GACE® Computer Science Assessment
Test at a Glance

Updated April 2019

See the GACE® Computer Science Assessment Study Companion for practice questions and preparation resources.

Assessment Name	Computer Science
Grade Level	P–12
Test Code	555
Testing Time	3 hours
Test Duration	3.5 hours
Test Format	Computer delivered
Number of Selected-response Questions	100
Question Format	The test consists of a variety of short-answer questions such as selected-response questions, where you select one answer choice or multiple answer choices (depending on what the question asks for), questions where you enter your answer in a text box, and other types of questions. You can review the possible question types in the *Guide to Taking a GACE Computer-delivered Test*.
Number of Constructed-response Questions	0

About this Assessment

The GACE Computer Science assessment is designed to measure the professional knowledge of prospective secondary school computer science teachers in the state of Georgia.

The testing time is the amount of time you will have to answer the questions on the test. Test duration includes time for tutorials and directional screens that may be included in the test.

The total number of questions that are scored is typically smaller than the total number of questions on the test. Most tests that contain selected-response questions also include embedded pretest questions, which are not used in calculating your score. By including pretest questions in the assessment, ETS is able to analyze actual test-taker performance on proposed new questions and determine whether they should be included in future versions of the test.

Content Specifications

This assessment is organized into content **subareas**. Each subarea is further defined by a set of **objectives** and their **knowledge statements**.

- The objectives broadly define what an entry-level educator in this field in Georgia public schools should know and be able to do.
- The knowledge statements describe in greater detail the knowledge and skills eligible for testing.
- Some tests also include content material at the evidence level. This content serves as descriptors of what each knowledge statement encompasses.

See a breakdown of the subareas and objectives for this assessment on the following pages.

Test Subareas

Subarea	Approx. Percentage of Test
I. Impacts of Computing	15%
II. Algorithms and Computational Thinking	25%
III. Programming	30%
IV. Data	15%
V. Computing Systems and Networks	15%

Test Objectives

Subarea I: Impacts of Computing

Objective 1: Understands and applies knowledge of impact of, obstacles to, and effects of computing

The beginning Computer Science teacher:

A. Understands computing as a way of expressing creativity, solving problems, enabling communication, and fostering innovation in a variety of fields and careers
   - Recognizes that computers can be used to showcase creativity
   - Recognizes the benefits of using computers to solve problems
   - Provides examples of how computers enable communication and collaboration
   - Provides examples of how computers foster innovation

B. Knows the obstacles to equal access to computing among different groups and the impact of those obstacles
   - Identifies obstacles to equal access to computing among different groups (e.g., groups defined by gender, socioeconomic status, disability/accessibility needs) and the impact of those obstacles
   - Identifies factors that contribute to the digital divide
   - Matches obstacles to equal access with effective solutions

C. Understands beneficial and harmful effects of computing innovations and the trade-offs between them
   - Analyzes computing innovations in terms of their social, economic, and cultural impacts, both beneficial and harmful

- Identifies trade-offs between beneficial and harmful effects of computer innovations

Objective 2: Understands and applies knowledge of issues regarding intellectual property, ethics, privacy, and security in computing

The beginning Computer Science teacher:

A. Knows different methods of protecting intellectual property rights and the trade-offs between them in a variety of contexts (e.g., Creative Commons, open source, copyright)
   - Using correct vocabulary, describes how different methods of protecting intellectual property rights work
   - Given a context, identifies appropriate methods of protecting intellectual property rights
   - Identifies and compares trade-offs between different methods of protecting intellectual property rights

B. Understands ethical and unethical computing practices and their social, economic, and cultural implications
   - Identifies ethical and unethical computing practices in context
   - Describes the social, economic, and cultural implications of ethical and unethical computing practices
   - Identifies the conditions under which a given computing practice is ethical or legal

C. Knows privacy and security issues regarding the acquisition, use, and disclosure of information in a digital world
   - Using correct vocabulary, describes privacy and security issues
   - In context, identifies appropriate strategies to safeguard privacy and ensure security
   - Describes trade-offs between local and cloud-based data storage
   - Identifies methods that digital services use to collect information about users

Subarea II: Algorithms and Computational Thinking

Objective 1: Understands and applies knowledge of abstraction, pattern recognition, problem decomposition, number base conversion, and algorithm formats

