ETS PRAXIS

The *PRAXIS*® Study Companion

Computer Science (5652)

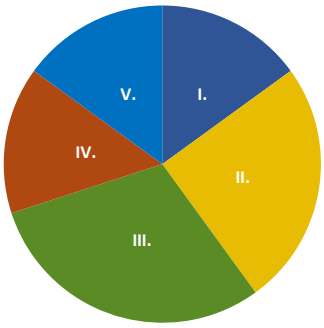
ETS Professional Educator PROGRAMS

www.ets.org/praxis

Table of Contents

Computer Sciences (5652)

Test at a Glance

<table>
<tr><td>Test Name</td><td colspan="3">Computer Science</td></tr>
<tr><td>Test Code</td><td colspan="3">5652</td></tr>
<tr><td>Time</td><td colspan="3">3 hours</td></tr>
<tr><td>Number of Questions</td><td colspan="3">100</td></tr>
<tr><td>Format</td><td colspan="3">The test consists of a variety of selected-response questions, where you select one or more answer choices. You can review the possible question types in Understanding Question Types.</td></tr>
<tr><td>Test Delivery</td><td colspan="3">Computer Delivered</td></tr>
<tr><td rowspan="8"> 
</td><td>Content Categories</td><td>Approximate Number of Questions*</td><td>Approximate Percentage of Examination</td></tr>
<tr><td>I. Impacts of Computing</td><td>15</td><td>15%</td></tr>
<tr><td>II. Algorithms and Computational Thinking</td><td>25</td><td>25%</td></tr>
<tr><td>III. Programming</td><td>30</td><td>30%</td></tr>
<tr><td>IV. Data</td><td>15</td><td>15%</td></tr>
<tr><td>V. Computing Systems and Networks</td><td>15</td><td>15%</td></tr>
<tr><td colspan="3">* includes both scored and unscored (pretest) questions. Depending on the number of pretest questions included in each scoring category, the total number of questions in that category may vary from one form of the test to another.</td></tr>
</table>

About The Test

The *Praxis* Computer Science test is designed to assess the computer science knowledge and competencies necessary for a beginning teacher of secondary school computer science. Examinees have typically completed a bachelor's program with an emphasis in computer science or computer science education.

The examinee will be required to understand and work with computer science concepts, use algorithms and computational thinking, work with code, manipulate data, and demonstrate knowledge of computing systems and networks.

The test is not designed to be aligned with any particular computer science curriculum, but it is intended to be consistent with the recommendations of national studies on computer science education, such as the *K -12 Computer Science Framework* (2016), the Computer Science Teachers Association (CSTA) *K -12 Computer Science Standards* (2017), and the International Society for Technology in Education (ISTE) *Computational Thinking Competencies*.

This test may contain some questions that do not count toward your score.

Content Topics

Content Topics in this chapter describe the knowledge and skills measured by the test.

Note: The use of "e.g." to start a list of examples implies that only a few examples are offered and the list is not exhaustive, whereas the use of "i.e." to start a list of examples implies that the given list of examples is complete.

Discussion Questions

Interspersed throughout the study topics are discussion areas, presented as open-ended questions or statements that are displayed in bulleted lists. These discussion areas are intended to help test your knowledge of fundamental concepts and your ability to apply those concepts to situations in the classroom or the real world. Most of the areas require you to combine several pieces of knowledge to formulate an integrated understanding and response. If you spend time on these areas, you will gain increased understanding and facility with the subject matter covered on the test. You might want to discuss these areas and your answers with a teacher or mentor.

Note that this study companion does **not** provide answers for the discussion area questions, but thinking about the answers to them will help improve your understanding of fundamental concepts and will probably help you answer a broad range of questions on the test.

