minutes), which tests proficiency in a wide variety of topics, and a free-response section (4 questions in 1 hour and 45 minutes), which requires the student to demonstrate the ability to solve problems involving more extended reasoning.

The multiple-choice and the free-response sections of the AP Computer Science A Exam require students to demonstrate their ability to solve problems, including their ability to design, write, and analyze programs and subprograms. Minor points of syntax are not tested on the exam. All code given is consistent with the AP Java subset. All student responses involving code must be written in Java. Students are expected to be familiar with and able to use the standard Java classes and interfaces listed in the AP Java subset. For both the multiple-choice and the free-response sections of the exam, a quick reference to the classes and interfaces in the AP Java subset will be provided. The Java Quick Reference is included in Appendix B.

In the determination of the grade for the exam, the multiple-choice section and the free-response section are given equal weight. Because the exam is designed for full coverage of the subject matter, it is not expected that many students will be able to correctly answer all the questions in either the multiple-choice section or the free-response section in the time allotted.

Multiple-choice questions on the exam are classified according to the type of content that is tested in the question. Questions may be listed in one or more of the classification categories. For example, a question that uses a looping construct to traverse the elements of an array would be listed under both the Data Structures and the Programming Fundamentals categories. The table below shows the classification categories and how they are represented in the multiple-choice section of the exam. Because questions can be classified in more than one category, the total of the percentages is greater than 100%.

Classification Category	Percent of multiple-choice items
Programming Fundamentals	55–75%
Data Structures	20–40%
Logic	5–15%
Algorithms/Problem Solving	25–45%
Object-Oriented Programming	15–25%
Recursion	5–15%
Software Engineering	2–10%

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

Computer Science A: Sample Multiple-Choice Questions

Following is a representative set of questions. The answer key for the Computer Science A multiple-choice questions is on page 43. Multiple-choice scores are based on the number of questions answered correctly. Points are not deducted for incorrect answers, and no points are awarded for unanswered questions. Because points are not deducted for incorrect answers, students are encouraged to answer all multiple-choice questions. Students should eliminate as many choices as they can on any questions for which they do not know the answer, and then select the best answer among the remaining choices.

Directions: Determine the answer to each of the following questions or incomplete statements, using the available space for any necessary scratch work. Then decide which is the best of the choices given and fill in the corresponding circle on the answer sheet. No credit will be given for anything written in the examination booklet. Do not spend too much time on any one problem

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate.
- Assume that declarations of variables and methods appear within the context of an enclosing class.
- Assume that method calls that are not prefixed with an object or class name and are not shown within a complete class definition appear within the context of an enclosing class.
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

1. Consider the following code segment.

```
for (int k = 0; k < 20; k = k + 2)
{
  if (k % 3 == 1)
  {
    System.out.print(k + "  ");
  }
}
```

What is printed as a result of executing the code segment?

(A) `4  16`

(B) `4  10  16`

(C) `0  6  12  18`

(D) `1  4  7  10  13  16  19`

(E) `0  2  4  6  8  10  12  14  16  18`

2. Consider the following code segment.

```
List<String> animals = new ArrayList<String>();

animals.add("dog");
animals.add("cat");
animals.add("snake");
animals.set(2, "lizard");
animals.add(1, "fish");
animals.remove(3);
System.out.println(animals);
```

What is printed as a result of executing the code segment?

(A) `[dog, fish, cat]`

(B) `[dog, fish, lizard]`

(C) `[dog, lizard, fish]`

(D) `[fish, dog, cat]`

(E) The code throws an `ArrayIndexOutOfBoundsException` exception.

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

3. Consider the following method.

```
public static void mystery(List<Integer> nums)
{
  for (int k = 0; k < nums.size(); k++)
  {
    if (nums.get(k).intValue() == 0)
    {
       nums.remove(k);
    }
  }
}
```

Assume that a `List<Integer> values` initially contains the following `Integer` values.

```
[0, 0, 4, 2, 5, 0, 3, 0]
```

What will `values` contain as a result of executing `mystery(values)`?

(A) `[0, 0, 4, 2, 5, 0, 3, 0]`

(B) `[4, 2, 5, 3]`

(C) `[0, 0, 0, 0, 4, 2, 5, 3]`

(D) `[0, 4, 2, 5, 3]`

(E) The code throws an `ArrayIndexOutOfBoundsException` exception.

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

4. At a certain high school students receive letter grades based on the following scale.

Integer Score	Letter Grade
93 or above	A
From 84 to 92 inclusive	B
From 75 to 83 inclusive	C
Below 75	F

Which of the following code segments will assign the correct string to `grade` for a given integer `score` ?

I.
```
if (score >= 93)
  grade = "A";
if (score >= 84 && score <= 92)
  grade = "B";
if (score >= 75 && score <= 83)
  grade = "C";
if (score < 75)
  grade = "F";
```

II.
```
if (score >= 93)
  grade = "A";
if (84 <= score <= 92)
  grade = "B";
if (75 <= score <= 83)
  grade = "C";
if (score < 75)
  grade = "F";
```

III.
```
if (score >= 93)
  grade = "A";
else if (score >= 84)
  grade = "B";
else if (score >= 75)
  grade = "C";
else
  grade = "F";
```

(A) II only

(B) III only

(C) I and II only

(D) I and III only

(E) I, II, and III

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

5. Consider the following output.

```
1  1  1  1  1
2  2  2  2
3  3  3
4  4
5
```

Which of the following code segments will produce this output?

(A)
```
for (int j = 1; j <= 5; j++)
{
  for (int k = 1; k <= 5; k++)
  {
    System.out.print(j + "  ");
  }
  System.out.println();
}
```

(B)
```
for (int j = 1; j <= 5; j++)
{
  for (int k = 1; k <= j; k++)
  {
    System.out.print(j + "  ");
  }
  System.out.println();
}
```

(C)
```
for (int j = 1; j <= 5; j++)
{
  for (int k = 5; k >= 1; k--)
  {
    System.out.print(j + "  ");
  }
  System.out.println();
}
```

(D)
```
for (int j = 1; j <= 5; j++)
{
  for (int k = 5; k >= j; k--)
  {
    System.out.print(j + "  ");
  }
  System.out.println();
}
```

(E)
```
for (int j = 1; j <= 5; j++)
{
  for (int k = j; k <= 5; k++)
  {
    System.out.print(k + "  ");
  }
  System.out.println();
}
```


© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

6. A car dealership needs a program to store information about the cars for sale. For each car, they want to keep track of the following information: number of doors (2 or 4), whether the car has air conditioning, and its average number of miles per gallon. Which of the following is the best object-oriented program design?

   (A) Use one class, `Car`, with three instance variables: `int numDoors`, `boolean hasAir`, and `double milesPerGallon`.

   (B) Use four unrelated classes: `Car`, `Doors`, `AirConditioning`, and `MilesPerGallon`.

   (C) Use a class `Car` with three subclasses: `Doors`, `AirConditioning`, and `MilesPerGallon`.

   (D) Use a class `Car`, with a subclass `Doors`, with a subclass `AirConditioning`, with a subclass `MilesPerGallon`.

   (E) Use three classes: `Doors`, `AirConditioning`, and `MilesPerGallon`, each with a subclass `Car`.

7. Consider the following declarations.

```
public interface Shape
{
  int isLargerThan(Shape other);
  // Other methods not shown
}
public class Circle implements Shape
{
  // Other methods not shown
}
```

   Which of the following method headings of `isLargerThan` can be added to the declaration of the `Circle` class so that it will satisfy the `Shape` interface?

   I. `public int isLargerThan(Shape other)`

   II. `public int isLargerThan(Circle other)`

   III. `public boolean isLargerThan(Object other)`

   (A) I only

   (B) II only

   (C) III only

   (D) I and II only

   (E) I, II, and III

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

Questions 8–9 refer to the following incomplete class declaration.

```
public class TimeRecord
{
  private int hours;
  private int minutes; // 0 ≤ minutes < 60
  /** Constructs a TimeRecord object.
   *  @param h the number of hours
   *           Precondition: h ≥ 0
   *  @param m the number of minutes
   *           Precondition: 0 ≤ m < 60
   */
  public TimeRecord(int h, int m)
  {
    hours = h;
    minutes = m;
  }

  /** @return the number of hours
   */
  public int getHours()
  { /* implementation not shown */ }

  /** @return the number of minutes
   *  Postcondition: 0 ≤ minutes < 60
   */
  public int getMinutes()
  { /* implementation not shown */ }

  /** Adds h hours and m minutes to this TimeRecord.
   *  @param h the number of hours
   *           Precondition: h ≥ 0
   *  @param m the number of minutes
   *           Precondition: m ≥ 0
   */
  public void advance(int h, int m)
  {
    hours = hours + h;
    minutes = minutes + m;
    /* missing code */
  }
  // Other methods not shown
}
```


© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

8. Which of the following can be used to replace `/* missing code */` so that `advance` will correctly update the time?

   (A)
   ```
   minutes = minutes % 60;
   ```

   (B)
   ```
   minutes = minutes + hours % 60;
   ```

   (C)
   ```
   hours = hours + minutes / 60;
   minutes = minutes % 60;
   ```

   (D)
   ```
   hours = hours + minutes % 60;
   minutes = minutes / 60;
   ```

   (E)
   ```
   hours = hours + minutes / 60;
   ```

9. Consider the following declaration that appears in a class other than `TimeRecord`.

   ```
   TimeRecord[] timeCards = new TimeRecord[100];
   ```

   Assume that `timeCards` has been initialized with `TimeRecord` objects. Consider the following code segment that is intended to compute the total of all the times stored in `timeCards`.