The beginning Computer Science teacher:

A. Understands abstraction as a foundation of computer science
   - Identifies, creates, or completes the correct ordering, from low to high, of an abstraction hierarchy
   - Identifies abstractions in context

- Identifies details that can be removed from a solution in order to generalize it

B. Knows how to use pattern recognition, problem decomposition, and abstraction to develop an algorithm
   - Given a table of values or other data source, identifies the patterns in the data and identifies algorithms that could produce the patterns
   - Identifies components that could be part of an algorithm to solve a problem
   - Identifies actions and actors when decomposing a problem
   - Identifies appropriate decomposition strategies

C. Understands number base conversion and binary, decimal, and hexadecimal number systems
   - Converts between number bases
   - Analyzes and compares representations of numbers in different bases

D. Understands how to develop and analyze algorithms expressed in multiple formats (e.g., natural language, flowcharts, pseudocode)
   - Interprets diagrams that describe algorithms, given an explanation of the symbols used
   - Compares algorithms written in multiple formats
   - Traces and analyzes algorithms written in different formats
   - Identifies correct sequencing of steps in an algorithm and errors in sequencing

Objective 2: Understands and applies knowledge of algorithm analysis, searching and sorting algorithms, recursive algorithms, and randomization

The beginning Computer Science teacher:

A. Is familiar with the limitations of computing in terms of time, space, and solvability as well as with the use of heuristic solutions that can address these limitations
   - Identifies and compares algorithms that are linear, quadratic, exponential, or logarithmic
   - Recognizes the existence of problems that cannot be solved by a computer
   - In context, identifies factors that prevent a problem from being solvable
   - Identifies situations where heuristic solutions are useful
   - In context, identifies space and time limitations of computational solutions to problems

B. Understands searching and sorting algorithms; can analyze sorting algorithms for correctness and can analyze searching algorithms for correctness and efficiency
   - Traces algorithms and predicts output and intermediate results

- Calculates the number of comparisons required for linear and binary search algorithms

C. Understands simple recursive algorithms (e.g., *n* factorial, sum of first *n* integers)
   - Traces simple recursive algorithms
   - Provides missing steps in incomplete simple recursive algorithms
   - Identifies parts of a recursive algorithm (e.g., base or stopping condition, recursive call)
   - Identifies errors in simple recursive algorithms
   - Identifies an iterative algorithm that is equivalent to a recursive algorithm

D. Is familiar with the use of randomization in computing
   - Identifies appropriate uses of randomization in a variety of applications
   - Identifies the difference between random and pseudorandom numbers

Subarea III: Programming

Objective 1: Understands and applies knowledge of programming control structures, standard operators, variables, correctness, extensibility, modifiability, and reusability

The beginning Computer Science teacher:

A. Understands how to write and modify computer programs in a text-based programming language
   - Describes what a program does or is able to choose the code segment that correctly implements a given intended purpose
   - Identifies missing code in a code segment with a stated intended purpose
   - Places statements in appropriate order to create a correct program
   - Identifies how changing one part of a code segment will affect the output

B. Understands how to analyze computer programs in terms of correctness
   - Traces code and indicates the output printed or the value of variables after code segment execution
   - Indicates the inputs that produce given outputs for a code segment
   - Describes what a program does or chooses the code segment that correctly implements a given intended purpose
   - Identifies valid preconditions and postconditions
   - Compares two code segments or algorithms
   - Identifies the type of error produced by a code segment (i.e., syntax, runtime, compile-time, overflow, round-off, logic)
   - Identifies errors in incorrect code and changes that can be made to correct them

C. Knows the concepts of extensibility, modifiability, and reusability

- Identifies the meaning of the terms
- Identifies functionally equivalent statements or code segments that differ in one of these three ways
- Identifies situations where the use of constants or variables would be preferred over hard-coded values
- Identifies opportunities for parameterization
- Chooses code that improves on given code by making it more extensible, modifiable, or reusable
- Identifies changes that would improve a given code segment