I. Impacts of Computing

A. Understands and applies knowledge of impact of, obstacles to, and effects of computing

1. Understand computing as a way of expressing creativity, solving problems, enabling communication, and fostering innovation in a variety of fields and careers
   a. recognize that computers can be used to showcase creativity
   b. recognize the benefits of using computers to solve problems
   c. provide examples of how computers enable communication and collaboration
   d. provide examples of how computers foster innovation
2. Know the obstacles to equal access to computing among different groups and the impact of those obstacles
   a. identify 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
   b. identify factors that contribute to the digital divide
   c. match obstacles to equal access with effective solutions
3. Understand beneficial and harmful effects of computing innovations and the trade-offs between them
   a. analyze computing innovations in terms of their social, economic, and cultural impacts, both beneficial and harmful

b. identify trade-offs between beneficial and harmful effects of computer innovations

B. Understands and applies knowledge of issues regarding intellectual property, ethics, privacy, and security in computing

1. Know 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)
   a. using correct vocabulary, describe how different methods of protecting intellectual property rights work
   b. given a context, identify appropriate methods of protecting intellectual property rights
   c. identify and compare trade-offs between different methods of protecting intellectual property rights
2. Understand ethical and unethical computing practices and their social, economic, and cultural implications
   a. identify ethical and unethical computing practices in context
   b. describe the social, economic, and cultural implications of ethical and unethical computing practices
   c. identify the conditions under which a given computing practice is ethical or legal
3. Know privacy and security issues regarding the acquisition, use, and disclosure of information in a digital world
   a. using correct vocabulary, describe privacy and security issues
   b. in context, identify appropriate strategies to safeguard privacy and ensure security
   c. describe trade-offs between local and cloud-based data storage
   d. identify methods that digital services use to collect information about users

Discussion Questions: Impacts of Computing

- Can you give examples of computing innovations?
- Can you describe ways that computing enables creativity and innovation?
- Can you give examples of computing innovations that have impacted the quality of life?
- Can you give examples of how computers enable communication and collaboration in your life?
- Can you give examples of how computers foster innovation?
- Can you recognize obstacles to equal access to computing among different groups (e.g., groups defined by gender, socioeconomic status, disability/accessibility needs)?
- Can you give examples of effective solutions that address equal access for these groups?

- Can you describe what is meant by the digital divide? Can you give examples?
- Given a computing innovation, can you identify beneficial impacts as well as harmful impacts?
- Can you explain the value of copyrights? How does copyright affect the use of teaching and learning materials in your classroom?
- Can you describe what open source means? Can you give examples of commonly used open source applications?
- Can you give examples of materials that have a Creative Commons license?
- Can you describe how Creative Commons licenses are different from other types of copyright licenses?
- Can you give examples of ethical and unethical computing practices?
- Can you describe the social, economic, and cultural implications of ethical and unethical computing practices?
- Can you describe strategies that are used to safeguard privacy?
- Can you describe strategies that are used to ensure security?
- Can you give examples of data that is stored locally? Can you give examples of data that is stored in the cloud? Can you describe advantages and disadvantages of local and cloud-based data storage?
- Can you describe what types of information digital service providers collect about their users? How does that influence practices in your classroom?

II. Algorithms and Computational Thinking

A. Understands and applies knowledge of abstraction, pattern recognition, problem decomposition, number base conversion, and algorithm formats

1. Understand abstraction as a foundation of computer science
   a. identify, create, or complete the correct ordering, from low to high, of an abstraction hierarchy
   b. identify abstractions in context
   c. identify details that can be removed from a solution in order to generalize it
2. Know how to use pattern recognition, problem decomposition, and abstraction to develop an algorithm
   a. given a table of values or other data source, identify the patterns in the data and identify algorithms that could produce the patterns
   b. identify components that could be part of an algorithm to solve a problem
   c. identify actions and actors when decomposing a problem
   d. identify appropriate decomposition strategies

3. Understand number base conversion and binary, decimal, and hexadecimal number systems
   a. convert between number bases
   b. analyze and compare representations of numbers in different bases
4. Understand how to develop and analyze algorithms expressed in multiple formats (e.g., natural language, flowcharts, pseudocode)
   a. interpret diagrams that describe algorithms, given an explanation of the symbols used
   b. compare algorithms written in multiple formats
   c. trace and analyze algorithms written in different formats
   d. identify correct sequencing of steps in an algorithm and errors in sequencing

B. Understands and applies knowledge of algorithm analysis, searching and sorting algorithms, recursive algorithms, and randomization

