AP Computer Science A

PREVIEW REVISED COURSE FRAMEWORK

Note: Earliest possible implementation 2025-2026 School Year

What AP® Stands For

Thousands of Advanced Placement teachers have contributed to the principles articulated here. These principles are not new; they are, rather, a reminder of how AP already works in classrooms nationwide. The following principles are designed to ensure that teachers' expertise is respected, required course content is understood, and that students are academically challenged and free to make up their own minds.

1. AP stands for clarity and transparency. Teachers and students deserve clear expectations. The Advanced Placement Program makes public its course frameworks and sample assessments. Confusion about what is permitted in the classroom disrupts teachers and students as they navigate demanding work.
2. AP is an unflinching encounter with evidence. AP courses enable students to develop as independent thinkers and to draw their own conclusions. Evidence and the scientific method are the starting place for conversations in AP courses.
3. AP opposes censorship. AP is animated by a deep respect for the intellectual freedom of teachers and students alike. If a school bans required topics from their AP courses, the AP Program removes the AP designation from that course and its inclusion in the AP Course Ledger provided to colleges and universities. For example, the concepts of evolution are at the heart of college biology, and a course that neglects such concepts does not pass muster as AP Biology.
4. AP opposes indoctrination. AP students are expected to analyze different perspectives from their own, and no points on an AP Exam are awarded for agreement with any specific viewpoint. AP students are not required to feel certain ways about themselves or the course content. AP courses instead develop students' abilities to assess the credibility of sources, draw conclusions, and make up their own minds.

   As the AP English Literature course description states: "AP students are not expected or asked to subscribe to any one specific set of cultural or political values, but are expected to have the maturity to analyze perspectives different from their own and to question the meaning, purpose, or effect of such content within the literary work as a whole."
5. AP courses foster an open-minded approach to the histories and cultures of different peoples. The study of different nationalities, cultures, religions, races, and ethnicities is essential within a variety of academic disciplines. AP courses ground such studies in primary sources so that students can evaluate experiences and evidence for themselves.
6. Every AP student who engages with evidence is listened to and respected. Students are encouraged to evaluate arguments but not one another. AP classrooms respect diversity in backgrounds, experiences, and viewpoints. The perspectives and contributions of the full range of AP students are sought and considered. Respectful debate of ideas is cultivated and protected; personal attacks have no place in AP.
7. AP is a choice for parents and students. Parents and students freely choose to enroll in AP courses. Course descriptions are available online for parents and students to inform their choice. Parents do not define which college-level topics are suitable within AP courses; AP course and exam materials are crafted by committees of professors and other expert educators in each field. AP courses and exams are then further validated by the American Council on Education and studies that confirm the use of AP scores for college credits by thousands of colleges and universities nationwide.

The AP Program encourages educators to review these principles with parents and students so they know what to expect in an AP course. Advanced Placement is always a choice, and it should be an informed one. AP teachers should be given the confidence and clarity that once parents have enrolled their child in an AP course, they have agreed to a classroom experience that embodies these principles.

© 2024 The Advanced Placement Program, AP, AP Central, and the acorn logo are registered trademarks of the College Board. All other products and services may be trademarks of their respective owners. Visit College Board on the web: **collegeboard.org**.

Contents

Acknowledgements

College Board would like to acknowledge the following committee members, consultants, and reviewers for their assistance with and commitment to the development of this course. All individuals and their affiliations were current at the time of contribution.

Don Blaheta, *Longwood University, Farmville, Virginia*
Alistar Campbell, *Hamilton College, Clinton, New York*
Adrienne Decker, *University of Buffalo, Buffalo, New York*
Dave Feinberg, *Columbus Academy, Gahanna, Ohio*
Tim Gallagher, *Winter Springs High School, Winter Springs, Florida*
Bo Hatfield, *Salem University, Salem, Massachusetts*
Cody Henrichsen, *Canyons Technical Education Center, Sandy, Utah*
Helen Hu, *Westminster College, Salt Lake City, Utah*
Daniela Marghitu, *Auburn University, Auburn, Alabama*
Sage Miller, *Webster Schroeder High School, Webster, New York*
Jerone Mitchell, *The Overlake School, Redmond, Washington*
Mohammad Mohammadi, *SUNY Oswego, Oswego, New York*
Briana Morrison, *University of Virginia, Charlottesville, Virginia*
Jill Pieritz, *Girls Preparatory School, Chattanooga, Tennessee*
Blythe Samuels, *Powhatan High School, Powhatan, Virginia*
Rob Schultz, *Bellbrook High School, Bellbrook, Ohio*
Schinnel Small, *University of South Florida, Tampa, Florida*
Lester Wainwright, *Charlottesville High School, Charlottesville, Virginia*
Kevin Wayne, *Princeton University, Princeton, New Jersey*

College Board Staff

Deborah Klipp, *Director, AP Computer Science A Course Lead*
Daniel Klag, *Senior Director, AP Computer Science Assessment Lead*
Jason VanBilliard, *Senior Director, AP Math and Computer Science Department Head*
Shu-Kang Chen, *Executive Director, AP STEM Department Head*
Sarah Muller, *Director, AP Product Development and Editorial*
Claire Lorenz, *Senior Director, AP Classroom Instruction Products*
Monica Roman, *Senior Director, AP Program Management*
Alesha Fox, *Executive Director, Course and Program Innovation*
Daniel McDonough, *Senior Director, AP Content and Assessment Publications*
Natalya Tabony, *Executive Director, AP Strategy & Analytics*
Dana Kopelman, *Executive Director, AP Instruction & Assessment Production*
Allison Thurber, *Vice President, AP Curriculum and Assessment*

SPECIAL THANKS *Crystal Furman and Maureen Reyes*

© 2024 College Board

About AP

The Advanced Placement® Program (AP®) enables willing and academically prepared students to pursue college-level studies—with the opportunity to earn college credit, advanced placement, or both—while still in high school. Through AP courses in 40 subjects, each culminating in a challenging exam, students learn to think critically, construct solid arguments, and see many sides of an issue—skills that prepare them for college and beyond. Taking AP courses demonstrates to college admission officers that students have sought the most challenging curriculum available to them, and research indicates that students who score a 3 or higher on an AP Exam typically experience greater academic success in college and are more likely to earn a college degree than non-AP students. Each AP teacher's syllabus is evaluated and approved by faculty from some of the nation's leading colleges and universities, and AP Exams are developed and scored by college faculty and experienced AP teachers. Most four-year colleges and universities in the United States grant credit, advanced placement, or both on the basis of successful AP Exam scores—more than 3,300 institutions worldwide annually receive AP scores.

AP Course Development

In an ongoing effort to maintain alignment with best practices in college-level learning, AP courses and exams emphasize challenging, research-based curricula aligned with higher education expectations.

Individual teachers are responsible for designing their own curriculum for AP courses, selecting appropriate college-level readings, assignments, and resources. This course and exam description presents the content and skills that are the focus of the corresponding college course and that appear on the AP Exam. It also organizes the content and skills into a series of units that represent a sequence found in widely adopted college textbooks and that many AP teachers have told us they follow in order to focus their instruction. The intention of this publication is to respect teachers' time and expertise by providing a roadmap that they can modify and adapt to their local priorities and preferences. Moreover, by organizing the AP course content and skills into units, the AP Program is able to provide teachers and students with free formative assessments—Progress Checks—that teachers can assign throughout the year to measure student progress as they acquire content knowledge and develop skills.

Enrolling Students: Equity and Access

The AP Program 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. College Board also believes that all students should have access to academically challenging coursework 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.

Offering AP Courses: The AP Course Audit

The AP Program unequivocally supports the principle that each school implements its own curriculum that will enable students to develop the content understandings and skills described in the course framework.

While the unit sequence represented in this publication is optional, the AP Program does have a short list of curricular and resource requirements that must be fulfilled before a school can label a course "Advanced Placement" or "AP." Schools wishing to offer AP courses must participate in the AP Course Audit, a process through which AP teachers' course materials are reviewed by college faculty. The AP Course Audit was created 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' courses meet or exceed the curricular and resource expectations that college and secondary school faculty have established for college-level courses.

© 2024 College Board

The AP Course Audit form is submitted by the AP teacher and the school principal (or designated administrator) to confirm awareness and understanding of the curricular and resource requirements. A syllabus or course outline, detailing how course requirements are met, is submitted by the AP teacher for review by college faculty.

Please visit **collegeboard.org/apcourseaudit** for more information to support the preparation and submission of materials for the AP Course Audit.

How the AP Program Is Developed

The scope of content for an AP course and exam is derived from an analysis of hundreds of syllabi and course offerings of colleges and universities. Using this research and data, a committee of college faculty and expert AP teachers work within the scope of the corresponding college course to articulate what students should know and be able to do upon the completion of the AP course. The resulting course framework is the heart of this course and exam description and serves as a blueprint of the content and skills that can appear on an AP Exam.

The AP Test Development Committees are responsible for developing each AP Exam, ensuring the exam questions are aligned to the course framework. The AP Exam development process is a multiyear endeavor; all AP Exams undergo extensive review, revision, piloting, and analysis to ensure that questions are accurate, fair, and valid, and that there is an appropriate spread of difficulty across the questions.