D. Understands the three basic constructs used in programming: sequence, selection, and iteration

- Traces code and indicates the output printed or the value of variables after code segment execution
- Indicates inputs that produce given outputs for a code segment
- Describes what a program does or chooses the code segment that correctly implements a given intended purpose
- Identifies missing code in a code segment with a stated intended purpose
- Identifies equivalent statements or code segments
- Identifies the three constructs when used in code
- Identifies which of the constructs are needed to implement given functionality
- Converts code that does not use iteration to equivalent code that uses iteration

E. Understands how to use standard operators (i.e., assignment, arithmetic, relational, logical) and operator precedence to write programs

- Traces code and indicates the output displayed or the value of variables after code segment execution
- Indicates inputs that produce given outputs for a code segment
- Describes what a program does or chooses the code segment that correctly implements a stated intended purpose
- Identifies missing code in a code segment with a stated intended purpose
- Identifies equivalent statements or code segments
- Places statements in appropriate order to create a correct program
- Uses Boolean algebra to identify equivalent Boolean expressions
- Writes a Boolean expression equivalent to a given code, or identifies code equivalent to a given Boolean expression or English description

- Identifies the correct implementation of a given formula, including formulas with fractions
- Evaluates expressions that include arithmetic operations

F. Understands how to use variables and a variety of data types

- Identifies variables and data types (e.g., integers, floating point, string, Booleans, arrays/lists)
- Identifies the need for type conversion
- Traces code and indicates the output printed or the value of variables after code segment execution
- Indicates the inputs that produce given outputs for a code segment
- Describes what a program does or chooses the code segment that correctly implements a stated intended purpose
- Identifies missing code in a code segment with a stated intended purpose
- Identifies equivalent statements or code segments
- Places statements in appropriate order to creates a correct program
- Describes the difference between integer and floating point numeric data types
- Describes the difference between integer and floating point division
- Describes the benefits of the use of each data type
- Distinguishes between global and local scope
- Identifies the most appropriate data type in a given context
- Identifies the correct sequence of string operations to produce a given output

Objective 2: Understands and applies knowledge of procedures, event-driven programs, usability, data structures, debugging, documenting and reviewing code, libraries and APIs, IDEs, and programming language paradigms, including object-oriented concepts

The beginning Computer Science teacher:

A. Understands how to write and call procedures with parameters and return values

- Traces code and indicates the output printed or the value of variables after code segment execution
- Indicates inputs that produce given outputs for a code segment
- Describes what a program does or chooses the code segment that correctly implements a stated intended purpose
- Identifies missing code in a code segment with a stated intended purpose
- Identifies equivalent statements or code segments
- Places statements in appropriate order to create a correct program

- Traces code when references to objects and arrays are passed to procedures
- Traces code that includes nested procedure calls

B. Knows the concepts of event-driven programs that respond to external events (e.g., sensors, messages, clicks)
   - Traces code and indicates the output printed or the value of variables after code segment execution
   - Indicates inputs that produce given outputs for a code segment
   - Describes what a program does or chooses the code segment that correctly implements a stated intended purpose
   - Identifies missing code in a code segment with a stated intended purpose
   - Identifies possible errors due to asynchronous events
   - Identifies aspects of concurrency in event-driven programming

C. Is familiar with usability and user experience (e.g., ease of use and accessibility)
   - Identifies code that improves on given code in terms of usability or user experience
   - Identifies meaningful error messages
   - Identifies features that improve accessibility

D. Is familiar with dictionaries/maps, stacks, and queues
   - Identifies a data structure based on a description of behavior or appropriate use
   - Given goals, constraints, or context, identifies the most appropriate data structure
   - Traces code that uses a particular data structure

E. Understands how to use debugging techniques and appropriate test cases
   - Identifies which test cases are most useful for given code
   - Differentiates between different types of errors (e.g., overflow, round-off, syntax, runtime, compile-time, logic)
   - Describes useful debugging techniques (e.g., where to put print statements)
   - Differentiates between empirical testing and proof
   - Identifies errors in code and solutions to those errors

F. Is familiar with characteristics of well-documented computer programs that are usable, readable, and modular
   - Identifies characteristics of good documentation
   - Identifies good and poor documentation practices in context

G. Is familiar with techniques to obtain and use feedback to produce high-quality code (e.g., code reviews, peer feedback, end user feedback)