1. Be 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
   a. identify and compare algorithms that are linear, quadratic, exponential, or logarithmic
   b. recognize the existence of problems that cannot be solved by a computer
   c. in context, identify factors that prevent a problem from being solvable
   d. identify situations where heuristic solutions are useful
   e. in context, identify space and time limitations of computational solutions to problems
2. Understand searching and sorting algorithms; can analyze sorting algorithms for correctness and can analyze searching algorithms for correctness and efficiency
   a. trace algorithms and predict output and intermediate results
   b. calculate the number of comparisons required for linear and binary search algorithms
3. Understand simple recursive algorithms (e.g., *n* factorial, sum of first *n* integers)
   a. trace simple recursive algorithms
   b. provide missing steps in incomplete simple recursive algorithms
   c. identify parts of a recursive algorithm (e.g., base or stopping condition, recursive call)
   d. identify errors in simple recursive algorithms
   e. identify an iterative algorithm that is equivalent to a recursive algorithm
4. Be familiar with the use of randomization in computing
   a. identify appropriate uses of randomization in a variety of applications
   b. identify the difference between random and pseudorandom numbers

Discussion Questions: Algorithms and Computational Thinking

- Can you give examples of various levels of abstractions?
- Can you describe how abstractions such as numbers, characters, and colors are represented as bits?
- Given a data source, such as a table of values, can you describe an algorithm that would produce those patterns of data?
- Can you apply decomposition strategies to decompose a problem?
- Can you convert between decimal numbers, binary numbers, octal numbers, and hexadecimal numbers?
- Can you trace an algorithm given in natural language, flowcharts, or pseudocode?
- Are you familiar with the pseudocode notation used in this test?
- Can you express an algorithm in natural language, in flowcharts, and in pseudocode, and can you translate from one format to another?
- Can you analyze an algorithm to identify steps that are missing or out of order?
- Are all problems solvable by an algorithm? Can you describe a problem that cannot be solved by a computer?
- Can you give examples of algorithms that are linear, quadratic, exponential, or logarithmic?
- Can you give examples of problems where heuristic solutions are useful?
- Can you describe common comparison-based sorting algorithms, such as insertion sort, selection sort, etc., and analyze the number of comparisons?
- Can you describe linear search and binary search, and analyze the number of comparisons?
- Can you describe a recursive algorithm?
- Can you trace a recursive algorithm, and analyze the number of recursive calls?
- Can you translate between a recursive algorithm and an equivalent iterative algorithm?
- Can you provide examples of problems where the use of randomization is useful?
- Can you describe the difference between random and pseudorandom numbers?
- Can you transform a random number generator to fit a specific range?

III. Programming

A. Understands and applies knowledge of programming control structures, standard operators, variables, correctness, extensibility, modifiability, and reusability

1. Understand how to write and modify computer programs in a text-based programming language
   a. describe what a program does or be able to choose the code segment that correctly implements a given intended purpose
   b. identify missing code in a code segment with a stated intended purpose
   c. place statements in appropriate order to create a correct program
   d. identify how changing one part of a code segment will affect the output
2. Understand how to analyze computer programs in terms of correctness
   a. trace code and indicate the output printed or the value of variables after code segment execution
   b. indicate the inputs that produce given outputs for a code segment
   c. describe what a program does or choose the code segment that correctly implements a given intended purpose
   d. identify valid preconditions and postconditions
   e. compare two code segments or algorithms
   f. identify the type of error produced by a code segment (i.e., syntax, runtime, compile-time, overflow, round-off, logic)
   g. identify errors in incorrect code and changes that can be made to correct them
3. Know the concepts of extensibility, modifiability, and reusability
   a. identify the meaning of the terms
   b. identify functionally equivalent statements or code segments that differ in one of these three ways
   c. identify situations where the use of constants or variables would be preferred over hard-coded values
   d. identify opportunities for parameterization
   e. choose code that improves on given code by making it more extensible, modifiable, or reusable
   f. identify changes that would improve a given code segment
4. Understand the three basic constructs used in programming: sequence, selection, and iteration
   a. trace code and indicate the output printed or the value of variables after code segment execution
   b. indicate inputs that produce given outputs for a code segment
   c. describe what a program does or choose the code segment that correctly implements a given intended purpose