Committee members are selected to represent a variety of perspectives and institutions (public and private, small and large schools and colleges), and a range of gender, racial/ethnic, and regional groups. A list of each subject's current AP Test Development Committee members is available on **apcentral.collegeboard.org**.

Throughout AP course and exam development, 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 and/or college credit.

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 and through-course performance assessments, as applicable, are scored by thousands of college faculty and expert AP teachers. Most are scored at the annual AP Reading, while a small portion is scored online. All AP 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 serves as Chief Faculty Consultant and, with the help of AP Readers in leadership positions, maintains the accuracy of the scoring standards. Scores on the free-response questions and performance assessments are weighted and combined with the results of the computer-scored multiple-choice questions, and this raw score is converted into a composite AP score on a 1–5 scale.

AP Exams are **not** norm-referenced or graded on a curve. Instead, they are criterion-referenced, which means that every student who meets the criteria for an AP score of 2, 3, 4, or 5 will receive that score, no matter how many students that is. The criteria for the number of points a student must earn on the AP Exam to receive scores of 3, 4, or 5—the scores that research consistently validates for credit and placement purposes—include:

- The number of points successful college students earn when their professors administer AP Exam questions to them.
- Performance that researchers have found to be predictive of an AP student succeeding when placed into a subsequent higher-level college course.
- The number of points college faculty indicate, after reviewing each AP question, that they expect is necessary to achieve each AP grade level.

Using and Interpreting AP Scores

The extensive work done by college faculty and AP teachers in the development of the course and exam and throughout the scoring process ensures that AP Exam scores accurately represent students' achievement in the equivalent college course. Frequent and regular research studies establish the validity of AP scores as follows:

AP Score	Credit Recommendation	College Grade Equivalent
5	Extremely well qualified	A
4	Well qualified	A-, B+, B
3	Qualified	B-, C+, C
2	Possibly qualified	n/a
1	No recommendation	n/a

© 2024 College Board

While colleges and universities are responsible for setting their own credit and placement policies, most private colleges and universities award credit and/or advanced placement for AP scores of 3 or higher. Additionally, most states in the U.S. have adopted statewide credit policies that ensure college credit for scores of 3 or higher at public colleges and universities. To confirm a specific college's AP credit/placement policy, a search engine is available at **apstudent.org/creditpolicies**.

BECOMING AN AP READER

Each June, thousands of AP teachers and college faculty members from around the world gather for seven days in multiple locations to evaluate and score the free-response sections of the AP Exams. Ninety-eight percent of surveyed educators who took part in the AP Reading say it was a positive experience.

There are many reasons to consider becoming an AP Reader, including opportunities to:

- **Bring positive changes to the classroom:** Surveys show that the vast majority of returning AP Readers—both high school and college educators—make improvements to the way they teach or score because of their experience at the AP Reading.
- **Gain in-depth understanding of AP Exam and AP scoring standards:** AP Readers gain exposure to the quality and depth of the responses from the entire pool of AP Exam takers, and thus are better able to assess their students' work in the classroom.
- **Receive compensation:** AP Readers are compensated for their work during the Reading. Expenses, lodging, and meals are covered for Readers who travel.
- **Score from home:** AP Readers have online distributed scoring opportunities for certain subjects. Check **collegeboard.org/apreading** for details.
- **Earn Continuing Education Units (CEUs):** AP Readers earn professional development hours and CEUs that can be applied to PD requirements by states, districts, and schools.

How to Apply

Visit **collegeboard.org/apreading** for eligibility requirements and to start the application process.

© 2024 College Board

About the AP Computer Science A Course

AP Computer Science A introduces students to computer science through programming. Fundamental topics in this course include the design of solutions to problems, the use of data structures to organize large sets of data, the development and implementation of algorithms to process data and discover new information, the analysis of potential solutions, and the ethical and social implications of computing systems. The course emphasizes object-oriented programming and design using the Java programming language.

College Course Equivalent

AP Computer Science A is equivalent to a first-semester, college-level course in computer science.

Prerequisites

It is recommended that a student in the AP Computer Science A course has successfully completed a first-year high school algebra course with a strong foundation of basic linear functions, composition of functions, and problem-solving strategies that require multiple approaches and collaborative efforts. In addition, students should be able to use a Cartesian (x, y) coordinate system to represent points on a plane. 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.

Computer Language

The AP Computer Science A course requires that solutions of problems be written in the Java programming language. 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.

Lab Requirement

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 includes a substantial lab component in which students design solutions to problems, express their solutions precisely (e.g., in the Java programming language), test their solutions, identify and correct errors (when mistakes occur), and compare possible solutions. College Board has developed several labs that are aligned to the course framework that fulfill the 20-hour lab requirement. The class period recommendations provided in the unit guides account for the time needed to complete each lab activity as described in the lab guide.

© 2024 College Board

AP COMPUTER SCIENCE A

Course Framework


elloWorld.ja
*/
public class He
{
pu
ain

Course Framework Components

Course Units

Unit 1: Using Objects and Methods

Unit 2: Selection and Iteration

Unit 3: Class Creation

Unit 4: Data Collections

Curriculum Framework Overview

This curriculum framework provides a clear and detailed description of the course requirements necessary for student success. The framework specifies what students must know, be able to do, and understand to qualify for college credit and/or placement.

The course framework includes two essential components:

- **Computational Thinking Practices**
  The computational thinking practices are central to the study and practice of computer science. Students should practice and develop these skills on a regular basis over the span of the course.
- **Course Content**
  The course content is organized into commonly taught units of study that provide a suggested sequence for the course. These units comprise the content and conceptual understandings that colleges and universities typically expect students to master to qualify for college credit and/or placement.

© 2024 College Board

Computational Thinking Practices: Skills

Practice 1	Practice 2	Practice 3	Practice 4	Practice 5
Design Code	*Develop Code*	*Analyze Code*	*Document Code and Computing Systems*	*Use Computers Responsibly*
1.A: Determine an appropriate program design to solve a problem or accomplish a task. **1.B:** Determine what knowledge can be extracted from data.	**2.A:** Write program code to implement an algorithm. **2.B:** Write program code involving data abstractions. **2.C:** Write program code involving procedural abstractions.	**3.A:** Determine the result or output based on statement execution order in an algorithm. **3.B:** Determine the result or output based on code that contains data abstractions. **3.C:** Determine the result or output based on code that contains procedural abstractions. **3.D:** Explain why a code segment will not compile or work as intended and modify the code to correct the error.	**4.A:** Describe the behavior of a code segment or program. **4.B:** Describe the initial conditions that must be met for a code segment to work as intended or described.	**5.A:** Explain how computing impacts society, economy, and culture.

© 2024 College Board

AP COMPUTER SCIENCE A

UNIT 1

Using Objects and Methods

© 2024 College Board

THIS PAGE IS INTENTIONALLY LEFT BLANK.

TOPIC 1.1
Introduction to Algorithms, Programming, and Compilers

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
1.1.A Represent patterns and algorithms found in everyday life using written language or diagrams.	**1.1.A.1** *Algorithms* define step-by-step processes to follow when completing a task or solving a problem. These algorithms can be represented using written language or diagrams. **1.1.A.2** *Sequencing* defines an order for when steps in a process are completed. Steps in a process are completed one at a time.
1.1.B Explain the code compilation and execution process.	**1.1.B.1** Code can be written in any text editor; however, an *Integrated Development Environment (IDE)* is often used to write programs because it provides tools for a programmer to write, compile, and run code. **1.1.B.2** A *compiler* checks code for some errors. Errors detectable by the compiler need to be fixed before the program can be run.
1.1.C Identify types of programming errors.	**1.1.C.1** A *syntax error* is a mistake in the program where the rules of the programming language are not followed. These errors are detected by the compiler. **1.1.C.2** A *logic error* is a mistake in the algorithm or program that causes it to behave incorrectly or unexpectedly. These errors are detected by testing the program with specific data to see if it produces the expected outcome. **1.1.C.3** A *run-time error* is a mistake in the program that occurs during the execution of a program. Run-time errors typically cause the program to terminate abnormally. **1.1.C.4** An *exception* is a type of run-time error that occurs as a result of an unexpected error that was not detected by the compiler. It interrupts the normal flow of the program's execution.

© 2024 College Board

TOPIC 1.2
Variables and Data Types

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
1.2.A Identify the most appropriate data type category for a particular specification.	**1.2.A.1** A *data type* is a set of values and a corresponding set of operations on those values. Data types can be categorized as either primitive or reference. **1.2.A.2** The *primitive data types* used in this course define the set of values and corresponding operations on those values for numbers and Boolean values. **1.2.A.3** A *reference type* is used to define objects that are not primitive types.
1.2.B Develop code to declare variables to store numbers and Boolean values.	**1.2.B.1** The three primitive data types used in this course are `int`, `double`, and `boolean`. An `int` value is an integer. `A double` value is a real number. `A boolean` value is either `true` or `false`. **EXCLUSION STATEMENT**—*The other five primitive data types* (`long`, `short`, `byte`, `float`, *and* `char`) *are outside the scope of the AP Computer Science A course and exam.* **1.2.B.2** A *variable* is a storage location that holds a value, which can change while the program is running. Every variable has a name and an associated data type. A variable of a primitive type holds a primitive value from that type.