- Identifies situations in which each of the three listed techniques are useful

H. Knows how to use libraries and APIs
   - Identifies correct call(s) and use of return values given an API definition
   - Identifies reasons to use or not use libraries in place of writing original code
   - Identifies applications (e.g., math libraries, random number generation) that use APIs

I. Understands programming techniques to validate correct input and detect incorrect input
   - Identifies effective input data validation strategies
   - Compares data validation (proper range and format) and data verification (e.g., password verification)
   - Identifies improvements to code for which data validation is required

J. Is familiar with the features and capabilities of integrated development environments (IDEs)
   - Identifies components of IDEs
   - Identifies benefits and drawbacks of using IDEs
   - Identifies the costs and benefits of context editors

K. Is familiar with the differences between low- and high-level programming languages
   - Identifies characteristics of low- and high-level languages

L. Is familiar with different programming paradigms
   - Identifies the terminology of procedural programming
   - Identifies the terminology of object-oriented programming
   - Compares programming paradigms

M. Knows object-oriented programming concepts
   - Identifies classes, instance variables, and methods given a diagram
   - Identifies the benefits of inheritance and encapsulation
   - Identifies distinctions between overloading and overriding

N. Is familiar with program compilation and program interpretation
   - Identifies differences between compilation and interpretation
   - Identifies differences between source code and object code

Subarea IV: Data

Objective 1: Understands and applies knowledge of digitalization, data encryption and decryption, and computational tools

The beginning Computer Science teacher:

A. Understands bits as the universal medium for expressing digital information
   - Performs calculations, using bits and bytes
   - Determines the number of bits and bytes required to store a given amount of data
   - Given the description of an encoding scheme, encodes or decodes data
   - Describes lossy and lossless data compression
   - Explains why binary numbers are fundamental to the operation of computer systems

B. Is familiar with concepts of data encryption and decryption
   - Distinguishes between encoding and encryption
   - Identifies trade-offs in the use of data encryption

C. Knows how to use computational tools, including spreadsheets, to analyze data in order to discover, explain, and visualize patterns, connections, and trends
   - Transforms data to make it more useful
   - Identifies specific data or characteristics of specific data that need to be removed or modified before an entire data set can be used
   - Describes the use of spreadsheet operations (e.g., formulas, filters, sorts, charts, graphs) to analyze and visualize data

Objective 2: Understands and applies knowledge of simulation, modeling, and manipulation of data

The beginning Computer Science teacher:

A. Is familiar with the use of computing in simulation and modeling
   - Describes questions that can be answered with a given simulation, or explains what data and process are required in a simulation in order to answer a given question
   - Traces code in a simulation context
   - Identifies missing code in a simulation context
   - Identifies the impact of changes to simulations (e.g., more or fewer variables, more or less data)
   - Identifies applications of simulation and modeling

B. Is familiar with methods to store, manage, and manipulate data
   - Uses terminology and concepts of files and databases
   - Identifies measures of file size (e.g., byte, kilo, mega, giga, tera, peta)
   - Identifies issues connected with the storage requirements of computing applications, including scale, redundancy, and backup

C. Is familiar with a variety of computational methods for data collection, aggregation, and generation
   - Identifies the benefits of working with publicly available data sets
   - Identifies the types of data generated by surveys and sensors
   - Identifies examples of crowdsourcing and citizen science
   - Identifies appropriate data-collection methods for a given context and purpose

Subarea V: Computing Systems and Networks

Objective 1: Understands and applies knowledge of operating systems, computing systems, communication between devices, and cloud computing

The beginning Computer Science teacher:

A. Knows that operating systems are programs that control and coordinate interactions between hardware and software components
   - Identifies hardware components and their functions
   - Identifies software components and their functions
   - Identifies common operating systems tasks
   - Identifies resource issues that have an impact on functionality

B. Is familiar with computing systems embedded in everyday objects (e.g., Internet of Things [IoT], ATMs, medical devices)
   - Describes what an embedded system is
   - Defines what the IoT is and how it is used
   - Describes how sensors are used in embedded systems

C. Knows the capabilities, features, and uses of different types of computing systems (e.g., desktop, mobile, cluster)
   - Identifies capabilities, features, and uses for each type of computer system
   - Identifies criteria to evaluate and compare computing systems

D. Is familiar with computers as layers of abstraction from hardware (e.g., logic gates, chips) to software (e.g., system software, applications)
   - Identifies appropriate abstraction layers for hardware and software components