   d. identify missing code in a code segment with a stated intended purpose
   e. identify equivalent statements or code segments
   f. identify the three constructs when used in code
   g. identify which of the constructs are needed to implement given functionality
   h. convert code that does not use iteration to equivalent code that uses iteration
5. Understand how to use standard operators (i.e., assignment, arithmetic, relational, logical) and operator precedence to write programs
   a. trace code and indicate the output displayed or the value of variables after code segment execution
   b. indicate inputs that produce given outputs for a code segment
   c. describe what a program does or choose the code segment that correctly implements a stated intended purpose
   d. identify missing code in a code segment with a stated intended purpose
   e. identify equivalent statements or code segments
   f. place statements in appropriate order to create a correct program
   g. use Boolean algebra to identify equivalent Boolean expressions
   h. write a Boolean expression equivalent to given code, or identify code equivalent to a given Boolean expression or English description
   i. identify the correct implementation of a given formula, including formulas with fractions
   j. evaluate expressions that include arithmetic operations
6. Understand how to use variables and a variety of data types
   a. identify variables and data types (e.g., integers, floating point, string, Booleans, arrays/lists)
   b. identify the need for type conversion
   c. trace code and indicate the output printed or the value of variables after code segment execution
   d. indicate the inputs that produce given outputs for a code segment
   e. describe what a program does or choose the code segment that correctly implements a stated intended purpose
   f. identify missing code in a code segment with a stated intended purpose
   g. identify equivalent statements or code segments
   h. place statements in appropriate order to create a correct program
   i. describe the difference between integer and floating point numeric data types
   j. describe the difference between integer and floating point division

k. describe the benefits of the use of each data type
l. distinguish between global and local scope
m. identify the most appropriate data type in a given context
n. identify the correct sequence of string operations to produce a given output

B. 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

1. Understand how to write and call procedures with parameters and return values
   a. trace code and indicate the output printed or the value of variables after code segment execution
   b. indicate inputs that produce given outputs for a code segment
   c. describe what a program does or choose the code segment that correctly implements a stated intended purpose
   d. identify missing code in a code segment with a stated intended purpose
   e. identify equivalent statements or code segments
   f. place statements in appropriate order to create a correct program
   g. trace code when references to objects and arrays are passed to procedures
   h. trace code that includes nested procedure calls
2. Know the concepts of event-driven programs that respond to external events (e.g., sensors, messages, clicks)
   a. trace code and indicate the output printed or the value of variables after code segment execution
   b. indicate inputs that produce given outputs for a code segment
   c. describe what a program does or choose the code segment that correctly implements a stated intended purpose
   d. identify missing code in a code segment with a stated intended purpose
   e. identify possible errors due to asynchronous events
   f. identify aspects of concurrency in event-driven programming
3. Be familiar with usability and user experience (e.g., ease of use and accessibility)
   a. identify code that improves on given code in terms of usability or user experience
   b. identify meaningful error messages
   c. identify features that improve accessibility
4. Be familiar with dictionaries/maps, stacks, and queues
   a. identify a data structure based on a description of behavior or appropriate use

   b. given goals, constraints, or context, identify the most appropriate data structure
   c. trace code that uses a particular data structure
5. Understand how to use debugging techniques and appropriate test cases
   a. identify which test cases are most useful for given code
   b. differentiate between different types of errors (e.g., overflow, round-off, syntax, runtime, compile-time, logic)
   c. describe useful debugging techniques (e.g., where to put print statements)
   d. differentiate between empirical testing and proof
   e. identify errors in code and solutions to those errors
6. Be familiar with characteristics of well-documented computer programs that are usable, readable, and modular
   a. identify characteristics of good documentation
   b. identify good and poor documentation practices in context
7. Be familiar with techniques to obtain and use feedback to produce high-quality code (e.g., code reviews, peer feedback, end user feedback)
   a. identify situations in which each of the three listed techniques are useful
8. Know how to use libraries and APIs
   a. identify correct call(s) and use of return values given an API definition
   b. identify reasons to use or not use libraries in place of writing original code
   c. identify applications (e.g., math libraries, random number generation) that use APIs
9. Understand programming techniques to validate correct input and detect incorrect input
   a. identify effective input data validation strategies
   b. compare data validation (proper range and format) and data verification (e.g., password verification)
   c. identify improvements to code for which data validation is required
10. Be familiar with the features and capabilities of integrated development environments (IDEs)
   a. identify components of IDEs
   b. identify benefits and drawbacks of using IDEs
   c. identify the costs and benefits of context editors
11. Be familiar with the differences between low- and high-level programming languages
   a. identify characteristics of low- and high-level languages