© 2024 College Board

TOPIC 1.3
Expressions and Output

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
1.3.A Develop code to generate output and determine the result that would be displayed.	**1.3.A.1** `System.out.print` and `System.out.println` display information on the computer display. `System.out.println` moves the cursor to a new line after the information has been displayed, while `System.out.print` does not.
1.3.B Develop code to utilize string literals and determine the result of using string literals.	**1.3.B.1** A *literal* is the code representation of a fixed value. **1.3.B.2** A *string literal* is a sequence of characters enclosed in double quotes. **1.3.B.3** *Escape sequences* are special sequences of characters that can be included in a string. They start with a `\` and have a special meaning in Java. Escape sequences used in this course include double quote `\"`, backslash `\\`, and newline `\n`.
1.3.C Develop code for arithmetic expressions and determine the result of these expressions.	**1.3.C.1** *Arithmetic expressions*, which consist of numeric values, variables, and operators, include expressions of type `int` and `double`. **1.3.C.2** The *arithmetic operators* consist of addition `+`, subtraction `-`, multiplication `*`, division `/`, and remainder (modulo) `%`. An arithmetic operation that uses two `int` values will evaluate to an `int` value. An arithmetic operation that uses at least one `double` value will evaluate to a `double` value. **EXCLUSION STATEMENT**—*Expressions that result in special double values (e.g., infinities and NaN) are outside the scope of the AP Computer Science A course and exam.* **1.3.C.3** When dividing numeric values that are both `int` values, the result is only the integer portion of the quotient. When dividing numeric values that use at least one `double` value, the result is the quotient. **1.3.C.4** The remainder (modulo) operator `%` is used to compute the remainder when one number is divided by another number. **EXCLUSION STATEMENT**—*The use of values less than* `0` *for* `a` *and the use of values less than or equal to* `0` *for* `b` *is outside the scope of the AP Computer Science A course and exam.*

© 2024 College Board

UNIT 1

Using Objects and Methods

LEARNING OBJECTIVE

1.3.C

Develop code for arithmetic expressions and determine the result of these expressions.

ESSENTIAL KNOWLEDGE

1.3.C.5

Operators can be used to construct compound expressions. At compile time, numeric values are associated with operators according to operator precedence to determine how they are grouped. Parentheses can be used to modify operator precedence. Multiplication, division, and remainder (modulo) have precedence over addition and subtraction. Operators with the same precedence are evaluated from left to right.

1.3.C.6

An attempt to divide an integer by the integer zero will result in an `ArithmeticException`.

EXCLUSION STATEMENT—*The use of dividing by zero when one numeric value is a* `double` *is outside the scope of the AP Computer Science A course and exam.*

© 2024 College Board

TOPIC 1.4
Assignment Statements and Input

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
1.4.A Develop code for assignment statements with expressions and determine the value that is stored in the variable as a result of these statements.	**1.4.A.1** Every variable must be assigned a value before it can be used in an expression. That value must be from a compatible data type. A variable is initialized the first time it is assigned a value. Reference types can be assigned a new object or `null` if there is no object. The literal `null` is a special value used to indicate that a reference is not associated with any object.
	1.4.A.2 The assignment operator `=` allows a program to initialize or change the value stored in a variable. The value of the expression on the right is stored in the variable on the left.
	EXCLUSION STATEMENT—*The use of assignment operators inside expressions (e.g.,* `a = b = 4; or a[i += 5])` *is outside the scope of the AP Computer Science A course and exam.*
	1.4.A.3 During execution, an expression is evaluated to produce a single value. The value of an expression has a type based on the evaluation of the expression.
1.4.B Develop code to read input.	**1.4.B.1** Input can come in a variety of forms, such as tactile, audio, visual, or text. The `Scanner` class is one way to obtain text input from the keyboard.
	EXCLUSION STATEMENT—*Any specific form of input from the user is outside the scope of the AP Computer Science A course and exam.*

© 2024 College Board

TOPIC 1.5
Casting and Range of Variables

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
1.5.A Develop code to cast primitive values to different primitive types in arithmetic expressions and determine the value that is produced as a result.	**1.5.A.1** The *casting operators* `(int)` and `(double)` can be used to convert from a `double` value to an `int` value (or vice versa). **1.5.A.2** Casting a `double` value to an `int` value causes the digits to the right of the decimal point to be truncated. **1.5.A.3** Some code causes `int` values to be automatically cast (widened) to `double` values. **1.5.A.4** Values of type `double` can be rounded to the nearest integer by `(int)(x + 0.5)` for non-negative numbers or `(int)(x – 0.5)` for negative numbers.
1.5.B Describe conditions when an integer expression evaluates to a value out of range.	**1.5.B.1** The constant `Integer.MAX_VALUE` holds the value of the largest possible `int` value. The constant `Integer.MIN_VALUE` holds the value of the smallest possible `int` value. **1.5.B.2** Integer values in Java are represented by values of type `int`, which are stored using a finite amount (4 bytes) of memory. Therefore, an `int` value must be in the range from `Integer.MIN_VALUE` to `Integer.MAX_VALUE` inclusive. **1.5.B.3** If an expression would evaluate to an `int` value outside of the allowed range, an integer overflow occurs. The result is an `int` value in the allowed range but not necessarily the value expected.
1.5.C Describe conditions that limit accuracy of expressions.	**1.5.C.1** Computers allot a specified amount of memory to store data based on the data type. If an expression would evaluate to a `double` that is more precise than can be stored in the allotted amount of memory, a round-off error occurs. The result will be rounded to the representable value. To avoid rounding errors that naturally occur, use `int` values. **EXCLUSION STATEMENT**—*Other special decimal data types that can be used to avoid rounding errors are outside the scope of the AP Computer Science A course and exam.*

© 2024 College Board

TOPIC 1.6
Compound Assignment Operators

LEARNING OBJECTIVE

1.6.A

Develop code for assignment statements with compound assignment operators and determine the value that is stored in the variable as a result.

ESSENTIAL KNOWLEDGE

1.6.A.1

Compound assignment operators `+=`, `-=`, `*=`, `/=`, and `%=` can be used in place of the assignment operator in numeric expressions. A compound assignment operator performs the indicated arithmetic operation between the value on the left and the value on the right and then assigns the result to the variable on the left.

1.6.A.2

The post-increment operator `++` and post-decrement operator `--` are used to add 1 or subtract 1 from the stored value of a numeric variable. The new value is assigned to the variable.

X EXCLUSION STATEMENT—*The use of increment and decrement operators in prefix form (i.e., `++x`) is outside the scope of the AP Computer Science A course and exam. The use of increment and decrement operators inside other expressions (i.e., `arr[x++]`) is outside the scope of the AP Computer Science A course and exam.*

© 2024 College Board

TOPIC 1.7

Application Program Interface (API) and Libraries

LEARNING OBJECTIVE

1.7.A

Identify the attributes and behaviors of a class found in the libraries contained in an API.

ESSENTIAL KNOWLEDGE

1.7.A.1

Libraries are collections of classes. An *Application Programming Interface (API)* specification informs the programmer how to use those classes. Documentation found in API specifications and libraries is essential to understanding the attributes and behaviors of a class defined by the API. A *class* defines a specific reference type. Classes in the APIs and libraries are grouped into packages. Existing classes and class libraries can be utilized to create objects.

1.7.A.2

Attributes refer to the data related to the class and are stored in variables. *Behaviors* refer to what instances of the class can do (or what can be done with it) and are defined by methods.

© 2024 College Board

TOPIC 1.8
Documentation with Comments

LEARNING OBJECTIVE

1.8.A

Describe the functionality and use of code through comments.

ESSENTIAL KNOWLEDGE

1.8.A.1

Comments are written for both the original programmer and other programmers to understand the code and its functionality, but are ignored by the compiler and are not executed when the program is run. Three types of comments in Java include `/* */`, which generates a block of comments; `//`, which generates a comment on one line; and `/** */`, which are Javadoc comments and are used to create API documentation.

1.8.A.2

A *precondition* is a condition that must be true just prior to the execution of a method in order for it to behave as expected. There is no expectation that the method will check to ensure preconditions are satisfied.

1.8.A.3

A *postcondition* is a condition that must always be true after the execution of a method. Postconditions describe the outcome of the execution in terms of what is being returned or the current value of the attributes of an object.

© 2024 College Board

TOPIC 1.9
Method Signatures

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
1.9.A Identify the correct method to call based on documentation and method signatures.	**1.9.A.1** A *method* is a named block of code that only runs when it is called. A *block of code* is any section of code that is enclosed in braces. *Procedural abstraction* allows a programmer to use a method by knowing what the method does even if they do not know how the method was written. **1.9.A.2** A *parameter* is a variable declared in the header of a method or constructor and can be used inside the body of the method. This allows values or arguments to be passed and used by a method or constructor. A *method signature* for a method with parameters consists of the method name and the ordered list of parameter types. A method signature for a method without parameters consists of the method name and an empty parameter list.
1.9.B Describe how to call methods.	**1.9.B.1** A *void method* does not have a return value and is therefore not called as part of an expression. **1.9.B.2** A *non-void method* returns a value that is the same type as the return type in the header. To use the return value when calling a non-void method, it must be stored in a variable or used as part of an expression. **1.9.B.3** An *argument* is a value that is passed into a method when the method is called. The arguments passed to a method must be compatible in number and order with the types identified in the parameter list of the method signature. When calling methods, arguments are passed using call by value. *Call by value* initializes the parameters with copies of the arguments. **1.9.B.4** Methods are said to be *overloaded* when there are multiple methods with the same name but different signatures. **1.9.B.5** A *method call* interrupts the sequential execution of statements, causing the program to first execute the statements in the method before continuing. Once the last statement in the method has been executed or a return statement is executed, the flow of control is returned to the point immediately following where the method was called.