E. Is familiar with the steps required to execute a computer program (fetch-decode-execute cycles)
    - Describes what happens during fetch, decode, and execute, including the order of the steps in the cycle

F. Is familiar with trade-offs between local, network, and cloud computing and storage
    - Identifies advantages and disadvantages in terms of performance, cost, security, reliability, and collaboration
    - Identifies means of storing binary data

G. Is familiar with communication between devices
    - Identifies and compares wireless communication systems
    - Identifies and compares wired communication systems
    - Identifies and compares network types

Objective 2: Understands and applies knowledge of networks, including security issues and the Web

The beginning Computer Science teacher:

A. Knows components of networks
    - Identifies network hardware devices and their functions
    - Describes possible abstraction models of networks

B. Is familiar with factors that have an impact on network functionality
    - Defines basic terminology (e.g., bandwidth, load, latency)
    - Estimates necessary bandwidth and data size for a given situation
    - Identifies critical resources for a given situation

C. Is familiar with how Internet and Web protocols work
    - Describes the purpose of protocols and identifies common Internet and Web protocols
    - Compares IPv4 and IPv6
    - Identifies and describes the basic parts of a URL (e.g., protocol, subdomain, domain name, port, path)
    - Describes the hierarchical structure of names in the domain name system (DNS)
    - Describes the purpose and function of IP addressing
    - Identifies how Internet protocols address reliability, redundancy, and error handling

D. Is familiar with digital and physical strategies for maintaining security

- Identifies characteristics of strong passwords (e.g., length, bits per character)
- Identifies digital and physical security strategies
- Identifies trade-offs in the use of security measures (e.g., encryption, decryption, digital signatures and certificates)

E. Is familiar with concepts of cybersecurity

- Identifies and defines the five pillars of cybersecurity: confidentiality, integrity, availability, nonrepudiation, and authentication

F. Is familiar with the components that make up the Web (e.g., HTTP, HTML, browsers, servers, clients)

- Identifies the uses of markup languages
- Identifies the purposes of browsers, servers, and clients

Code Segments

Some stimulus material contains code segments written in pseudocode. The notation used in the pseudocode is described below

Pseudocode Notation

Explanation	Notation
Assignment operator	←
Arithmetic operators	`+ - / * ^ %` Note that `/` indicates floating point division unless stated otherwise.
Relational operators	`== < > ≤ ≥ ≠`
Logical operators	**`and`** **`or`** **`not`**
String concatenation operator	`+`
Boolean values	**`true`** **`false`**
Null	**`null`**
Comments	`// this is a single-line comment`

Placeholder for missing code	For example,
	`/* missing code */`
	`/* missing condition */`

Print A comment is used where necessary to indicate if a line feed or blank is appended to the argument.	**`print`** `arg`
Data types	**`boolean`** **`char`** **`double`** **`float`** **`int`** **`int[]`** **`int[][]`** **`short`** **`String`**
Array initialization and reference	**`int[]`** `a ← {1, 2, 3}` **`int`** `b[0..2] ← {1, 2, 3}` **`int[][]`** `c` `a[0]`

Explanation	Notation

Conditional statements: Indentation and **end if** statements are significant.
 	
Example:
`if ( x > 10 )`
`    print "big number"`
`else`
`    print "small number"`
`end if`
	if (condition)
	    block of statements
	end if
	
	if (condition)
	    block of statements
	else
	    another block of statements
	end if

Iterative statements: Indentation and **end** statements are significant.	**for** (initialization; condition; increment)
	    block of statements
	end for
	
	while (condition)
	    block of statements
	end while
	
	do
	    block of statements
	while (condition)
	
	repeat
	    block of statements
	until (condition)

Procedures: Indentation and **end** statements are significant.
 	
The return type is indicated in the procedure header and is based on the value returned by the procedure or is **void** if the procedure does not return a value.
	int procedureName (arg1,
	arg2,
	...)
	    block of statements
	    return value
	end procedureName
	
	void procedureName (arg1,
	arg2,
	...)
	    block of statements
	end procedureName

Classes	**class** className
	    variable declarations
	    procedures
	end class className

Object-oriented keywords	**extends** **new** **public** **private**