   ```
   TimeRecord total = new TimeRecord(0,0);
   for (int k = 0; k < timeCards.length; k++)
   {
     /* missing expression */ ;
   }
   ```

   Which of the following can be used to replace `/* missing expression */` so that the code segment will work as intended?

   (A)
   ```
   timeCards[k].advance()
   ```

   (B)
   ```
   total += timeCards[k].advance()
   ```

   (C)
   ```
   total.advance(timeCards[k].hours,
                 timeCards[k].minutes)
   ```

   (D)
   ```
   total.advance(timeCards[k].getHours(),
                 timeCards[k].getMinutes())
   ```

   (E)
   ```
   timeCards[k].advance(timeCards[k].getHours(),
                        timeCards[k].getMinutes())
   ```

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

10. Consider the following instance variable and method.

```
private int[] arr;

/** Precondition: arr contains no duplicates;
 *                the elements in arr are in ascending order.
 *  @param low an int value such that 0 ≤ low ≤ arr.length
 *  @param high an int value such that low - 1 ≤ high < arr.length
 *  @param num an int value
 */
public int mystery(int low, int high, int num)
{
  int mid = (low + high) / 2;
  if (low > high)
  {
    return low;
  }
  else if (arr[mid] < num)
  {
    return mystery(mid + 1, high, num);
  }
  else if (arr[mid] > num)
  {
    return mystery(low, mid − 1, num);
  }
  else // arr[mid] == num
  {
    return mid;
  }
}
```

What is returned by the call `mystery(0, arr.length − 1, num)`?

(A) The number of elements in `arr` that are less than `num`

(B) The number of elements in `arr` that are less than or equal to `num`

(C) The number of elements in `arr` that are equal to `num`

(D) The number of elements in `arr` that are greater than `num`

(E) The index of the middle element in `arr`

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

Questions 11–12 refer to the following information.

Consider the following instance variable `nums` and method `findLongest` with line numbers added for reference. Method `findLongest` is intended to find the longest consecutive block of the value `target` occurring in the array `nums`; however, `findLongest` does not work as intended.

For example, if the array `nums` contains the values [7, 10, 10, 15, 15, 15, 15, 10, 10, 10, 15, 10, 10], the call `findLongest(10)` should return 3, the length of the longest consecutive block of 10s.

```
private int[] nums;

public int findLongest(int target)
{
    int lenCount = 0;
    int maxLen = 0;

Line 1:  for (int val : nums)
Line 2:  {
Line 3:    if (val == target)
Line 4:    {
Line 5:      lenCount++;
Line 6:    }
Line 7:    else
Line 8:    {
Line 9:      if (lenCount > maxLen)
Line 10:     {
Line 11:       maxLen = lenCount;
Line 12:     }
Line 13:   }
Line 14: }
Line 15: if (lenCount > maxLen)
Line 16: {
Line 17:   maxLen = lenCount;
Line 18: }
Line 19: return maxLen;
    }
```

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

11. The method `findLongest` does not work as intended. Which of the following best describes the value returned by a call to `findLongest ?`

    (A) It is the length of the shortest consecutive block of the value `target` in `nums`.

    (B) It is the length of the array `nums`.

    (C) It is the number of occurrences of the value `target` in `nums`.

    (D) It is the length of the first consecutive block of the value `target` in `nums`.

    (E) It is the length of the last consecutive block of the value `target` in `nums`.

12. Which of the following changes should be made so that method `findLongest` will work as intended?

    (A) Insert the statement `lenCount = 0;` between lines 2 and 3.

    (B) Insert the statement `lenCount = 0;` between lines 8 and 9.

    (C) Insert the statement `lenCount = 0;` between lines 10 and 11.

    (D) Insert the statement `lenCount = 0;` between lines 11 and 12.

    (E) Insert the statement `lenCount = 0;` between lines 12 and 13.

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

13. Consider the following instance variable and method.

```
private int[] numbers;

/** Precondition: numbers contains int values in no particular order.
 */
public int mystery(int num)
{
  for (int k = numbers.length - 1; k >= 0; k--)
  {
    if (numbers[k] < num)
    {
      return k;
    }
  }
  return -1;
}
```

Which of the following best describes the contents of `numbers` after the following statement has been executed?

```
int m = mystery(n);
```

(A) All values in positions `0` through `m` are less than `n`.

(B) All values in positions `m+1` through `numbers.length-1` are less than `n`.

(C) All values in positions `m+1` through `numbers.length-1` are greater than or equal to `n`.

(D) The smallest value is at position `m`.

(E) The largest value that is smaller than `n` is at position `m`.

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

14. Consider the following method.

```
/** @param x an int value such that x >= 0
 */
public void mystery(int x)
{
  System.out.print(x % 10);
  if ((x / 10) != 0)
  {
    mystery(x / 10);
  }
  System.out.print(x % 10);
}
```

Which of the following is printed as a result of the call `mystery(1234)`?

(A) `1234`

(B) `4321`

(C) `12344321`

(D) `43211234`

(E) Many digits are printed due to infinite recursion.

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

15. Consider the following two classes.

```
public class Dog
{
  public void act()
  {
    System.out.print("run ");
    eat();
  }
  public void eat()
  {
    System.out.print("eat ");
  }
}
public class UnderDog extends Dog
{
  public void act()
  {
    super.act();
    System.out.print("sleep ");
  }
  public void eat()
  {
    super.eat();
    System.out.print("bark ");
  }
 }
```

Assume that the following declaration appears in a class other than `Dog`.

```
Dog fido = new UnderDog();
```

What is printed as a result of the call `fido.act()` ?

(A) `run eat`

(B) `run eat sleep`

(C) `run eat sleep bark`

(D) `run eat bark sleep`

(E) Nothing is printed due to infinite recursion.

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

16. Consider the following recursive method.

```
public static int mystery(int n)
{
  if (n <= 1)
  {
    return 0;
  }
  else
  {
    return 1 + mystery(n / 2);
  }
}
```

Assuming that k is a nonnegative integer and $m = 2^k$, what value is returned as a result of the call `mystery(`m`)` ?

(A) 0

(B) k

(C) m

(D) $\frac{m}{2} + 1$

(E) $\frac{k}{2} + 1$

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

17. Consider the following instance variable and method.

```
private int[] array;

/** Precondition: array.length > 0
 */
public int checkArray()
{
  int loc = array.length / 2;
  for (int k = 0; k < array.length; k++)
  {
    if (array[k] > array[loc])
    {
      loc = k;
    }
  }
  return loc;
}
```

Which of the following is the best postcondition for `checkArray` ?

(A) Returns the index of the first element in array `array` whose value is greater than `array[loc]`

(B) Returns the index of the last element in array `array` whose value is greater than `array[loc]`

(C) Returns the largest value in array `array`

(D) Returns the index of the largest value in array `array`

(E) Returns the index of the largest value in the second half of array `array`

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

18. Consider the following methods.

```
public void changer(String x, int y)
{
  x = x + "peace";
  y = y * 2;
}

public void test()
{
  String s = "world";
  int n = 6;
  changer(s, n);

  /* End of method */
}
```

When the call `test()` is executed, what are the values of `s` and `n` at the point indicated by `/* End of method */` ?

	s	n
(A)	world	6
(B)	worldpeace	6
(C)	world	12
(D)	worldpeace	12
(E)	peace	12

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

19. Consider the following code segment.

```
int[][] mat = new int[3][4];
for (int row = 0; row < mat.length; row++)
{
   for (int col = 0; col < mat[0].length; col++)
   {
     if (row < col)
     {
       mat[row][col] = 1;
     }
     else if (row == col)
     {
       mat[row][col] = 2;
     }
     else
     {
       mat[row][col] = 3;
     }
   }
}
```

What are the contents of `mat` after the code segment has been executed?