© 2024 College Board

TOPIC 1.10
Calling Class Methods

LEARNING OBJECTIVE

1.10.A

Develop code to call class methods and determine the result of those calls.

ESSENTIAL KNOWLEDGE

1.10.A.1

Class methods are associated with the class, not instances of the class. Class methods include the keyword `static` in the header before the method name.

1.10.A.2

Class methods are typically called using the class name along with the dot operator. When the method call occurs in the defining class, the use of the class name is optional in the call.

© 2024 College Board

TOPIC 1.11
Math Class

LEARNING OBJECTIVE

1.11.A

Develop code to write expressions that incorporate calls to built-in mathematical libraries and determine the value that is produced as a result.

ESSENTIAL KNOWLEDGE

1.11.A.1

The `Math` class is part of the `java.lang` package. Classes in the `java.lang` package are available by default.

1.11.A.2

The `Math` class contains only class methods. The following `Math` class methods—including what they do and when they are used—are part of the AP Java Quick Reference:

- `static int abs(int x)` returns the absolute value of an `int` value.
- `static double abs(double x)` returns the absolute value of a `double` value.
- `static double pow(double base, double exponent)` returns the value of the first parameter raised to the power of the second parameter.
- `static double sqrt(double x)` returns the positive square root of a `double` value.
- `static double random()` returns a `double` value greater than or equal to `0.0` and less than `1.0`.

1.11.A.3

The values returned from `Math.random()` can be manipulated using arithmetic and casting operators to produce a random `int` or `double` in a defined range based on specified criteria. Each endpoint of the range can be *inclusive*, meaning the value is included, or *exclusive*, meaning the value is not included.

© 2024 College Board

TOPIC 1.12
Objects: Instances of Classes

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
1.12.A Explain the relationship between a class and an object.	**1.12.A.1** An *object* is a specific instance of a class with defined attributes. A *class* is the formal implementation, or blueprint, of the attributes and behaviors of an object.
	1.12.A.2 A class hierarchy can be developed by putting common attributes and behaviors of related classes into a single class called a *superclass*. Classes that extend a superclass, called *subclasses*, can draw upon the existing attributes and behaviors of the superclass without replacing these in the code. This creates an *inheritance relationship* from the subclasses to the superclass.
	X EXCLUSION STATEMENT—*Designing and implementing inheritance relationships are outside the scope of the AP Computer Science A course and exam.*
	1.12.A.3 All classes in Java are subclasses of the `Object` class.
1.12.B Develop code to declare variables to store reference types.	**1.12.B.1** A variable of a reference type holds an object reference, which can be thought of as the memory address of that object.

© 2024 College Board

TOPIC 1.13
Object Creation and Storage (Instantiation)

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
1.13.A Identify, using its signature, the correct constructor being called.	**1.13.A.1** A class contains *constructors* that are called to create objects. They have the same name as the class. **1.13.A.2** A *constructor signature* consists of the constructor's name, which is the same as the class name, and the ordered list of parameter types. The parameter list, in the header of a constructor, lists the types of the values that are passed and their variable names. **1.13.A.3** Constructors are said to be overloaded when there are multiple constructors with different signatures.
1.13.B Develop code to declare variables of the correct types to hold object references.	**1.13.B.1** A variable of a reference type holds an object reference or, if there is no object, `null`.
1.13.C Develop code to create an object by calling a constructor.	**1.13.C.1** An object is typically created using the keyword `new` followed by a call to one of the class's constructors. **1.13.C.2** Parameters allow constructors to accept values to establish the initial values of the attributes of the object. **1.13.C.3** A *constructor argument* is a value that is passed into a constructor when the constructor is called. The arguments passed to a constructor must be compatible in order and number with the types identified in the parameter list in the constructor signature. When calling constructors, arguments are passed using call by value. Call by value initializes the parameters with copies of the arguments. **1.13.C.4** A constructor call interrupts the sequential execution of statements, causing the program to first execute the statements in the constructor before continuing. Once the last statement in the constructor has been executed, the flow of control is returned to the point immediately following where the constructor was called.

© 2024 College Board

TOPIC 1.14
Calling Instance Methods

LEARNING OBJECTIVE

1.14.A

Develop code to call instance methods and determine the result of these calls.

ESSENTIAL KNOWLEDGE

1.14.A.1

Instance methods are called on objects of the class. The dot operator is used along with the object name to call instance methods.

1.14.A.2

A method call on a `null` reference will result in a `NullPointerException`.

© 2024 College Board

TOPIC 1.15
`String` Manipulation

LEARNING OBJECTIVE

1.15.A
Develop code to create string objects and determine the result of creating and combining strings.

ESSENTIAL KNOWLEDGE

1.15.A.1
A `String` object represents a sequence of characters and can be created by using a string literal.

1.15.A.2
The `String` class is part of the `java.lang` package. Classes in the `java.lang` package are available by default.

1.15.A.3
A `String` object is immutable, meaning once a `String` object is created, its attributes cannot be changed. Methods called on a `String` object do not change the content of the `String` object.

1.15.A.4
Two `String` objects can be concatenated together or combined using the `+` or `+=` operator, resulting in a new `String` object. A primitive value can be concatenated with a `String` object. This causes the implicit conversion of the primitive value to a `String` object.

1.15.A.5
A `String` object can be concatenated with any object, which implicitly calls the object's `toString` method (a behavior which is guaranteed to exist by the inheritance relationship every class has with the `Object` class). An object's `toString` method returns a string value representing the object. Subclasses of `Object` often override the `toString` method with class-specific implementation. *Method overriding* occurs when a public method in a subclass has the same method signature as a public method in the superclass, but the behavior of the method is specific to the subclass.

EXCLUSION STATEMENT—*Overriding the* `toString` *method of a class is outside the scope of the AP Computer Science A course and exam.*

1.15.B
Develop code to call methods on string objects and determine the result of calling these methods.

1.15.B.1
A `String` object has index values from `0` to one less than the length of the string. Attempting to access indices outside this range will result in an `IndexOutOfBoundsException`.

© 2024 College Board

LEARNING OBJECTIVE

1.15.B

Develop code to call methods on string objects and determine the result of calling these methods.

ESSENTIAL KNOWLEDGE

1.15.B.2

The following `String` methods—including what they do and when they are used—are part of the AP Java Quick Reference:

- `int length()` returns the number of characters in a `String` object.
- `String substring(int from, int to)` returns the substring beginning at index `from` and ending at index `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.
- `boolean equals(Object other)` returns `true` if `this` corresponds to the same sequence of characters as `other`; returns `false` otherwise.
- `int compareTo(String other)` returns a value `< 0` if `this` is less than `other`; returns zero if `this` is equal to `other`; returns a value `> 0` if `this` is greater than `other`. Strings are ordered based upon the alphabet.

EXCLUSION STATEMENT—*Using the* `equals` *method to compare one* `String` *object with an object of a type other than* `String` *is outside the scope of the AP Computer Science A course and exam.*

1.15.B.3

A string identical to the single element substring at position `index` can be created by calling `substring(index, index + 1)`.

© 2024 College Board

THIS PAGE IS INTENTIONALLY LEFT BLANK.

AP COMPUTER SCIENCE A

UNIT 2

Selection and Iteration

© 2024 College Board

THIS PAGE IS INTENTIONALLY LEFT BLANK.

TOPIC 2.1

Algorithms with Selection and Repetition

LEARNING OBJECTIVE

2.1.A

Represent patterns and algorithms that involve selection and repetition found in everyday life using written language or diagrams.

ESSENTIAL KNOWLEDGE

2.1.A.1

The building blocks of algorithms include sequencing, selection, and repetition.

2.1.A.2

Algorithms can contain selection, through decision making, and repetition, via looping.

2.1.A.3

Selection occurs when a choice of how the execution of an algorithm will proceed is based on a true or false decision.

2.1.A.4

Repetition is when a process repeats itself until a desired outcome is reached.

2.1.A.5

The order in which sequencing, selection, and repetition are used contributes to the outcome of the algorithm.

© 2024 College Board

TOPIC 2.2
Boolean Expressions

LEARNING OBJECTIVE

2.2.A

Develop code to create Boolean expressions with relational operators and determine the result of these expressions.

ESSENTIAL KNOWLEDGE

2.2.A.1

Values can be compared using the *relational operators* `==` and `!=` to determine whether the values are the same. With primitive types, this compares the actual primitive values. With reference types, this compares the object references.

2.2.A.2

Numeric values can be compared using the relational operators `<`, `>`, `<=`, and `>=` to determine the relationship between the values.

2.2.A.3

An expression involving relational operators evaluates to a Boolean value.

© 2024 College Board

TOPIC 2.3
`if` Statements

LEARNING OBJECTIVE

2.3.A

Develop code to represent branching logical processes by using selection statements and determine the result of these processes.