12. Be familiar with different programming paradigms
    a. identify the terminology of procedural programming
    b. identify the terminology of object-oriented programming
    c. compare programming paradigms
13. Know object-oriented programming concepts
    a. identify classes, instance variables, and methods given a diagram
    b. identify the benefits of inheritance and encapsulation
    c. identify distinctions between overloading and overriding
14. Be familiar with program compilation and program interpretation
    a. identify differences between compilation and interpretation
    b. identify differences between source code and object code

Discussion Questions: Programming

- Can you analyze a pseudocode program and trace its execution?
- Given a pseudocode program, can you describe what is does?
- Given an incomplete pseudocode program, can you identify the missing code?
- Can you analyze a pseudocode program in terms of correctness?
- Can you give examples of different types of errors (syntax, run-time, compile-time, overflow, round-off, logic) and describe how to fix the errors?
- Given a pseudocode program, can you identify valid preconditions and postconditions?
- Can you describe the concepts of extensibility, modifiability, and reusability?
- Given a pseudocode program, can you describe how to improve it by making it more extensible, modifiable, or reusable?
- Can you give examples where the use of constants or variables would be preferred over hard-coded values?
- Can you trace code that uses loops (**`for`**, **`while`**, **`do-while`**, etc.) and convert from one type of loop to another?
- Can you trace code that uses selection (**`if`**, **`if/else`**, **`switch`**, **`case`**, etc.) and convert from one type of selection to another?
- Can you evaluate expressions that use standard operators (assignment, arithmetic, relational, logical) and operator precedence?
- Can you use the rules of Boolean algebra to identify equivalent Boolean expressions?
- Can you describe the difference between decimal division and integer division?
- Can you describe the standard data types (integers, floating point, string, Booleans, arrays/lists)?
- Can you convert from one data type to another?
- Can you describe how the standard data types are stored in memory?

- Can you describe the difference between integer and floating point numeric data types?
- Can you evaluate an expression that uses string operations?
- Given a pseudocode program, can you determine the scope of a variable?
- Can you trace code that uses procedures with parameters and return values?
- Can you trace code that includes nested procedure calls?
- Can you analyze programs that respond to external events (sensors, messages, clicks)?
- Can you describe possible issues caused by concurrency?
- Can you describe how you have addressed usability and user experience in previous projects?
- Can you trace code that uses dictionaries/ maps, stacks, and queues?
- Can you give an example where the use of a dictionary/map data structure is appropriate? Can you give examples for stacks and queues?
- Can you describe debugging techniques you have used in previous projects?
- Can you describe how you selected test cases when testing code in previous projects?
- Can you describe good and poor documentation practices?
- Can you give examples of how you used code reviews, peer feedback, or end user feedback in your past projects?
- Can you describe how you worked with libraries and API in previous projects?
- Can you describe what input data validation techniques you used in previous projects?
- Can you describe features and capabilities of IDEs?
- Can you give examples of low-level and high-level programming languages?
- Can you describe differences between different programming paradigms (procedural, object-oriented, etc.)?
- Can you give an example of overloading and an example of overriding, and explain the difference between overloading and overriding?

IV. Data

A. Understands and applies knowledge of digitalization, data encryption and decryption, and computational tools

1. Understand bits as the universal medium for expressing digital information
   a. perform calculations, using bits and bytes
   b. determine the number of bits and bytes required to store a given amount of data

   c. given the description of an encoding scheme, encode or decode data
   d. describe lossy and lossless data compression
   e. explain why binary numbers are fundamental to the operation of computer systems
2. Be familiar with concepts of data encryption and decryption
   a. distinguish between encoding and encryption
   b. identify trade-offs in the use of data encryption
3. Know how to use computational tools, including spreadsheets, to analyze data in order to discover, explain, and visualize patterns, connections, and trends
   a. transform data to make it more useful
   b. identify specific data or characteristics of specific data that need to be removed or modified before an entire data set can be used
   c. describe the use of spreadsheet operations (e.g., formulas, filters, sorts, charts, graphs) to analyze and visualize data