(A)
```
{{2, 1, 1},
 {3, 2, 1},
 {3, 3, 2},
 {3, 3, 3}}
```

(B)
```
{{2, 3, 3},
 {1, 2, 3},
 {1, 1, 2},
 {1, 1, 1}}
```

(C)
```
{{2, 3, 3, 3},
 {1, 2, 3, 3},
 {1, 1, 2, 3}}
```

(D)
```
{{2, 1, 1, 1},
 {3, 2, 1, 1},
 {3, 3, 2, 1}}
```

(E)
```
{{1, 1, 1, 1},
 {2, 2, 2, 2},
 {3, 3, 3, 3}}
```

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

20. Consider the following method.

```
/** Precondition: arr contains only positive values.
 */
public static void doSome(int[] arr, int lim)
{
  int v = 0;
  int k = 0;
  while (k < arr.length && arr[k] < lim)
  {
    if (arr[k] > v)
    {
      v = arr[k]; /* Statement S */
    }
    k++; /* Statement T */
  }
}
```

Assume that `doSome` is called and executes without error. Which of the following are possible combinations for the value of `lim`, the number of times *Statement S* is executed, and the number of times *Statement T* is executed?

	Value of `lim`	Executions of *Statement S*	Executions of *Statement T*
I.	5	0	5
II.	7	4	9
III.	3	5	2

(A) I only

(B) II only

(C) III only

(D) I and III only

(E) II and III only

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

21. Consider the following instance variable, `arr`, and incomplete method, `partialSum`. The method is intended to return an integer array `sum` such that for all `k`, `sum[k]` is equal to `arr[0] + arr[1] + ... + arr[k]`. For instance, if `arr` contains the values `{ 1, 4, 1, 3 }`, the array `sum` will contain the values `{ 1, 5, 6, 9 }`.

```
private int[] arr;
public int[] partialSum()
{
  int[] sum = new int[arr.length];
  for (int j = 0; j < sum.length; j++)
  {
    sum[j] = 0;
  }
  /* missing code */
  return sum;
}
```

The following two implementations of `/* missing code */` are proposed so that `partialSum` will work as intended.

Implementation 1

```
for (int j = 0; j < arr.length; j++)
{
  sum[j] = sum[j - 1] + arr[j];
}
```

Implementation 2

```
for (int j = 0; j < arr.length; j++)
{
  for (int k = 0; k <= j; k++)
  {
    sum[j] = sum[j] + arr[k];
  }
}
```

Which of the following statements is true?

(A) Both implementations work as intended, but implementation 1 is faster than implementation 2.

(B) Both implementations work as intended, but implementation 2 is faster than implementation 1.

(C) Both implementations work as intended and are equally fast.

(D) Implementation 1 does not work as intended, because it will cause an `ArrayIndexOutOfBoundsException`.

(E) Implementation 2 does not work as intended, because it will cause an `ArrayIndexOutOfBoundsException`.

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

22. Consider the following declaration for a class that will be used to represent points in the *xy*-coordinate plane.

```
public class Point
{
  private int x;      // x-coordinate of the point
  private int y;      // y-coordinate of the point

  public Point()
  {
    x = 0;
    y = 0;
  }

  public Point(int a, int b)
  {
    x = a;
    y = b;
  }

  // Other methods not shown
}
```

The following incomplete class declaration is intended to extend the above class so that points can be named.

```
public class NamedPoint extends Point
{
  private String name; // name of point

  // Constructors go here

  // Other methods not shown
}
```


© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

Consider the following proposed constructors for this class.

I.
```
public NamedPoint()
{
  name = "";
}
```

II.
```
public NamedPoint(int d1, int d2, String pointName)
{
  x = d1;
  y = d2;
  name = pointName;
}
```

III.
```
public NamedPoint(int d1, int d2, String pointName)
{
  super(d1, d2);
  name = pointName;
}
```

Which of these constructors would be legal for the `NamedPoint` class?

(A) I only

(B) II only

(C) III only

(D) I and III only

(E) II and III only

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

23. Consider a `shuffle` method that is intended to return a new array that contains all the elements from `nums`, but in a different order. Let `n` be the number of elements in `nums`. The `shuffle` method should alternate the elements from `nums[0]` … `nums[n / 2 - 1]` with the elements from `nums[n / 2]` … `nums[n - 1]`, as illustrated in the following examples.

Example 1

	0	1	2	3	4	5	6	7
`nums`	10	20	30	40	50	60	70	80

	0	1	2	3	4	5	6	7
`result`	10	50	20	60	30	70	40	80

Example 2

	0	1	2	3	4	5	6
`nums`	10	20	30	40	50	60	70

	0	1	2	3	4	5	6
`result`	10	40	20	50	30	60	70

The following implementation of the `shuffle` method does not work as intended.

```
public static int[] shuffle(int[] nums)
{
  int n = nums.length;
  int[] result = new int[n];

  for (int j = 0; j < n / 2; j++)
  {
    result[j * 2] = nums[j];
    result[j * 2 + 1] = nums[j + n / 2];
  }

  return result;
}
```

Which of the following best describes the problem with the given implementation of the `shuffle` method?

(A) Executing `shuffle` may cause an `ArrayIndexOutOfBoundsException`.

(B) The first element of the returned array (`result[0]`) may not have the correct value.

(C) The last element of the returned array (`result[result.length - 1]`) may not have the correct value.

(D) One or more of `nums[0]` … `nums[nums.length / 2 - 1]` may have been copied to the wrong position(s) in the returned array.

(E) One or more of `nums[nums.length / 2]` … `nums[nums.length - 1]` may have been copied to the wrong position(s) in the returned array.

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

24. Consider the following `Util` class, which contains two methods. The completed `sum1D` method returns the sum of all the elements of the 1-dimensional array `a`. The incomplete `sum2D` method is intended to return the sum of all the elements of the 2-dimensional array `m`.

```
public class Util
{
  /** Returns the sum of the elements of the 1-dimensional array a */
  public static int sum1D(int[] a)
  {  /* implementation not shown */  }

  /** Returns the sum of the elements of the 2-dimensional array m */
  public static int sum2D(int[][] m)
  {
    int sum = 0;

    /* missing code */

    return sum;
  }
}
```

Assume that `sum1D` works correctly. Which of the following can replace `/* missing code */` so that the `sum2D` method works correctly?

I.
```
for (int k = 0; k < m.length; k++)
{
  sum += sum1D(m[k]);
}
```

II.
```
for (int[] row : m)
{
  sum += sum1D(row);
}
```

III.
```
for (int[] row : m)
{
  for (int v : row)
  {
    sum += v;
  }
}
```

(A) I only

(B) II only

(C) I and II only

(D) II and III only

(E) I, II, and III

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

25. The following `sort` method correctly sorts the integers in `elements` into ascending order.

```
Line 1:  public static void sort(int[] elements)
Line 2:  {
Line 3:    for (int j = 0; j < elements.length - 1; j++)
Line 4:    {
Line 5.      int index = j;
Line 6:
Line 7:      for (int k = j + 1; k < elements.length; k++)
Line 8:      {
Line 9:        if (elements[k] < elements[index])
Line 10:       {
Line 11:         index = k;
Line 12:       }
Line 13:     }
Line 14:
Line 15:     int temp = elements[j];
Line 16:     elements[j] = elements[index];
Line 17:     elements[index] = temp;
Line 18:   }
Line 19: }
```


© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

Which of the following changes to the `sort` method would correctly sort the integers in `elements` into **descending** order?

I. Replace line 9 with:
Line 9: `if (elements[k] > elements[index])`

II. Replace lines 15–17 with:
Line 15: `int temp = elements[index];`
Line 16: `elements[index] = elements[j];`
Line 17: `elements[j] = temp;`

III. Replace line 3 with:
Line 3: `for (int j = elements.length − 1; j > 0; j−−)`
and replace line 7 with:
Line 7: `for (int k = 0; k < j; k++)`

(A) I only

(B) II only

(C) I and II only

(D) I and III only

(E) I, II, and III

Answers to Computer Science A Multiple-Choice Questions

1 – B	6 – A	11 – C	16 – B	21 – D
2 – A	7 – A	12 – E	17 – D	22 – D
3 – D	8 – C	13 – C	18 – A	23 – C
4 – D	9 – D	14 – D	19 – D	24 – E
5 – D	10 – A	15 – D	20 – B	25 – D

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

Sample Free-Response Questions

Following is a representative set of questions. Additional sample questions can be found in the AP section of the College Board website.

Directions: SHOW ALL YOUR WORK. REMEMBER THAT PROGRAM SEGMENTS ARE TO BE WRITTEN IN JAVA.

Notes:

- Assume that the classes listed in the Java Quick Reference have been imported where appropriate
- Unless otherwise noted in the question, assume that parameters in method calls are not `null` and that methods are called only when their preconditions are satisfied.
- In writing solutions for each question, you may use any of the accessible methods that are listed in classes defined in that question. Writing significant amounts of code that can be replaced by a call to one of these methods will not receive full credit.