ESSENTIAL KNOWLEDGE

2.3.A.1

Selection statements change the sequential execution of statements.

2.3.A.2

An `if` statement is a type of selection statement that affects the flow of control by executing different segments of code based on the value of a Boolean expression.

2.3.A.3

A *one-way selection* (`if` statement) is used when there is a segment of code to execute under a certain condition. In this case, the body is executed only when the Boolean expression is `true`.

2.3.A.4

A *two-way selection* (`if-else` statement) is used when there are two segments of code—one to be executed when the Boolean expression is `true` and another segment for when the Boolean expression is `false`. In this case, the body of the `if` is executed when the Boolean expression is `true`, and the body of the `else` is executed when the Boolean expression is `false`.

© 2024 College Board

TOPIC 2.4

Nested `if` Statements

LEARNING OBJECTIVE

2.4.A

Develop code to represent nested branching logical processes and determine the result of these processes.

ESSENTIAL KNOWLEDGE

2.4.A.1

Nested if statements consist of `if`, `if-else`, or `if-else-if` statements within `if`, `if-else`, or `if-else-if` statements.

2.4.A.2

The Boolean expression of the inner nested `if` statement is evaluated only if the Boolean expression of the outer `if` statement evaluates to `true`.

2.4.A.3

A *multi-way selection* (`if-else-if`) is used when there are a series of expressions with different segments of code for each condition. Multi-way selection is performed such that no more than one segment of code is executed based on the first expression that evaluates to `true`. If no expression evaluates to `true` and there is a trailing `else` statement, then the body of the `else` is executed.

© 2024 College Board

TOPIC 2.5
Compound Boolean Expressions

LEARNING OBJECTIVE

2.5.A

Develop code to represent compound Boolean expressions and determine the result of these expressions.

ESSENTIAL KNOWLEDGE

2.5.A.1

Logical operators `!` (not), `&&` (and), and `||` (or) are used with Boolean expressions. The expression `!a` evaluates to `true` if `a` is `false` and evaluates to `false` otherwise. The expression `a && b` evaluates to `true` if both `a` and `b` are `true` and evaluates to `false` otherwise. The expression `a || b` evaluates to `true` if `a` is `true`, `b` is `true`, or both, and evaluates to `false` otherwise. The order of precedence for evaluating logical operators is `!` (not), `&&` (and), then `||` (or). An expression involving logical operators evaluates to a Boolean value.

2.5.A.2

Short-circuit evaluation occurs when the result of a logical operation using `&&` or `||` can be determined by evaluating only the first Boolean expression. In this case, the second Boolean expression is not evaluated.

© 2024 College Board

TOPIC 2.6
Comparing Boolean Expressions

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
2.6.A Compare equivalent Boolean expressions.	**2.6.A.1** Two Boolean expressions are *equivalent* if they evaluate to the same value in all cases. Truth tables can be used to prove Boolean expressions are equivalent. **2.6.A.2** *De Morgan's law* can be applied to Boolean expressions to create equivalent Boolean expressions. Under De Morgan's law, the Boolean expression `!(a && b)` is equivalent to `!a || !b` and the Boolean expression `!(a || b)` is equivalent to `!a && !b`.
2.6.B Develop code to compare object references using Boolean expressions and determine the result of these expressions.	**2.6.B.1** Two different variables can hold references to the same object. Object references can be compared using `==` and `!=`. **2.6.B.2** An object reference can be compared with `null`, using `==` or `!=`, to determine if the reference actually references an object. **2.6.B.3** Classes often define their own `equals` method, which can be used to specify the criteria for equivalency for two objects of the class. The equivalency of two objects is most often determined using attributes from the two objects.

EXCLUSION STATEMENT—*Overriding the* `equals` *method is outside the scope of the AP Computer Science A course and exam.*

© 2024 College Board

TOPIC 2.7
`while` Loops

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
2.7.A Identify when an iterative process is required to achieve a desired result.	**2.7.A.1** *Iteration* is a form of repetition. Iteration statements change the flow of control by repeating a segment of code zero or more times as long as the Boolean expression controlling the loop evaluates to `true`.
	2.7.A.2 An *infinite loop* occurs when the Boolean expression in an iterative statement always evaluates to `true`.
	2.7.A.3 The loop body of an iterative statement will not execute if the Boolean expression initially evaluates to `false`.
	2.7.A.4 *Off by one* errors occur when the iteration statement loops one time too many or one time too few.
2.7.B Develop code to represent iterative processes using `while` loops and determine the result of these processes.	**2.7.B.1** A `while` loop is a type of iterative statement. In `while` loops, the Boolean expression is evaluated before each iteration of the loop body, including the first. When the expression evaluates to `true`, the loop body is executed. This continues until the Boolean expression evaluates to `false`, whereupon the iteration terminates.

© 2024 College Board

TOPIC 2.8
`for` Loops

LEARNING OBJECTIVE

2.8.A

Develop code to represent iterative processes using `for` loops and determine the result of these processes.

ESSENTIAL KNOWLEDGE

2.8.A.1

A `for` loop is a type of iterative statement. There are three parts in a `for` loop header: the initialization, the Boolean expression, and the update.

2.8.A.2

In a `for` loop, the initialization statement is only executed once before the first Boolean expression evaluation. The variable being initialized is referred to as a *loop control variable*. The Boolean expression is evaluated immediately after the loop control variable is initialized and then followed by each execution of the increment statement until it is `false`. In each iteration, the update is executed after the entire loop body is executed and before the Boolean expression is evaluated again.

2.8.A.3

A `for` loop can be rewritten into an equivalent `while` loop (and vice versa).

© 2024 College Board

TOPIC 2.9
Implementing Selection and Iteration Algorithms

LEARNING OBJECTIVE

2.9.A

Develop code for standard and original algorithms (without data structures) and determine the result of these algorithms.

ESSENTIAL KNOWLEDGE

2.9.A.1

There are standard algorithms to:

- identify if an integer is or is not evenly divisible by another integer
- identify the individual digits in an integer
- determine the frequency with which a specific criterion is met
- determine a minimum or maximum value
- compute a sum or average

© 2024 College Board

TOPIC 2.10
Implementing String Algorithms

LEARNING OBJECTIVE

2.10.A

Develop code for standard and original algorithms that involve strings and determine the result of these algorithms.

ESSENTIAL KNOWLEDGE

2.10.A.1

There are standard string algorithms to:

- find if one or more substrings have a particular property
- determine the number of substrings that meet specific criteria
- create a new string with the characters reversed

© 2024 College Board

TOPIC 2.11
Nested Iteration

LEARNING OBJECTIVE

2.11.A

Develop code to represent nested iterative processes and determine the result of these processes.

ESSENTIAL KNOWLEDGE

2.11.A.1

Nested iteration statements are iteration statements that appear in the body of another iteration statement. When a loop is nested inside another loop, the inner loop must complete all its iterations before the outer loop can continue to its next iteration.

© 2024 College Board

TOPIC 2.12

Informal Run-Time Analysis

LEARNING OBJECTIVE

2.12.A

Calculate statement execution counts and informal run-time comparison of iterative statements.

ESSENTIAL KNOWLEDGE

2.12.A.1

A *statement execution count* indicates the number of times a statement is executed by the program. Statement execution counts are often calculated informally through tracing and analysis of the iterative statements.

© 2024 College Board

AP COMPUTER SCIENCE A

UNIT 3

Class Creation

© 2024 College Board

THIS PAGE IS INTENTIONALLY LEFT BLANK.

TOPIC 3.1
Abstraction and Program Design

LEARNING OBJECTIVE

3.1.A

Represent the design of a program by creating diagrams that indicate the classes in the program and the data and procedural abstractions found in each class by including all attributes and behaviors.

ESSENTIAL KNOWLEDGE

3.1.A.1

Abstraction is the process of reducing complexity by focusing on the main idea. By hiding details irrelevant to the question at hand and bringing together related and useful details, abstraction reduces complexity and allows one to focus on the idea.

3.1.A.2

Data abstraction provides a separation between the abstract properties of a data type and the concrete details of its representation. Data abstraction manages complexity by giving data a name without referencing the specific details of the representation. Data can take the form of a single variable or a collection of data, such as in a class or a set of data.

3.1.A.3

An *attribute* is a type of data abstraction that is defined in a class outside any method or constructor. An *instance variable* is an attribute whose value is unique to each instance of the class. A *class variable* is an attribute shared by all instances of the class.

3.1.A.4

Procedural abstraction provides a name for a process and allows a method to be used only knowing what it does, not how it does it. Through *method decomposition*, a programmer breaks down larger behaviors of the class into smaller behaviors by creating methods to represent each individual smaller behavior. A procedural abstraction may extract shared features to generalize functionality instead of duplicating code. This allows for code reuse, which helps manage complexity.

3.1.A.5

Using parameters allows procedures to be generalized, enabling the procedures to be reused with a range of input values or arguments.

3.1.A.6

Using procedural abstraction in a program allows programmers to change the internals of a method (to make it faster, more efficient, use less storage, etc.) without needing to notify method users of the change as long as the method signature and what the method does is preserved.

3.1.A.7

Prior to implementing a class, it is helpful to take time to design each class including its attributes and behaviors. This design can be represented using natural language or diagrams.