B. Understands and applies knowledge of simulation, modeling, and manipulation of data

1. Be familiar with the use of computing in simulation and modeling
   a. describe questions that can be answered with a given simulation, or explain what data and process are required in a simulation in order to answer a given question
   b. trace code in a simulation context
   c. identify missing code in a simulation context
   d. identify the impact of changes to simulations (e.g., more or fewer variables, more or less data)
   e. identify applications of simulation and modeling
2. Be familiar with methods to store, manage, and manipulate data
   a. use terminology and concepts of files and databases
   b. identify measures of file size (e.g., byte, kilo, mega, giga, tera, peta)
   c. identify issues connected with the storage requirements of computing applications, including scale, redundancy, and backup
3. Be familiar with a variety of computational methods for data collection, aggregation, and generation
   a. identify the benefits of working with publicly available data sets
   b. identify the types of data generated by surveys and sensors

c. identify examples of crowdsourcing and citizen science
d. identify appropriate data-collection methods for a given context and purpose

Discussion areas: Data

- Can you describe how bits and binary numbers are fundamental elements of computer systems?
- Can you give examples of encoding and encryption, and explain the difference between them?
- Can you describe the difference between lossy and lossless data compression?
- Can you describe some steps that can be used to transform raw data to make it more useful or meaningful?
- Can you describe how spreadsheet operations can be used to transform and visualize data?
- Can you give examples of simulation and modeling?
- Can you analyze a simulation to identify missing code or suggest modifications?
- Can you describe the relationships between the measures of file size (byte, kilo, mega, giga, tera, peta)?
- Can you describe the impact of redundancy, scale, and backup on storage requirements?
- Can you describe methods used to collect, aggregate, and generate data?
- Can you give examples of crowdsourcing and citizen science?
- Can you identify situations where simulation and modeling will be useful?
- Can you determine the storage size needed for a given dataset?
- Can you transform binary data to other forms such as decimal and hexadecimal?
- Can you identify the missing or erroneous step(s) in a simulation?
- Can you identify potential issues during data processing/manipulation?
- Can you determine what data can be generated from a given simulation?

V. Computing Systems and Networks

A. Understands and applies knowledge of operating systems, computing systems, communication between devices, and cloud computing

1. Know that operating systems are programs that control and coordinate interactions between hardware and software components
   a. identify hardware components and their functions
   b. identify software components and their functions
   c. identify common operating systems tasks
   d. identify resource issues that have an impact on functionality

2. Be familiar with computing systems embedded in everyday objects (e.g., Internet of Things [IoT], ATMs, medical devices)
   a. describe what an embedded system is
   b. define what the IoT is and how it is used
   c. describe how sensors are used in embedded systems
3. Know the capabilities, features, and uses of different types of computing systems (e.g., desktop, mobile, cluster)
   a. identify capabilities, features, and uses for each type of computer system
   b. identify criteria to evaluate and compare computing systems
4. Be familiar with computers as layers of abstraction from hardware (e.g., logic gates, chips) to software (e.g., system software, applications)
   a. identify appropriate abstraction layers for hardware and software components
5. Be familiar with the steps required to execute a computer program (fetch-decode-execute cycles)
   a. describe what happens during fetch, decode, and execute, including the order of the steps in the cycle
6. Be familiar with trade-offs between local, network, and cloud computing and storage
   a. identify advantages and disadvantages in terms of performance, cost, security, reliability, and collaboration
   b. identify means of storing binary data
7. Be familiar with communication between devices
   a. identify and compare wireless communication systems
   b. identify and compare wired communication systems
   c. identify and compare network types

B. Understands and applies knowledge of networks, including security issues and the Web

1. Know components of networks
   a. identify network hardware devices and their functions
   b. describe possible abstraction models of networks
2. Be familiar with factors that have an impact on network functionality
   a. define basic terminology (e.g., bandwidth, load, latency)
   b. estimate necessary bandwidth and data size for a given situation
   c. identify critical resources for a given situation
3. Be familiar with how Internet and Web protocols work
   a. describe the purpose of protocols and identify common Internet and Web protocols
   b. compare IPv4 and IPv6
   c. identify and describe the basic parts of a URL (e.g., protocol, subdomain, domain name, port, path)
   d. describe the hierarchical structure of names in the domain

name system (DNS)

e. describe the purpose and function of IP addressing

f. identify how Internet protocols address reliability, redundancy, and error handling

4. Be familiar with digital and physical strategies for maintaining security
   a. identify characteristics of strong passwords (e.g., length, bits per character)
   b. identify digital and physical security strategies
   c. identify trade-offs in the use of security measures (e.g., encryption, decryption, digital signatures and certificates)
5. Be familiar with concepts of cybersecurity
   a. identify and define the five pillars of cybersecurity: confidentiality, integrity, availability, nonrepudiation, and authentication
6. Be familiar with the components that make up the Web (e.g., HTTP, HTML, browsers, servers, clients)
   a. identify the uses of markup languages
   b. identify the purposes of browsers, servers, and clients