1. A travel agency maintains a list of information about airline flights. Flight information includes a departure time and an arrival time. You may assume that the two times occur on the same day. These times are represented by objects of the `Time` class.

   The declaration for the `Time` class is shown below. It includes a method `minutesUntil`, which returns the difference (in minutes) between the current `Time` object and another `Time` object.

```
public class Time
{
   /** @return difference, in minutes, between this time and other;
    *          difference is negative if other is earlier than this time
    */
   public int minutesUntil(Time other)
   {  /* implementation not shown */  }

   // There may be instance variables, constructors, and methods that are not
   // shown.
}
```

   For example, assume that `t1` and `t2` are `Time` objects where `t1` represents 1:00 P.M. and `t2` represents 2:15 P.M. The call `t1.minutesUntil(t2)` will return `75` and the call `t2.minutesUntil(t1)` will return `-75`.

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

The declaration for the `Flight` class is shown below. It has methods to access the departure time and the arrival time of a flight. You may assume that the departure time of a flight is earlier than its arrival time.

```
public class Flight
{
  /** @return time at which the flight departs
   */
  public Time getDepartureTime()
  { /* implementation not shown */ }

  /** @return time at which the flight arrives
   */
  public Time getArrivalTime()
  { /* implementation not shown */ }

  // There may be instance variables, constructors, and methods that are not shown.
}
```

A trip consists of a sequence of flights and is represented by the `Trip` class. The `Trip` class contains a `List` of `Flight` objects that are stored in chronological order. You may assume that for each flight after the first flight in the list, the departure time of the flight is later than the arrival time of the preceding flight in the list. A partial declaration of the `Trip` class is shown below. You will write two methods for the `Trip` class.

```
public class Trip
{
  /** The list of flights (if any) that make up this trip, stored in chronological
      order */
  private List<Flight> flights;

  /** @return the number of minutes from the departure of the first flight to the
   *          arrival of the last flight if there are one or more flights in the trip;
   *          0, if there are no flights in the trip
   */
  public int getDuration()
  {  /* to be implemented in part (a) */  }

  /**  Precondition: the departure time for each flight is later than the arrival
   *                 time of its preceding flight
   *   @return the smallest number of minutes between the arrival of
   *           a flight and the departure of the flight immediately after it,
   *           if there are two or more flights in the trip;
   *           -1, if there are fewer than two flights in the trip
   */
  public int getShortestLayover()
  {  /* to be implemented in part (b) */  }

  // There may be instance variables, constructors, and methods that are not
  // shown.
}
```

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

(a) Complete method `getDuration` below.

```
/**  @return the number of minutes from the departure of the first
 *           flight to the arrival of the last flight if there are one or
 *           more flights in the trip;
 *           0, if there are no flights in the trip
 */
public int getDuration()
```

(b) Write the `Trip` method `getShortestLayover`. A layover is the number of minutes from the arrival of one flight in a trip to the departure of the flight immediately after it. If there are two or more flights in the trip, the method should return the shortest layover of the trip; otherwise, it should return -1.

For example, assume that the instance variable `flights` of a `Trip` object `vacation` contains the following flight information.

	Departure Time	Arrival Time	Layover (minutes)
Flight 0	11:30 a.m.	12:15 p.m.	} 60
Flight 1	1:15 p.m.	3:45 p.m.	} 15
Flight 2	4:00 p.m.	6:45 p.m.	} 210
Flight 3	10:15 p.m.	11:00 p.m.	

The call `vacation.getShortestLayover()` should return 15.

Complete method `getShortestLayover` below.

```
/** Precondition: the departure time for each flight is later than the arrival
 *                time of its preceding flight
 *  @return the smallest number of minutes between the arrival of a
 *          flight and the departure of the flight immediately after it, if
 *          there are two or more flights in the trip;
 *          -1, if there are fewer than two flights in the trip
 */
public int getShortestLayover()
```


© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

2. Consider the following incomplete `StringUtil` class declaration. You will write implementations for the two methods listed in this class. Information about the `Person` class used in the `replaceNameNickname` method will be presented in part (b).

```
public class StringUtil
{
  /** @param str a String with length > 0
   *  @param oldstr a String
   *  @param newstr a String
   *  @return a new String in which all occurrences of the substring
   *          oldstr in str are replaced by the substring newstr
   */
  public static String apcsReplaceAll(String str,
                                      String oldStr,
                                      String newStr)
  {  /* to be implemented in part (a) */  }

  /** @param str a String
   *  @param people a list of references to Person objects
   *  @return a copy of str modified so that each occurrence of a first
   *          name in people is replaced by the corresponding nickname
   */
  public static String replaceNameNickname(String str,
                                           List<Person>
                                           people)
  {  /* to be implemented in part (b) */  }

  // There may be methods that are not shown.
}
```

(a) Write the `StringUtil` method `apcsReplaceAll`, which examines a given `String` and replaces all occurrences of a designated substring with another specified substring. In writing your solution, you may NOT use the `replace`, `replaceAll`, or `replaceFirst` methods in the Java `String` class.

The following table shows several examples of the result of calling `StringUtil.apcsReplaceAll(str, oldstr,0 newstr)`.

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

str	oldstr	newstr	String returned	Comment
`"to be or not to be"`	`"to"`	`"2"`	`"2 be or not 2 be"`	Each occurrence of `"to"` in the original string has been replaced by "2"
`"advanced calculus"`	`"math"`	`"science"`	`"advanced calculus"`	No change, because the string `"math"` was not in the original string
`"gogogo"`	`"go"`	`"gone"`	`"gonegonegone"`	Each occurrence of `"go"` in the original string has been replaced by `"gone"`
`"aaaaa"`	`"aaa"`	`"b"`	`"baa"`	The first occurrence of `"aaa"` in the original string has been replaced by `"b"`

Complete method `apcsReplaceAll` below.

```
/** @param str a String with length > 0
 *  @param oldstr a String
 *  @param newstr a String
 *  @ return a new String in which all occurrences of the substring
 *            oldstr in str are replaced by the substring newstr
 */
public static String apcsReplaceAll(String str,
                                    String oldStr,
                                    String newStr)
```


© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

(b) The following `Person` class contains information that includes a first (given) name and a nickname for the person.

```
public class Person
{
  /** @return the first name of this Person */
  public String getFirstName()
  { /* implementation not shown */ }

  /** @return the nickname of this Person */
  public String getNickname()
  { /* implementation not shown */ }

  // There may be instance variables, constructors, and methods not shown.
}
```

Write the `StringUtil` method `replaceNameNickname`, which takes a string and a list of `Person` objects that contain first names and a corresponding nicknames. The method is to replace all names by their nicknames in the given string. The list of `Person` objects is processed in order from lowest index to highest index. In writing your solution, you may NOT use the `replace`, `replaceAll`, or `replaceFirst` methods in the Java `String` class.

For example, assume the following table represents the data contained in the list `people`.

	`getFirstName()`	`getNickname()`
0	`"Henry"`	`"Hank"`
1	`"Elizabeth"`	`"Liz"`
2	`"John"`	`"Jack"`
3	`"Margaret"`	`"Peggy"`

Assume also that `String str` represents the following string.

```
"After Henry drove Elizabeth to dinner in Johnson City, Henry
paid for an appetizer and Elizabeth paid for dessert."
```

The call `StringUtil.replaceNameNickname(str, people)` should return the following string:

```
"After Hank drove Liz to dinner in Jackson City, Hank paid for
an appetizer and Liz paid for dessert."
```

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

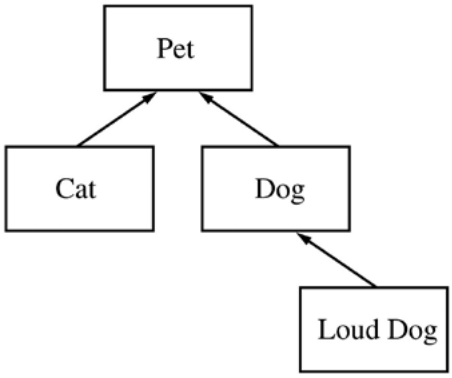
In writing your solution, you must use the method `apcsReplaceAll` specified in the `StringUtil` class. Assume that `apcsReplaceAll` works as specified, regardless of what you wrote in part (a).

Complete method `replaceNameNickname` below.

```
/** @param str a String
 *  @param people a list of references to Person objects
 *  @return a copy of str modified so that each occurrence of a first
 *          name in people is replaced by the corresponding nickname
 */
public static String replaceNameNickname(String str,
                                         List<Person> people)
```


© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

3. Consider the hierarchy of classes shown in the following diagram.



Note that a `Cat` "is-a" `Pet`, a `Dog` "is-a" `Pet`, and a `LoudDog` "is-a" `Dog`.