© 2024 College Board

TOPIC 3.2
Impact of Program Design

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
3.2.A Explain the social and ethical implications of computing systems.	**3.2.A.1** *System reliability* refers to the program being able to perform its tasks as expected under stated conditions without failure. Programmers should make an effort to maximize system reliability by testing the program with a variety of conditions. **3.2.A.2** The creation of programs has impacts on society, economies, and culture. These impacts can be both beneficial and harmful. Programs meant to fill a need or solve a problem can have unintended harmful effects beyond their intended use. **3.2.A.3** Legal issues and intellectual property concerns arise when creating programs. Programmers often reuse code written by others and published as open source and free to use. Incorporation of code that is not published as open source requires the programmer to obtain permission and often purchase the code before integrating it into their program.
3.2.B Describe the consequences of using artificial intelligence to develop code.	**3.2.B.1** *Artificial intelligence* is the simulation of intelligent behavior by a computer system or algorithm. The use of artificial intelligence when developing code may decrease the amount of time it takes to develop code. **3.2.B.2** The use of artificial intelligence when developing code may result in incomplete programs or programs that do not work as intended. In such cases, programmers will often need to make necessary changes manually to correct the program. **3.2.B.3** A well-thought-out and thoroughly articulated program design may increase the usefulness of artificial intelligence when developing code that works as intended. **3.2.B.4** The use of artificial intelligence in developing code has the potential to create programs that contain various types of bias, including bias toward different groups of people.

© 2024 College Board

TOPIC 3.3

Anatomy of a Class

LEARNING OBJECTIVE

3.3.A

Develop code to designate access and visibility constraints to classes, data, constructors, and methods.

ESSENTIAL KNOWLEDGE

3.3.A.1

Data encapsulation is a technique in which the implementation details of a class are kept hidden from external classes. The keywords `public` and `private` affect the access of classes, data, constructors, and methods. The keyword `private` restricts access to the declaring class, while the keyword `public` allows access from classes outside the declaring class.

3.3.A.2

In this course, classes are always designated `public` and are declared with the keyword `class`.

3.3.A.3

In this course, constructors are always designated `public`.

3.3.A.4

Instance variables belong to the object, and each object has its own copy of the variable.

3.3.A.5

Access to attributes should be kept internal to the class in order to accomplish encapsulation. Therefore, it is good programming practice to designate the instance variables for these attributes as `private` unless the class specification states otherwise.

3.3.A.6

Access to behaviors can be internal or external to the class. Methods designated as `public` can be accessed internally or externally to a class whereas methods designated as `private` can only be accessed internally to the class.

© 2024 College Board

TOPIC 3.4
Constructors

LEARNING OBJECTIVE

3.4.A

Develop code to declare instance variables for the attributes to be initialized in the body of the constructors of a class.

ESSENTIAL KNOWLEDGE

3.4.A.1

An object's *state* refers to its attributes and their values at a given time and is defined by instance variables belonging to the object. This defines a *has-a* relationship between the object and its instance variables.

3.4.A.2

A constructor is used to set the initial state of an object, which should include initial values for all instance variables. When a constructor is called, memory is allocated for the object and the associated object reference is returned. Constructor parameters, if specified, provide data to initialize instance variables.

3.4.A.3

When a mutable object is a constructor parameter, the instance variable should be initialized with a copy of the referenced object. In this way, the instance variable does not hold a reference to the original object, and methods are prevented from modifying the state of the original object.

3.4.A.4

When no constructor is written, Java provides a no-parameter constructor, and the instance variables are set to default values according to the data type of the attribute. This constructor is called the *default constructor*.

3.4.A.5

The default value for an attribute of type `int` is `0`. The default value of an attribute of type `double` is `0.0`. The default value of an attribute of type `boolean` is `false`. The default value of a reference type is `null`.

© 2024 College Board

TOPIC 3.5
Methods: How to Write Them

LEARNING OBJECTIVE

3.5.A

Develop code to define behaviors of an object through methods written in a class using primitive values and determine the result of calling these methods.

ESSENTIAL KNOWLEDGE

3.5.A.1

A `void` method does not return a value. Its header contains the keyword `void` before the method name.

3.5.A.2

A non-void method returns a single value. Its header includes the return type in place of the keyword `void`.

3.5.A.3

In non-void methods, a return expression compatible with the return type is evaluated, and the value is returned. This is referred to as *return by value*.

3.5.A.4

The `return` keyword is used to return the flow of control to the point where the method or constructor was called. Any code that is sequentially after a return statement will never be executed. Executing a return statement inside a selection or iteration statement will halt the statement and exit the method or constructor.

3.5.A.5

An *accessor method* allows objects of other classes to obtain a copy of the value of instance variables or class variables. An accessor method is a non-void method.

3.5.A.6

A *mutator (modifier) method* is a method that changes the values of the instance variables or class variables. A mutator method is often a void method.

3.5.A.7

Methods with parameters receive values through those parameters and use those values in accomplishing the method's task.

3.5.A.8

When an argument is a primitive value, the parameter is initialized with a copy of that value. Changes to the parameter have no effect on the corresponding argument.

© 2024 College Board

TOPIC 3.6

Methods: Passing and Returning References of an Object

LEARNING OBJECTIVE

3.6.A

Develop code to define behaviors of an object through methods written in a class using object references and determine the result of calling these methods.

ESSENTIAL KNOWLEDGE

3.6.A.1

When an argument is an object reference, the parameter is initialized with a copy of that reference; it does not create a new independent copy of the object. If the parameter refers to a mutable object, the method or constructor can use this reference to alter the state of the object. It is good programming practice to not modify mutable objects that are passed as parameters unless required in the specification.

3.6.A.2

When the return expression evaluates to an object reference, the reference is returned, not a reference to a new copy of the object.

3.6.A.3

Methods cannot access the private data and methods of a parameter that holds a reference to an object unless the parameter is the same type as the method's enclosing class.

© 2024 College Board

TOPIC 3.7
Class Variables and Methods

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
3.7.A Develop code to define behaviors of a class through class methods.	**3.7.A.1** Class methods cannot access or change the values of instance variables or call instance methods without being passed an instance of the class via a parameter.
	3.7.A.2 Class methods can access or change the values of class variables and can call other class methods.
3.7.B Develop code to declare the class variables that belong to the class.	**3.7.B.1** Class variables belong to the class, with all objects of a class sharing a single copy of the class variable. Class variables are designated with the `static` keyword before the variable type.
	3.7.B.2 Class variables that are designated `public` are accessed outside of the class by using the class name and the dot operator, since they are associated with a class, not objects of a class.
	3.7.B.3 When a variable is declared `final`, its value cannot be modified.

© 2024 College Board

TOPIC 3.8

Scope and Access

LEARNING OBJECTIVE

3.8.A

Explain where variables can be used in the code.

ESSENTIAL KNOWLEDGE

3.8.A.1

Local variables are variables declared in the headers or bodies of blocks of code. Local variables can only be accessed in the block in which they are declared. Since constructors and methods are blocks of code, parameters to constructors or methods are also considered local variables. These variables may only be used within the constructor or method and cannot be declared to be `public` or `private`.

3.8.A.2

When there is a local variable or parameter with the same name as an instance variable, the variable name will refer to the local variable instead of the instance variable within the body of the constructor or method.

© 2024 College Board

TOPIC 3.9
`this` Keyword

LEARNING OBJECTIVE

3.9.A

Develop code for expressions that are self-referencing and determine the result of these expressions.

ESSENTIAL KNOWLEDGE

3.9.A.1

Within an instance method or a constructor, the keyword `this` acts as a special variable that holds a reference to the current object—the object whose method or constructor is being called.

3.9.A.2

The keyword `this` can be used to pass the current object as an argument in a method call.

3.9.A.3

Class methods do not have a `this` reference.

© 2024 College Board

THIS PAGE IS INTENTIONALLY LEFT BLANK.

AP COMPUTER SCIENCE A

UNIT 4

Data Collections

© 2024 College Board

THIS PAGE IS INTENTIONALLY LEFT BLANK.

TOPIC 4.1
Ethical and Social Issues Around Data Collection

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
4.1.A Explain the risks to privacy from collecting and storing personal data on computer systems.	**4.1.A.1** When using a computer, personal privacy is at risk. When developing new programs, programmers should attempt to safeguard the personal privacy of the user. Personal data can be kept secure by ensuring that attributes are encapsulated and unable to be accessed or modified outside the class.
4.1.B Explain the importance of recognizing data quality and potential issues when using a data set.	**4.1.B.1** *Algorithmic bias* describes systemic and repeated errors in a program that create unfair outcomes for a specific group of users. **4.1.B.2** Programmers should be aware of the data set collection method and the potential for bias when using this method before using the data to extrapolate new information or drawing conclusions. **4.1.B.3** Some data sets are incomplete or contain inaccurate data. Using such data in the development or use of a program can cause the program to work incorrectly or inefficiently.
4.1.C Identify an appropriate data set to use in order to solve a problem or answer a specific question.	**4.1.C.1** Contents of a data set might be related to a specific question or topic and might not be appropriate to give correct answers or extrapolate information for a different question or topic.

© 2024 College Board

TOPIC 4.2
Introduction to Using Data Sets

LEARNING OBJECTIVE

4.2.A

Represent patterns and algorithms that involve data sets found in everyday life using written language or diagrams.