Discussion Questions: Computing Systems and Networks

- Can you describe common operating system tasks?
- Can you describe hardware and software components and their functions?
- Can you describe what is meant by the Internet of Things?
- Can you describe what happens during each step of the fetch-decode-execute cycle and put the steps in the correct order?
- Can you identify advantages and disadvantages of local, network, and cloud computing and storage?
- Can you identify and compare wireless communication systems?
- Can you describe, identify, and compare wired communication systems?
- Can you identify and compare network types?
- Can you identify network hardware devices and their functions?
- Can you describe what bandwidth, load, and latency mean?
- Can you describe how the Internet works?
- Can you describe digital and physical security strategies that you have used?
- Can you describe confidentiality, integrity, availability, nonrepudiation, and authentication?

- Can you describe components that make up the Web (HTTP, HTML, browsers, servers, clients, etc.)?

Pseudocode Notation

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

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`**

Explanation	Notation
Array initialization and reference	**int[ ]** a ← {1, 2, 3} **int** b[0..2] ← {1, 2, 3} **int[ ][ ] c** a[0]

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)

Explanation	Notation
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**

Computer Sciences (5652) Sample Test Questions

Sample Questions

The sample questions that follow represent a number of the types of questions and topics that appear on the test. They are not, however, representative of the entire scope of the test in either content or difficulty. Answers with explanations follow the questions.

Directions: The sample consists of a variety of selected-response questions, where you select one or more answer choices.

For the following question, select all the answer choices that apply.

1. A technology company has developed an application that allows users to combine a text file and a recording of a person speaking to create an accurate audio representation of the person speaking the text. Which of the following are possible uses of the application?

   Select all that apply.

   (A) Audio versions of unrecorded speeches delivered by famous individuals could be produced from transcripts and recorded speeches delivered by the same individual.
   (B) An audio file could be fabricated in which an individual seems to be uttering slanderous statements about another individual.
   (C) Security systems based on voice activation could be circumvented by unauthorized users.

2. A programmer uses code published online under a Creative Commons Attribution (CC BY) license in a commercial product. Which of the following best describes an acceptable use of the code?

   (A) Copying code from the online source into the programmer's product without any other actions
   (B) Copying code from the online source into the programmer's product and limiting the copied code to ten code lines
   (C) Copying code from the online source into the programmer's product and changing all variable names
   (D) Copying code from the online source into the programmer's product and crediting the original author in the manner indicated by the license

3. Which of the following best describes the primary way in which a distributed denial-of-service (DDoS) attack differs from a denial-of-service (DoS) attack?

   (A) The goal of the attack
   (B) The number of computers being attacked
   (C) The number of computers launching the attack
   (D) The time period in which the attack occurs

4. Consider the following list.

   - Assembly language
   - Block-based programming language
   - Logic gate
   - Machine language

   Which of the following arranges the list in order from highest level of abstraction to lowest level of abstraction?

   (A) Block-based programming language, assembly language, machine language, logic gate
   (B) Block-based programming language, machine language, assembly language, logic gate
   (C) Block-based programming language, machine language, logic gate, assembly language
   (D) Machine language, block-based programming language, assembly language, logic gate

5. Chloe is playing a game in which she rolls a pair of dice 6 times. Her initial score is 0, and after each roll her score is recalculated. The table shows the outcomes of each roll and Chloe's recalculated score.

Roll	1st	2nd	3rd	4th	5th	6th
Die One	3	1	6	1	6	2
Die Two	4	5	2	6	6	2
Score	7	13	13	13	0	4

Consider the following pseudocode segment.

```
int score ← 0
int dieOne ← 0
int dieTwo ← 0
for ( int i ← 0; i < 6; i ← i + 1 )
    dieOne ← getDieOneValue ()     // returns the value
                                   // of the first die
    dieTwo ← getDieTwoValue ()     // returns the value
                                   // of the second die
    score ← newScore ( dieOne, dieTwo, score )
end for
```

Based on the data in the table, which of the following correctly implements the `newScore` procedure?