The class `Pet` is specified as an abstract class as shown in the following declaration. Each `Pet` has a name that is specified when it is constructed.

```
public abstract class Pet
{
  private String name;

  public Pet(String petName)
  {  name = petName;  }

  public String getName()
  {  return name;  }

  public abstract String speak();
}
```

The subclass `Dog` has the partial class declaration shown below.

```
public class Dog extends Pet
{
  public Dog(String petName)
  {  /* implementation not shown */  }

  public String speak()
  {  /* implementation not shown */  }
}
```

(a) Given the class hierarchy shown above, write a complete class declaration for the class `Cat`, including implementations of its constructor and method(s). The `Cat` method `speak` returns `"meow"` when it is invoked.

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

(b) Assume that class `Dog` has been declared as shown at the beginning of the question. If the `String` *dog-sound* is returned by the `Dog` method `speak`, then the `LoudDog` method speak returns a `String` containing *dog-sound* repeated two times.

Given the class hierarchy shown previously, write a complete class declaration for the class `LoudDog`, including implementations of its constructor and method(s).

(c) Consider the following partial declaration of class `Kennel`.

```
public class Kennel
{
  private List<Pet> petList;


  /** For every Pet in the kennel, prints the name followed by
   *  the result of a call to its speak method, one line per Pet.
   */
  public void allSpeak()
  {  /* to be implemented in part (c) */  }

  // There may be instance variables, constructors, and methods that are
  // not shown.
}
```

Write the `Kennel` method `allSpeak`. For each `Pet` in the kennel, `allSpeak` prints a line with the name of the `Pet` followed by the result of a call to its `speak` method.

In writing `allSpeak`, you may use any of the methods defined for any of the classes specified for this problem. Assume that these methods work as specified, regardless of what you wrote in parts (a) and (b).

Complete method `allSpeak` below.

```
/** For each Pet in the kennel, prints the name followed by
 *  the result of a call to its speak method, one line per Pet.
 */
public void allSpeak()
```


© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

4. This question involves manipulation of one-dimensional and two-dimensional arrays. In part (a), you will write a method to shift the elements of a one-dimensional array. In parts (b) and (c), you will write methods to shift the elements of a two-dimensional array.

   (a) Consider the following incomplete `ArrayUtil` class, which contains a `static shiftArray` method.

```
public class ArrayUtil
{
  /** Shifts each array element to the next higher index, discarding the
   *  original last element, and inserts the new number at the front.
   *  @param arr the array to manipulate
   *             Precondition: arr.length > 0
   *  @param num the new number to insert at the front of arr
   *  Postcondition: The original elements of arr have been shifted to
   *                 the next higher index, and arr[0] == num.
   *                 The original element at the highest index has been
   *                 discarded.
   */
  public static void shiftArray(int[] arr, int num)
  {  /* to be implemented in part (a) */  }

  // There may be methods that are not shown.
}
```

   Write the `ArrayUtil` method `shiftArray`. This method stores the integer `num` at the front of the array `arr` after shifting each of the original elements to the position with the next higher index. The element originally at the highest index is lost.

   For example, if `arr` is the array `{11, 12, 13, 14, 15}` and `num` is 27, the call to `shiftArray` changes `arr` as shown below.

Before call	0	1	2	3	4
arr:	11	12	13	14	15

After call	0	1	2	3	4
arr:	27	11	12	13	14

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

Complete method `shiftArray` below.

```
/** Shifts each array element to the next higher index, discarding the
 *  original last element, and inserts the new number at the front.
 *  @param arr the array to manipulate
 *             Precondition: arr.length > 0
 *  @Param num the new number to insert at the front of arr
 *  Postcondition: The original elements of arr have been shifted to
 *                 the next higher index, and arr[0] == num.
 *                 The original element at the highest index has been
 *                 discarded.
 */
public static void shiftArray(int[] arr, int num)
```


© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

(b) Consider the following incomplete `NumberMatrix` class, which represents a two-dimensional matrix of integers. Assume that the matrix contains at least one integer.

```
public class NumberMatrix
{
  private int[][] matrix;

  /** Constructs a number matrix. */
  public NumberMatrix(int[][] m)
  {  matrix = m;  }

  /** Shifts each matrix element to the next position in row-major order
   *  and inserts the new number at the front. The last element in the last
   *  row is discarded.
   *  @param num the new number to insert at the front of matrix
   *  Postcondition: The original elements of matrix have been shifted to
   *                 the next higher position in row-major order, and
   *                 matrix[0][0] == num.
   *                 The original last element in the last row is discarded.
   */
  public void shiftMatrix(int num)
  {  /* to be implemented in part (b) */  }

  /** Rotates each matrix element to the next higher position in row-major
   *  order.
   *  Postcondition: The original elements of matrix have been shifted
   *                 to the next higher position in row-major order, and
   *                 matrix[0][0] == the original last element.
   */
  public void rotateMatrix()
  {  /* to be implemented in part (c) */  }

  // There may be instance variables, constructors, and methods that are not
  // shown.
}
```

Write the `NumberMatrix` method `shiftMatrix`. This method stores a new value `num` into the two-dimensional array `matrix` after shifting the elements to the next higher position in row-major order. The element originally at the last position in row-major order is lost.

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

For example, if `m1` is a reference to a `NumberMatrix` object, then the call `m1.shiftMatrix(48)` will change the values in `matrix` as shown below.

Before call

	0	1	2
0	13	14	15
1	16	17	18
2	19	20	21
3	22	23	24

After call

	0	1	2
0	48	13	14
1	15	16	17
2	18	19	20
3	21	22	23

In writing `shiftMatrix`, you must call the `shiftArray` method in part (a). Assume that `shiftArray` works correctly regardless of what you wrote in part (a).

Complete method `shiftMatrix` below.

```
/** Shifts each matrix element to the next position in row-major order
 *  and inserts the new number at the front. The last element in the last
 *  row is discarded.
 *  @param num the new number to insert at the front of matrix
 *  Postcondition: The original elements of matrix have been shifted
 *                 to the next higher position in row-major order, and
 *                 matrix[0][0] == num.
 *                 The original last element in the last row is discarded.
 */
public void shiftMatrix(int num)
```

(c) Write the `NumberMatrix` method `rotateMatrix`. This method rotates all the elements to the next position in row-major order. The element originally at the last position is stored in the first position of the matrix.

In writing `rotateMatrix`, you must call the `shiftMatrix` method in part (b). Assume that `shiftMatrix` works correctly regardless of what you wrote in part (b).

Complete method `rotateMatrix` below.

```
/** Rotates each matrix element to the next higher position in row-major
 *  order.
 *  Postcondition: The original elements of matrix have been shifted to
 *                 the next higher position in row-major order, and
 *                 matrix[0][0] == the original last element.
 */
public void rotateMatrix()
```


© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

Suggested Solutions to Free-Response Questions

Note: There are many correct variations of these solutions.

Question 1

(a)

```
public int getDuration()
{
  if (flights.size() == 0)
  {
    return 0;
  }

  Time depart = flights.get(0).getDepartureTime();
  Time arrive = flights.get(flights.size() - 1).getArrivalTime();
  return depart.minutesUntil(arrive);
}
```

(b)

```
public int getShortestLayover()
{
  if (flights.size() < 2)
  {
    return -1;
  }

  int shortest = getDuration();
  for (int k = 1; k < flights.size(); k++)
  {
    Flight flight1 = flights.get(k - 1);
    Flight flight2 = flights.get(k);
    Time arrive = flight1.getArrivalTime();
    Time depart = flight2.getDepartureTime();
    int layover = arrive.minutesUntil(depart);
    if (layover < shortest)
    {
      shortest = layover;
    }
  }

  return shortest;
}
```

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

Question 2

(a)
Iterative version:

```
public static String apcsReplaceAll(String str,
                                    String oldStr,
                                    String newStr)
{
  String firstPart = "";
  String lastPart = str;
  int pos = lastPart.indexOf(oldStr);
  while (pos >= 0)
  {
    firstPart += lastPart.substring(0, pos);
    firstPart += newStr;
    lastPart = lastPart.substring(pos + oldStr.length());
    pos = lastPart.indexOf(oldStr);
  }
  return firstPart + lastPart;
}
```

Recursive version:

```
public static String apcsReplaceAll(String str,
                                    String oldStr,
                                    String newStr)
{
  int pos = str.indexOf(oldStr);
  if (pos < 0)
  {
    return str;
  }
  else
  {
    String firstPart = str.substring(0, pos);
    String restOfStr = str.substring(pos + oldStr.length());
    String lastPart = apcsReplaceAll(restOfStr, oldStr, newStr);
    return firstPart + newStr + lastPart;
  }
}
```


© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

(b)

```
public static String replaceNameNickname(String str,
                                         List<Person> people)
{
  for (Person p : people)
  {
    str = apcsReplaceAll(str, p.getFirstName(), p.getNickname());
  }
  return str;
}
```

Question 3

(a)

```
public class Cat extends Pet
{
  public Cat(String petName)
  {  super(petName);  }

  public String speak()
  {
    return "meow";
  }
}
```

(b)

```
public class LoudDog extends Dog
{
  public LoudDog(String petName)
  {  super(petName);  }

  public String speak()
  {
    return super.speak() + super.speak();
  }
}
```

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

(c)

```
public void allSpeak()
{
  for (Pet a : petList)
  {
    System.out.println(a.getName() + a.speak());
  }
}
```

Question 4

(a)

```
public static void shiftArray(int[] arr, int num)
{
  for (int k = arr.length − 1; k > 0; k−−)
  {
    arr[k] = arr[k - 1];
  }

  arr[0] = num;
}
```

(b)

```
public void shiftMatrix(int num)
{
  for (int[] row: matrix)
  {
    int temp = row[row.length − 1];
    ArrayUtil.shiftArray(row, num);
    num = temp;
  }
}
```

(c)

```
public void rotateMatrix()
{
  shiftMatrix(matrix[matrix.length − 1][matrix[0].length − 1]);
}
```


© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

APPENDIX A

AP Computer Science Java Subset

The AP Java subset is intended to outline the features of Java that may appear on the AP Computer Science A Exam. The AP Java subset is NOT intended as an overall prescription for computer science courses — the subset itself will need to be supplemented in order to address all topics in a typical introductory curriculum. For example, input and output must be part of a course on computer programming. However, there are many ways to handle input and output in Java. Because of this variation, details of input and output (except for basic text output using `System.out.print` and `System.out.println`) are not tested on the AP Computer Science A Exam.

This appendix describes the Java subset that students will be expected to understand when they take the AP Computer Science A Exam. A number of features are also mentioned that are potentially relevant in an introductory computer science course but are not tested on the exam. Omission of a feature from the AP Java subset does not imply any judgment that the feature is inferior or not worthwhile.

The AP Java subset was selected to

1. enable the test designers to formulate meaningful questions.
2. help students with test preparation.
3. enable instructors to follow a variety of approaches in their courses.

To help students with test preparation, the AP Java subset was intentionally kept small. Language constructs and library features were omitted that did not add significant functionality and that can, for the formulation of exam questions, be expressed by other mechanisms in the subset.

The AP Java subset gives instructors flexibility in how they use Java in their course. For example, some courses teach how to perform input/output using streams or readers/writers, others teach graphical user interface construction, and yet others rely on a tool or library that handles input/output. For the purpose of the AP Computer Science A Exam, these choices are incidental and are not central for the problem solving process or for the mastery of computer science concepts. The AP Java subset does not address handling of user input at all. That means that the subset is not complete. To create actual programs, instructors need to present additional mechanisms in their courses.

The following section contains the language features that may be tested on the AP Computer Science A Exam. The Java Quick Reference contains a list specifying which Standard Java classes, interfaces, constants, and methods may be used on the exam. This document is available to students when they take the exam, is available at AP Central, and is included in Appendix B.

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

Language Features and other Testable Topics

Tested in the AP CS A Exam	Notes	Not tested in the AP CS A Exam, but potentially relevant/useful
Comments `/* */`, `//`, and `/** */` Javadoc `@param` and `@return` comment tags		Javadoc tool
Primitive Types `int`, `double`, `boolean`		`char`, `byte`, `short`, `long`, `float`

Operators
Arithmetic: `+`, `-`, `*`, `/`, `%`
Increment/Decrement: `++`, `--`
Assignment: `=`, `+=`, `-=`, `*=`, `/=`, `%=`
Relational: `==`, `!=`, `<`, `<=`, `>`, `>=`
Logical: `!`, `&&`, `||`
Numeric casts: `(int)`, `(double)`
String concatenation: `+`
	1, 2, 3, 4, 5	`&`, `|`, `^`
		`(char)`, `(float)`
		`StringBuilder`
		Shift: `<<`, `>>`, `>>>`
		Bitwise: `~`, `&`, `|`, `^`
		Conditional: `?:`

Object Comparison object identity `(==, !=)` vs. object equality `(equals)`, `String compareTo`		implementation of `equals` `Comparable`
Escape Sequences `\"`, `\\`, `\n` inside strings		`\'`, `\t`, `\unnnn`
Input / Output `System.out.print`, `System.out.println`	6	`Scanner`, `System.in`, `System.out`, `System.err`, `Stream` input/output, GUI input/output, parsing input: `Integer.parseInt`, `Double.parseDouble` formatting output: `System.out.printf`

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

Tested in the AP CS A Exam	Notes	Not tested in the AP CS A Exam, but potentially relevant/useful

Exceptions
`ArithmeticException`,
`NullPointerException`,
`IndexOutOfBoundsException`,
`ArrayIndexOutOfBoundsException`,
`IllegalArgumentException`
		`try/catch/finally`
		`throw, throws`
		`assert`

Arrays 1-dimensional arrays, 2-dimensional rectangular arrays, initializer list: `{ ... }`, row-major order of 2-dimensional array elements	7, 8	`new` *type*`[]{ ... }`, ragged arrays (non-rectangular), arrays with 3 or more dimensions

Control Statements
`if, if/else,`
`while, for,`
enhanced `for` (for-each),
`return`
		`switch,`
		`break, continue,`
		`do-while`

Variables parameter variables, local variables, `private` instance variables: visibility `(private)` `static` (class) variables: visibility `(public, private)`, `final`		`final` parameter variables, `final` local variables, `final` instance variables

Methods
visibility `(public, private)`,
`static, non-static,`
method signatures,
overloading, overriding,
parameter passing
	9, 10	visibility `(protected)`,
		`public static void`
		`main(String[] args)`,
		command line arguments,
		variable number of parameters,
		`final`

Constructors `super(), super(`*args*`)`	11, 12	default initialization of instance variables, initialization blocks, `this(`*args*`)`

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

Tested in the AP CS A Exam	Notes	Not tested in the AP CS A Exam, but potentially relevant/useful
Classes `new`, visibility `(public)`, accessor methods, modifier (mutator) methods Design/create/modify class. Create subclass of a superclass `(abstract, non-abstract)`. Create class that implements an interface.	13, 14	`final`, visibility `(private, protected)`, nested classes, inner classes, enumerations
Interfaces Design/create/modify an interface.	13, 14	
Inheritance Understand inheritance hierarchies. Design/create/modify subclasses. Design/create/modify classes that implement interfaces.		
Packages `import` *packageName.className*		`import` *packageName*`.*` , `static` import, `package` *packageName* , class path
Miscellaneous OOP "is-a" and "has-a" relationships, `null`, `this`, `super.`*method(args)*	15, 16	`instanceof` *(class)* cast `this.`*var*, `this.`*method*`(`*args*`)`,
Standard Java Library `Object`, `Integer, Double`, `String`, `Math`, `List<E>, ArrayList<E>`	17, 18	`clone`, autoboxing, `Collection<E>`, `Arrays, Collections`

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

Notes

1. Students are expected to understand the operator precedence rules of the listed operators.
2. The increment/decrement operators `++` and `--` are part of the AP Java subset. These operators are used only for their side effect, not for their value. That is, the postfix form (for example, `x++`) is always used, and the operators are not used inside other expressions. For example, `arr[x++]` is not used.
3. Students need to understand the "short circuit" evaluation of the `&&` and `||` operators.
4. Students are expected to understand "truncation towards `0`" behavior as well as the fact that positive floating-point numbers can be rounded to the nearest integer as

   `(int)(x + 0.5)`, negative numbers as `(int)(x - 0.5)`.
5. String concatenation `+` is part of the AP Java subset. Students are expected to know that concatenation converts numbers to strings and invokes `toString` on objects.
6. User input is not included in the AP Java subset. There are many possible ways for supplying user input: e.g., by reading from a `Scanner`, reading from a stream (such as a file or a URL), or from a dialog box. There are advantages and disadvantages to the various approaches. The exam does not prescribe any one approach. Instead, if reading input is necessary, it will be indicated in a way similar to the following:

   ```
   double x = /* call to a method that reads a floating-point number */;
   ```

   or

   ```
   double x = ...;    // read user input
   ```
7. Both arrays of primitive types (e.g., `int[]`, `int[][]`) and arrays of objects (e.g., `Student[]`, `Student[][]`) are in the subset.
8. Students need to understand that 2-dimensional arrays are stored as arrays of arrays. For the purposes of the AP CS A Exam, students should assume that 2-dimensional arrays are rectangular (not ragged) and the elements are indexed in row-major order. For example, given the declaration

   ```
   int[][] m = {{1, 2, 3}, {4, 5, 6}};
   ```

   `m.length` is 2 (the number of rows), `m[0].length` is 3 (the number of columns), `m[r][c]` represents the element at row `r` and column `c`, and `m[r]` represents row `r` (e.g., `m[0]` is of type `int[]` and references the array `{1, 2, 3}`).