ESSENTIAL KNOWLEDGE

4.2.A.1

Data sets are a collection of specific pieces of information or data.

4.2.A.2

Data sets can be manipulated and analyzed to solve a problem or answer a question. When analyzing data sets, values within the set are accessed and utilized one at a time and then processed according to the desired outcome.

4.2.A.3

Data can be represented in a diagram by using a chart or table. This visual can be used to plan the algorithm that will be used to manipulate the data.

© 2024 College Board

TOPIC 4.3
Array Creation and Access

LEARNING OBJECTIVE

4.3.A

Develop code used to represent collections of related data using one-dimensional (1D) array objects.

ESSENTIAL KNOWLEDGE

4.3.A.1

An *array* stores multiple values of the same type. The values can be either primitive values or object references.

4.3.A.2

The length of an array is established at the time of creation and cannot be changed. The length of an array can be accessed through the `length` attribute.

4.3.A.3

When an array is created using the keyword new, all of its elements are initialized to the default values for the element data type. The default value for `int` is `0`, for `double` is `0.0`, for `boolean` is `false`, and for a reference type is `null`.

4.3.A.4

Initializer lists can be used to create and initialize arrays.

4.3.A.5

Square brackets `[]` are used to access and modify an element in a 1D array using an index.

4.3.A.6

The valid index values for an array are `0` through one less than the length of the array, inclusive. Using an index value outside of this range will result in an `ArrayIndexOutOfBoundsException`.

© 2024 College Board

TOPIC 4.4
Array Traversals

LEARNING OBJECTIVE

4.4.A

Develop code used to traverse the elements in a 1D array and determine the result of these traversals.

ESSENTIAL KNOWLEDGE

4.4.A.1

Traversing an array is when repetition statements are used to access all or an ordered sequence of elements in an array.

4.4.A.2

Traversing an array with an indexed `for` loop or `while` loop requires elements to be accessed using their indices.

4.4.A.3

An enhanced `for` loop header includes a variable, referred to as the enhanced `for` loop variable. For each iteration of the enhanced `for` loop, the enhanced `for` loop variable is assigned a copy of an element without using its index.

4.4.A.4

Assigning a new value to the enhanced `for` loop variable does not change the value stored in the array.

4.4.A.5

When an array stores object references, the attributes can be modified by calling methods on the enhanced `for` loop variable. This does not change the object references stored in the array.

4.4.A.6

Code written using an enhanced `for` loop to traverse elements in an array can be rewritten using an indexed `for` loop or a `while` loop.

© 2024 College Board

TOPIC 4.5
Implementing Array Algorithms

LEARNING OBJECTIVE

4.5.A

Develop code for standard and original algorithms for a particular context or specification that involves arrays and determine the result of these algorithms.

ESSENTIAL KNOWLEDGE

4.5.A.1

There are standard algorithms that utilize array traversals to:

- determine a minimum or maximum value
- compute a sum or average
- determine if at least one element has a particular property
- determine if all elements have a particular property
- determine the number of elements having a particular property
- access all consecutive pairs of elements
- determine the presence or absence of duplicate elements
- shift or rotate elements left or right
- reverse the order of the elements

© 2024 College Board

TOPIC 4.6
Using Text Files

LEARNING OBJECTIVE

4.6.A

Develop code to read data from a text file.

ESSENTIAL KNOWLEDGE

4.6.A.1

A *file* is storage for data that persists when the program is not running. The data in a file can be retrieved during program execution.

4.6.A.2

A file can be connected to the program using the `File` and `Scanner` classes.

4.6.A.3

A file can be opened by creating a `File` object, using the name of the file as the argument of the constructor.

- `File(String str)` is the `File` constructor that accepts a `String` file name to open for reading, where `str` is the pathname for the file.

4.6.A.4

When using the `File` class, it is required to indicate what to do if the file with the provided name cannot be opened. One way to accomplish this is to add `throws IOException` to the header of the method that uses the file. If the file name is invalid, the program will terminate.

4.6.A.5

The `File` and `IOException` classes are part of the `java.io` package. An `import` statement must be used to make these classes available for use in the program.

4.6.A.6

The following `Scanner` methods and constructor—including what they do and when they are used—are part of the AP Java Quick Reference:

- `Scanner(File f)` is the `Scanner` constructor that accepts a `File` for reading.
- `int nextInt()` returns the next `int` read from the file or input source if available. If the next `int` does not exist or is out of range, it will result in an `InputMismatchException`.
- `double nextDouble()` returns the next `double` read from the file or input source. If the next `double` does not exist, it will result in an `InputMismatchException`.
- `boolean nextBoolean()` returns the next `boolean` read from the file or input source. If the next `boolean` does not exist, it will result in an `InputMismatchException`.
- `String nextLine()` returns the next line of text as a `String` read from the file or input source; returns the empty string if called immediately after another `Scanner` method that is reading from the file or input source.
- `String next()` returns the next `String` read from the file or input source.

© 2024 College Board

LEARNING OBJECTIVE

4.6.A

Develop code to read data from a text file.

ESSENTIAL KNOWLEDGE

- `boolean hasNext()` returns `true` if there is a next item to read in the file or input source; returns `false` otherwise.
- `void close()` closes this scanner.

EXCLUSION STATEMENT—*Accepting input from the keyboard is outside the scope of the AP Computer Science A course and exam.*

4.6.A.7

Using `nextLine` and the other `Scanner` methods together on the same input source sometimes requires code to adjust for the methods' different ways of handling whitespace.

EXCLUSION STATEMENT—*Writing or analyzing code that uses both* `nextLine` *and other* `Scanner` *methods on the same input source is outside the scope of the AP Computer Science A course and exam.*

4.6.A.8

The following additional `String` method—including what it does and when it is used—is part of the AP Java Quick Reference:

- `String[] split(String del)` returns a `String` array where each element is a substring of `this String`, which has been split around matches of the given expression `del`.

4.6.A.9

A `while` loop can be used to detect if the file still contains elements to read by using the `hasNext` method as the condition of the loop.

4.6.A.10

A file should be closed when the program is finished using it. The `close` method from `Scanner` is called to close the file.

© 2024 College Board

TOPIC 4.7
Wrapper Classes

LEARNING OBJECTIVE

4.7.A

Develop code to use `Integer` and `Double` objects from their primitive counterparts and determine the result of using these objects.

ESSENTIAL KNOWLEDGE

4.7.A.1

The `Integer` class and `Double` class are part of the `java.lang` package. An `Integer` object is immutable, meaning once an `Integer` object is created, its attributes cannot be changed. A `Double` object is immutable, meaning once a `Double` object is created, its attributes cannot be changed.

4.7.A.2

Autoboxing is the automatic conversion that the Java compiler makes between primitive types and their corresponding object wrapper classes. This includes converting an `int` to an `Integer` and a `double` to a `Double`. The Java compiler applies autoboxing when a primitive value is:

- passed as a parameter to a method that expects an object of the corresponding wrapper class
- assigned to a variable of the corresponding wrapper class

4.7.A.3

Unboxing is the automatic conversion that the Java compiler makes from the wrapper class to the primitive type. This includes converting an `Integer` to an `int` and a `Double` to a `double`. The Java compiler applies unboxing when a wrapper class object is:

- passed as a parameter to a method that expects a value of the corresponding primitive type
- assigned to a variable of the corresponding primitive type

4.7.A.4

The following class `Integer` method—including what it does and when it is used—is part of the AP Java Quick Reference:

- `static int parseInt(String s)` returns the `String` argument as a signed `int`.

4.7.A.5

The following class `Double` method—including what it does and when it is used—is part of the AP Java Quick Reference:

- `static double parseDouble(String s)` returns the `String` argument as a signed `double`.

© 2024 College Board

TOPIC 4.8
`ArrayList` Methods

LEARNING OBJECTIVE

4.8.A

Develop code for collections of related objects using `ArrayList` objects and determine the result of calling methods on these objects.

ESSENTIAL KNOWLEDGE

4.8.A.1

An `ArrayList` object is mutable in size and contains object references.

4.8.A.2

The `ArrayList` constructor `ArrayList()` constructs an empty list.

4.8.A.3

Java allows the generic type `ArrayList<E>`, where the type parameter `E` specifies the type of the elements. When `ArrayList<E>` is specified, the types of the reference parameters and return type when using the `ArrayList` methods are type `E`. `ArrayList<E>` is preferred over `ArrayList`. For example, `ArrayList<String> names = new ArrayList<String>();` allows the compiler to find errors that would otherwise be found at run-time.

4.8.A.4

The `ArrayList` class is part of the `java.util` package. An `import` statement can be used to make this class available for use in the program.

4.8.A.5

The following `ArrayList` methods—including what they do and when they are used—are part of the AP Java Quick Reference:

- `int size()` returns the number of elements in the list.
- `boolean add(E obj)` appends `obj` to end of list; returns `true`.
- `void add(int index, E obj)` inserts `obj` at position `index` (`0 <= index <= size`), moving elements at position `index` and higher to the right (adds 1 to their indices) and adds 1 to size.
- `E get(int index)` returns the element at position `index` in the list.
- `E set(int index, E obj)` replaces the element at position `index` with `obj`; returns the element formerly at position `index`.
- `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 subtracts 1 from size; returns the element formerly at position `index`.