   Students are expected to be able to access a row of a 2-dimensional array, assign it to a 1-dimensional array reference, pass it as a parameter, and use loops (including for-each) to traverse the rows. However, students are not expected to analyze or implement code that replaces an entire row in a 2-dimensional array, such as

   ```
   int[][] m = {{1, 2, 3}, {4, 5, 6}};
   int[] a = {7, 8, 9};
   m[0] = a;    // Outside the Subset
   ```

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

9. The `main` method and command-line arguments are not included in the subset. In free-response questions, students are not expected to invoke programs. In the *AP Computer Science Labs*, program invocation with `main` may occur, but the `main` method will be kept very simple.
10. Students are required to understand when the use of `static` methods is appropriate. In the exam, `static` methods are always invoked through a class (explicitly or implicitly), never an object (i.e., *ClassName.staticMethod*() or *staticMethod*(), not *obj.staticMethod*()).
11. If a subclass constructor does not explicitly invoke a superclass constructor, the Java compiler automatically inserts a call to the no-argument constructor of the superclass.
12. Students are expected to implement constructors that initialize all instance variables. Class constants are initialized with an initializer:

    ```
    public static final int MAX_SCORE = 5;
    ```

    The rules for default initialization (with `0`, `false` or `null`) are not included in the subset. Initializing instance variables with an initializer is not included in the subset. Initialization blocks are not included in the subset.
13. Students are expected to write interfaces or class declarations when given a general description of the interface or class.
14. Students are expected to extend classes and implement interfaces. Students are also expected to have knowledge of inheritance that includes understanding the concepts of method overriding and polymorphism. Students are expected to implement their own subclasses.

    Students are expected to read the definition of an abstract class and understand that the abstract methods need to be implemented in a subclass. Students are similarly expected to read the definition of an interface and understand that the abstract methods need to be implemented in an implementing class.
15. Students are expected to understand that conversion from a subclass reference to a superclass reference is legal and does not require a cast. Class casts (generally from `Object` to another class) are not included in the AP Java subset. Array type compatibility and casts between array types are not included in the subset.
16. The use of `this` is restricted to passing the implicit parameter in its entirety to another method (e.g., *obj.method*`(this)`) and to descriptions such as "the implicit parameter `this`". Students are not required to know the idiom "`this.`*var* = *var*", where *var* is both the name of an instance variable and a parameter variable.
17. The use of generic collection classes and interfaces is in the AP Java subset, but students need not implement generic classes or methods.
18. Students are expected to know a subset of the constants and methods of the listed Standard Java Library classes and interfaces. Those constants and methods are enumerated in the Java Quick Reference (Appendix B).

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

APPENDIX B

Exam Appendix — Java Quick Reference

Accessible methods from the Java library that may be included on the exam

class java.lang.Object

- `boolean equals(Object other)`
- `String toString()`

class java.lang.Integer

- `Integer(int value)`
- `int intValue()`
- `Integer.MIN_VALUE` // minimum value represented by an `int` or `Integer`
- `Integer.MAX_VALUE` // maximum value represented by an `int` or `Integer`

class java.lang.Double

- `Double(double value)`
- `double doubleValue()`

class java.lang.String

- `int length()`
- `String substring(int from, int to)` // returns the substring beginning at `from`
  // and ending at `to-1`
- `String substring(int from)` // returns `substring(from, length())`
- `int indexOf(String str)` // returns the index of the first occurrence of `str`;
  // returns `-1` if not found
- `int compareTo(String other)` // returns a value < 0 if `this` is less than `other`
  // returns a value = 0 if `this` is equal to `other`
  // returns a value > 0 if `this` is greater than `other`

class java.lang.Math

- `static int abs(int x)`
- `static double abs(double x)`
- `static double pow(double base, double exponent)`
- `static double sqrt(double x)`
- `static double random()` // returns a `double` in the range [0.0, 1.0)

interface java.util.List<E>

- `int size()`
- `boolean add(E obj)` // appends `obj` to end of list; returns `true`
- `void add(int index, E obj)` // inserts `obj` at position `index` ($0 \le \texttt{index} \le \texttt{size}$),
  // moving elements at position `index` and higher
  // to the right (adds 1 to their indices) and adjusts size
- `E get(int index)`
- `E set(int index, E obj)` // replaces the element at position `index` with `obj`
  // returns the element formerly at the specified position
- `E remove(int index)` // removes element from position `index`, moving elements
  // at position `index + 1` and higher to the left
  // (subtracts 1 from their indices) and adjusts size
  // returns the element formerly at the specified position

class java.util.ArrayList<E> implements java.util.List<E>

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

APPENDIX C — SAMPLE SEARCH AND SORT ALGORITHMS

Sequential Search

The Sequential Search Algorithm below finds the index of a value in an array of integers as follows:

1. Traverse `elements` until `target` is located, or the end of `elements` is reached.
2. If `target` is located, return the index of `target` in `elements`; Otherwise return `-1`.

```
/**
 * Finds the index of a value in an array of integers.
 *
 * @param elements an array containing the items to be searched.
 * @param target the item to be found in elements.
 * @return an index of target in elements if found; -1 otherwise.
 */
public static int sequentialSearch(int[] elements, int target)
{
  for (int j = 0; j < elements.length; j++)
  {
    if (elements[j] == target)
    {
      return j;
    }
  }

  return -1;
}
```


© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

Binary Search

The Binary Search Algorithm below finds the index of a value in an array of integers sorted in ascending order as follows:

1. Set `left` and `right` to the minimum and maximum indexes of `elements` respectively.
2. Loop until `target` is found, or `target` is determined not to be in `elements` by doing the following for each iteration:
   a. Set `middle` to the index of the middle item in `elements[left] ... elements[right]` inclusive.
   b. If `target` would have to be in `elements[left] ... elements[middle - 1]` inclusive, then set `right` to the maximum index for that range.
   c. Otherwise, if `target` would have to be in `elements[middle + 1] ... elements[right]` inclusive, then set `left` to the minimum index for that range.
   d. Otherwise, return `middle` because `target == elements[middle]`.
3. Return `-1` if `target` is not contained in `elements`.

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

```
/**
 * Find the index of a value in an array of integers sorted in ascending order.
 *
 * @param elements an array containing the items to be searched.
 *                 Precondition: items in elements are sorted in ascending order.
 * @param target the item to be found in elements.
 * @return an index of target in elements if target found;
 *         -1 otherwise.
 */
public static int binarySearch(int[] elements, int target)
{
  int left = 0;
  int right = elements.length - 1;

  while (left <= right)
  {
    int middle = (left + right) / 2;
    if (target < elements[middle])
    {
      right = middle - 1;
    }
    else if (target > elements[middle])
    {
      left = middle + 1;
    }
    else
    {
      return middle;
    }
  }

  return -1;
}
```


© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

Selection Sort

The Selection Sort Algorithm below sorts an array of integers into ascending order as follows:

1. Loop from `j = 0` to `j = elements.length−2`, inclusive, completing `elements.length−1` passes.
2. In each pass, swap the item at index `j` with the minimum item in the rest of the array (`elements[j+1]` through `elements[elements.length−1]`).

At the end of each pass, items in `elements[0]` through `elements[j]` are in ascending order and each item in this sorted portion is at its final position in the array

```
/**
 * Sort an array of integers into ascending order.
 *
 * @param elements an array containing the items to be sorted.
 *
 * Postcondition: elements contains its original items and items in elements
 *                are sorted in ascending order.
 */
public static void selectionSort(int[] elements)
{
  for (int j = 0; j < elements.length − 1; j++)
  {
     int minIndex = j;

     for (int k = j + 1; k < elements.length; k++)
     {
       if (elements[k] < elements[minIndex])
       {
          minIndex = k;
       }
     }

     int temp = elements[j];
     elements[j] = elements[minIndex];
     elements[minIndex] = temp;
  }
}
```

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

Insertion Sort

The Insertion Sort Algorithm below sorts an array of integers into ascending order as follows:

1. Loop from `j = 1` to `j = elements.length−1` inclusive, completing `elements.length−1` passes.
2. In each pass, move the item at index `j` to its proper position in `elements[0]` to `elements[j]`:
   a. Copy item at index `j` to `temp`, creating a "vacant" element at index `j` (denoted by `possibleIndex`).
   b. Loop until the proper position to maintain ascending order is found for `temp`.
   c. In each inner loop iteration, move the "vacant" element one position lower in the array.
3. Copy `temp` into the identified correct position (at `possibleIndex`).

At the end of each pass, items at `elements[0]` through `elements[j]` are in ascending order.

```
/**
 * Sort an array of integers into ascending order.
 *
 * @param elements an array containing the items to be sorted.
 *
 * Postcondition: elements contains its original items and items in elements
 *                are sorted in ascending order.
 */
public static void insertionSort(int[] elements)
{
  for (int j = 1; j < elements.length; j++)
  {
    int temp = elements[j];
    int possibleIndex = j;
    while (possibleIndex > 0 && temp < elements[possibleIndex - 1])
    {
      elements[possibleIndex] = elements[possibleIndex - 1];
      possibleIndex--;
    }
    elements[possibleIndex] = temp;
  }
}
```


© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

Merge Sort

The Merge Sort Algorithm below sorts an array of integers into ascending order as follows:

`mergeSort`

This top-level method creates the necessary temporary array and calls the `mergeSortHelper` recursive helper method.

`mergeSortHelper`

This recursive helper method uses the Merge Sort Algorithm to sort `elements[from]` ... `elements[to]` inclusive into ascending order:

1. If there is more than one item in this range,
   a. divide the items into two adjacent parts, and
   b. call `mergeSortHelper` to recursively sort each part, and
   c. call the `merge` helper method to merge the two parts into sorted order.
2. Otherwise, exit because these items are sorted.

`merge`

This helper method merges two adjacent array parts, each of which has been sorted into ascending order, into one array part that is sorted into ascending order:

1. As long as both array parts have at least one item that hasn't been copied, compare the first un-copied item in each part and copy the minimal item to the next position in `temp`.
2. Copy any remaining items of the first part to `temp`.
3. Copy any remaining items of the second part to `temp`.
4. Copy the items from `temp[from]` ... `temp[to]` inclusive to the respective locations in `elements`.

```
/**
 * Sort an array of integers into ascending order.
 *
 * @param elements an array containing the items to be sorted.
 *
 * Postcondition: elements contains its original items and items in elements
 *                are sorted in ascending order.
 */
public static void mergeSort(int[] elements)
{
  int n = elements.length;
  int[] temp = new int[n];
  mergeSortHelper(elements, 0, n - 1, temp);
}
```

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

```
/**
 * Sorts elements[from] ... elements[to] inclusive into ascending order.
 *
 * @param elements an array containing the items to be sorted.
 * @param from the beginning index of the items in elements to be sorted.
 * @param to the ending index of the items in elements to be sorted.
 * @param temp a temporary array to use during the merge process.
 *
 * Precondition:
 *     (elements.length == 0 or
 *      0 <= from <= to <= elements.length) and
 *     elements.length == temp.length
 * Postcondition: elements contains its original items and the items in elements
 *                [from] ... <= elements[to] are sorted in ascending order.
 */
private static void mergeSortHelper(int[] elements,
                                    int from, int to, int[] temp)
{
  if (from < to)
  {
    int middle = (from + to) / 2;
    mergeSortHelper(elements, from, middle, temp);
    mergeSortHelper(elements, middle + 1, to, temp);
    merge(elements, from, middle, to, temp);
  }
}
```


© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

```
/**
 * Merges two adjacent array parts, each of which has been sorted into ascending
 * order, into one array part that is sorted into ascending order.
 *
 * @param elements an array containing the parts to be merged.
 * @param from the beginning index in elements of the first part.
 * @param mid the ending index in elements of the first part.
 *        mid+1 is the beginning index in elements of the second part.
 * @param to the ending index in elements of the second part.
 * @param temp a temporary array to use during the merge process.
 *
 * Precondition: 0 <= from <= mid <= to <= elements.length and
 *   elements[from] ... <= elements[mid] are sorted in ascending order and
 *   elements[mid + 1] ... <= elements[to] are sorted in ascending order and
 *   elements.length == temp.length
 * Postcondition: elements contains its original items and
 *   elements[from] ... <= elements[to] are sorted in ascending order and
 *   elements[0] ... elements[from - 1] are in original order and
 *   elements[to + 1] ... elements[elements.length - 1] are in original order.
 */
private static void merge(int[] elements,
                          int from, int mid, int to, int[] temp)
{
  int i = from;
  int j = mid + 1;
  int k = from;

  while (i <= mid && j <= to)
  {
    if (elements[i] < elements[j])
    {
      temp[k] = elements[i];
      i++;
    }
    else
    {
      temp[k] = elements[j];
      j++;
    }
    k++;
  }
```

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

```
  while (i <= mid)
  {
    temp[k] = elements[i];
    i++;
    k++;
  }

  while (j <= to)
  {
    temp[k] = elements[j];
    j++;
    k++;
  }

  for (k = from; k <= to; k++)
  {
    elements[k] = temp[k];
  }
}
```


© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

Resources for AP Teachers

AP Central (apcentral.collegeboard.org)

Essential course resources, including the Course Description and other official publications

- AP Exam Information and Resources, including practice exams
- Classroom resources, including curriculum modules, labs, and more — from both the AP program and AP teachers
- A database of upcoming professional development workshops and summer institutes

AP Course Audit

- Curricular/resource requirements
- Four annotated sample syllabi
- Syllabus development guides
- Example textbook lists
- Syllabus development tutorial

Advances in AP

Learn about forthcoming changes to AP courses

AP Teacher Communities

Join a lively community of fellow teachers by discussing all things AP on the discussion boards. Share strategies, ask questions, and engage in lively discussions with teachers worldwide.

Higher Ed

- Detailed information about each course and exam
- An overview of the course and exam redesign and what it means for colleges and universities
- Information about the new AP | Cambridge Capstone Program pilot
- Research reports on AP students' performance in subsequent college course and their progress towards a degree
- Guides and resources for effective policy review
- Promising practices in admission and policy-setting

College Board Store

Released exams, lab manuals, teachers' guides, and other publications

© 2014 The College Board. Visit the College Board on the Web: www.collegeboard.org.

Contact Us

New York Office
45 Columbus Avenue
New York, NY 10023-6992
212-713-8000
212-713-8277/55 (Fax)

AP Services
P.O. Box 6671
Princeton, NJ 08541-6671
609-771-7300
888-225-5427 (toll free in the U.S. and Canada)
610-290-8979 (Fax)
Email: apexams@info.collegeboard.org

AP Canada Office
2950 Douglas Street, Suite 550
Victoria, BC, Canada V8T 4N4
250-472-8561
800-667-4548 (toll free in Canada only)
Email: gewonus@ap.ca

International Services
Serving all countries outside the U.S. and Canada
45 Columbus Avenue
New York, NY 10023-6992
212-373-8738
Email: international@collegeboard.org

Middle States Regional Office
Serving Delaware, District of Columbia, Maryland, New Jersey, New York, Pennsylvania, Puerto Rico, and the U.S. Virgin Islands
Three Bala Plaza East, Suite 501
Bala Cynwyd, PA 19004-1501
866-392-3019
610-227-2580 (Fax)
Email: msro@info.collegeboard.org

Midwestern Regional Office
Serving Illinois, Indiana, Iowa, Kansas, Michigan, Minnesota, Missouri, Nebraska, North Dakota, Ohio, South Dakota, West Virginia, and Wisconsin
8700 West Bryn Mawr Avenue, Suite 900N
Chicago, IL 60631-3512
866-392-4086
847-653-4528 (Fax)
Email: mro@info.collegeboard.org

New England Regional Office
Serving Connecticut, Maine, Massachusetts, New Hampshire, Rhode Island, and Vermont
1601 Trapelo Road, Suite 12
Waltham, MA 02451-7333
866-392-4089
781-663-2743 (Fax)
Email: nero@info.collegeboard.org

Southern Regional Office
Serving Alabama, Florida, Georgia, Kentucky, Louisiana, Mississippi, North Carolina, South Carolina, Tennessee and Virginia
3700 Crestwood Parkway, Suite 700
Duluth, GA 30096-7155
866-392-4088
770-225-4062 (Fax)
Email: sro@info.collegeboard.org

Southwestern Regional Office
Serving Arkansas, New Mexico, Oklahoma, and Texas
4330 Gaines Ranch Loop, Suite 200
Austin, TX 78735-6735
866-392-3017
512-721-1841 (Fax)
Email: swro@info.collegeboard.org

Western Regional Office
Serving Alaska, Arizona, California, Colorado, Hawaii, Idaho, Montana, Nevada, Oregon, Utah, Washington, and Wyoming
2099 Gateway Place, Suite 550
San Jose, CA 95110-1017
866-392-4078
408-367-1459 (Fax)
Email: wro@info.collegeboard.org

apcentral.collegeboard.org

I.N. 120085144