4.8.A.6

The indices for an `ArrayList` start at `0` and end at the number of elements `– 1`.

© 2024 College Board

TOPIC 4.9
`ArrayList` Traversals

LEARNING OBJECTIVE

4.9.A

Develop code used to traverse the elements of an `ArrayList` and determine the results of these traversals.

ESSENTIAL KNOWLEDGE

4.9.A.1

Traversing an `ArrayList` is when iteration or recursive statements are used to access all or an ordered sequence of the elements in an `ArrayList`.

4.9.A.2

Deleting elements during a traversal of an `ArrayList` requires the use of special techniques to avoid skipping elements.

4.9.A.3

Attempting to access an index value outside of its range will result in an `IndexOutOfBoundsException`.

4.9.A.4

Changing the size of an `ArrayList` while traversing it using an enhanced `for` loop can result in a `ConcurrentModifcationException`. Therefore, when using an enhanced `for` loop to traverse an `ArrayList`, you should not add or remove elements.

© 2024 College Board

TOPIC 4.10
Implementing `ArrayList` Algorithms

LEARNING OBJECTIVE

4.10.A

Develop code for standard and original algorithms for a particular context or specification that involve `ArrayList` objects and determine the result of these algorithms.

ESSENTIAL KNOWLEDGE

4.10.A.1

There are standard `ArrayList` algorithms that utilize traversals to:

- determine a minimum or maximum value
- compute a sum or average
- determine if at least one element has a particular property
- determine if all elements have a particular property
- determine the number of elements having a particular property
- access all consecutive pairs of elements
- determine the presence or absence of duplicate elements
- shift or rotate elements left or right
- reverse the order of the elements
- insert elements
- delete elements

4.10.A.2

Some algorithms require multiple `String`, array, or `ArrayList` objects to be traversed simultaneously.

© 2024 College Board

TOPIC 4.11
2D Array Creation and Access

LEARNING OBJECTIVE

4.11.A

Develop code used to represent collections of related data using two-dimensional (2D) array objects.

ESSENTIAL KNOWLEDGE

4.11.A.1

A 2D array is stored as an array of arrays. Therefore, the way 2D arrays are created and indexed is similar to 1D array objects. The size of a 2D array is established at the time of creation and cannot be changed. 2D arrays can store either primitive data or object reference data.

EXCLUSION STATEMENT—*Nonrectangular 2D array objects are outside the scope of the AP Computer Science A course and exam.*

4.11.A.2

When a 2D array is created using the keyword `new`, all of its elements are initialized to the default values for the element data type. The default value for `int` is `0`, for `double` is `0.0`, for `boolean` is `false`, and for a reference type is `null`.

4.11.A.3

The initializer list used to create and initialize a 2D array consists of initializer lists that represent 1D arrays; for example, `int[][] arr2D = { {1, 2, 3}, {4, 5, 6} };`.

4.11.A.4

The square brackets `[row][col]` are used to access and modify an element in a 2D array. For the purposes of the exam, when accessing the element at `arr[first][second]`, the first index is used for rows, the second index is used for columns.

4.11.A.5

A single array that is a row of a 2D array can be accessed using the 2D array name and a single set of square brackets containing the row index.

4.11.A.6

The number of rows contained in a 2D array can be accessed through the `length` attribute. The valid row index values for a 2D array are `0` through one less than the number of rows or the length of the array, inclusive. The number of columns contained in a 2D array can be accessed through the `length` attribute of one of the rows. The valid column index values for a 2D array are `0` through one less than the number of columns or the length of any given row of the array, inclusive. For example, given a 2D array named `values`, the number of rows is `values.length` and the number of columns is `values[0].length`. Using an index value outside of these ranges will result in an `ArrayIndexOutOfBoundsException`.

© 2024 College Board

TOPIC 4.12
2D Array Traversals

LEARNING OBJECTIVE

4.12.A

Develop code used to traverse the elements in a 2D array and determine the result of these traversals.

ESSENTIAL KNOWLEDGE

4.12.A.1

Nested iteration statements are used to traverse and access all or an ordered sequence of elements in a 2D array. Since 2D arrays are stored as arrays of arrays, the way 2D arrays are traversed using `for` loops and enhanced `for` loops is similar to 1D array objects. Nested iteration statements can be written to traverse the 2D array in row-major order, column-major order, or a uniquely defined order. *Row-major order* refers to an ordering of 2D array elements where traversal occurs across each row, whereas *column-major order* traversal occurs down each column.

4.12.A.2

The outer loop of a nested enhanced `for` loop used to traverse a 2D array traverses the rows. Therefore, the enhanced `for` loop variable must be the type of each row, which is a 1D array. The inner loop traverses a single row. Therefore, the inner enhanced `for` loop variable must be the same type as the elements stored in the 1D array. Assigning a new value to the enhanced `for` loop variable does not change the value stored in the array.

© 2024 College Board

TOPIC 4.13

Implementing 2D Array Algorithms

LEARNING OBJECTIVE

4.13.A

Develop code for standard and original algorithms for a particular context or specification that involves 2D arrays and determine the result of these algorithms.

ESSENTIAL KNOWLEDGE

4.13.A.1

There are standard algorithms that utilize 2D array traversals to:

- determine a minimum or maximum value of all the elements or for a designated row, column, or other subsection
- compute a sum or average of all the elements or for a designated row, column, or other subsection
- determine if at least one element has a particular property in the entire 2D array or for a designated row, column, or other subsection
- determine if all elements of the 2D array or a designated row, column, or other subsection have a particular property
- determine the number of elements in the 2D array or in a designated row, column, or other subsection having a particular property
- access all consecutive pairs of elements
- determine the presence or absence of duplicate elements in the 2D array or in a designated row, column, or other subsection
- shift or rotate elements in a row left or right or in a column up or down
- reverse the order of the elements in a row or column

© 2024 College Board

TOPIC 4.14
Searching Algorithms

LEARNING OBJECTIVE

4.14.A

Develop code used for linear search algorithms to search for specific information in a collection and determine the results of executing a search.

ESSENTIAL KNOWLEDGE

4.14.A.1

Linear search algorithms are standard algorithms that check each element in order until the desired value is found or all elements in the array or `ArrayList` have been checked. Linear search algorithms can begin the search process from either end of the array or `ArrayList`.

4.14.A.2

When applying linear search algorithms to 2D arrays, each row must be accessed then linear search applied to each row of the 2D array.

© 2024 College Board

TOPIC 4.15
Sorting Algorithms

LEARNING OBJECTIVE

4.15.A

Determine the result of executing each step of sorting algorithms to sort the elements of a collection.

ESSENTIAL KNOWLEDGE

4.15.A.1

Selection sort and insertion sort are iterative sorting algorithms that can be used to sort elements in an array or `ArrayList`.

4.15.A.2

Selection sort repeatedly selects the smallest (or largest) element from the unsorted portion of the list and swaps it into its correct (and final) position in the sorted portion of the list.

4.15.A.3

Insertion sort inserts an element from the unsorted portion of a list into its correct (but not necessarily final) position in the sorted portion of the list by shifting elements of the sorted portion to make room for the new element.

© 2024 College Board

TOPIC 4.16
Recursion

LEARNING OBJECTIVE

4.16.A

Determine the result of calling recursive methods.

ESSENTIAL KNOWLEDGE

4.16.A.1

A *recursive method* is a method that calls itself. Recursive methods contain at least one base case, which halts the recursion, and at least one recursive call. Recursion is another form of repetition.

4.16.A.2

Each recursive call has its own set of local variables, including the parameters. Parameter values capture the progress of a recursive process, much like loop control variable values capture the progress of a loop.

4.16.A.3

Any recursive solution can be replicated through the use of an iterative approach and vice versa.

X EXCLUSION STATEMENT—*Writing recursive code is outside the scope of the AP Computer Science A course and exam.*

© 2024 College Board

TOPIC 4.17
Recursive Searching and Sorting

LEARNING OBJECTIVE	ESSENTIAL KNOWLEDGE
4.17.A Determine the result of executing recursive algorithms that use strings or collections.	**4.17.A.1** Recursion can be used to traverse `String` objects, arrays, and `ArrayList` objects.
4.17.B Determine the result of each iteration of a binary search algorithm used to search for information in a collection.	**4.17.B.1** Data must be in sorted order to use the binary search algorithm. *Binary search* starts at the middle of a sorted array or `ArrayList` and eliminates half of the array or `ArrayList` in each recursive call until the desired value is found or all elements have been eliminated. **4.17.B.2** Binary search is typically more efficient than linear search. **EXCLUSION STATEMENT**—*Search algorithms other than linear and binary search are outside the scope of the AP Computer Science A course and exam.* **4.17.B.3** The binary search algorithm can be written either iteratively or recursively.
4.17.C Determine the result of each iteration of the merge sort algorithm when used to sort a collection.	**4.17.C.1** *Merge sort* is a recursive sorting algorithm that can be used to sort elements in an array or `ArrayList`. **EXCLUSION STATEMENT**—*Sorting algorithms other than selection, insertion, and merge sort are outside the scope of the AP Computer Science A course and exam.* **4.17.C.2** Merge sort repeatedly divides an array into smaller subarrays until each subarray is one element and then recursively merges the sorted subarrays back together in sorted order to form the final sorted array.

© 2024 College Board

© 2024 The College Board.