(A)

```
int newScore ( int diceOne, int diceTwo, int oldScore )
    return oldScore + diceOne + diceTwo
end newScore
```

(B)

```
int newScore ( int diceOne, int diceTwo, int oldScore )
    oldScore ← oldScore + diceOne + diceTwo
    if ( ( diceOne == 6 ) or ( diceTwo == 6 ) )
        oldScore ← oldScore - diceOne - diceTwo
    else
        if ( ( diceOne == 6 ) and ( diceTwo == 6 ) )
            oldScore ← 0
        end if
    end if
    return oldScore
end newScore
```

(C)

```
int newScore ( int diceOne, int diceTwo, int oldScore )
    if ( ( diceOne ≠ 6 ) and ( diceTwo ≠ 6 ) )
        return oldScore + diceOne + diceTwo
    else
        if ( ( diceOne == 6 ) and ( diceTwo == 6 ) )
            return 0
        else
            return oldScore
        end if
    end if
end newScore
```

(D)

```
int newScore ( int diceOne, int diceTwo, int oldScore )
    if ( diceOne == diceTwo )
        return 0
    else
        if ( ( diceOne == 6 ) or ( diceTwo == 6 ) )
            return oldScore
        else
            return oldScore + diceOne + diceTwo
        end if
    end if
end newScore
```

6. Which of the following is the hexadecimal representation of the decimal number 231_{10} ?

(A) 17_{16}
(B) $E4_{16}$
(C) $E7_{16}$
(D) $F4_{16}$

7. A sales representative has a list of clients to visit during an upcoming sales trip. No two clients live in the same city, and each client will be visited one time. The sales representative will return to the starting city at the end of the trip. To minimize travel expenses, a programmer at the sales representative's company has implemented an algorithm that generates all possible orderings of the clients' cities and then evaluates each ordering with respect to travel expenses. Which of the following best describes the running time of the algorithm in terms of the number of cities?

   (A) Factorial
   (B) Linear
   (C) Logarithmic
   (D) Quadratic

8. Consider the following pseudocode procedure, which sorts an integer array `arr` of length `len`. The first element of `arr` is at index `0`. A call `swap ( arr, i , j )` swaps the values of `arr[i]` and `arr[j]`.

```
void sort ( int[] arr, int len )
    int pos ← 0
    while ( pos < len )
        if ( pos == 0 )
            pos ← pos + 1
        else
            if ( arr[pos] > arr[pos - 1] )
                pos ← pos + 1
            else
                swap ( arr, pos, pos - 1 )
                pos ← pos - 1
            end if
        end if
    end while
end sort
```

   If `arr` originally contains the values `{2, 1, 5, 3, 4}`, what will the values in `arr` be after 6 iterations of the **`while`** loop?

   (A) `{1, 2, 3, 4, 5}`
   (B) `{1, 2, 3, 5, 4}`
   (C) `{1, 2, 5, 3, 4}`
   (D) `{2, 1, 5, 3, 4}`

9. Consider the recursive pseudocode procedure `f`, which is intended to return the product of consecutive integers from `5` to `n`, inclusive, for `n ≥ 5` and to return zero otherwise. For example, `f ( 7 )` returns `210`, which is `5*6*7`.

```
int f ( int n )
    if ( n < 5 )
        return 0
    else
        if ( n == 5 )
            return 5
        else
            /* missing statement */
        end if
    end if
end f
```

Which of the following could replace `/* missing statement */` so that the procedure `f` works as intended?

(A) `return n * f ( n - 1 )`
(B) `return n * f ( n - 5 )`
(C) `return ( n - 5 ) * f ( n )`
(D) `return ( n - 5 ) + f ( n - 1 )`

10. Consider the pseudocode procedure `findMax`, which is intended to return the largest value in an integer array `numList` of length `n`. The first element of `numList` is at index `0`.

```
int findMax ( int[] numList, int n )
    int max ← numList[0]
    int i ← 1
    while ( i < n )
        if ( /* missing condition */ )
            max ← numList[i]
        end if
        i ← i + 1
    end while
    return max
end findMax
```

Which of the following could replace `/* missing condition */` so that `findMax` works as intended?

(A) `numList[i] < n`
(B) `numList[i] < max`
(C) `numList[i] > max`
(D) `numList[i] > numList[n - 1]`

11. Consider the following pseudocode procedure, which is intended to print the odd integers from 1 up to `n`.

```
// precondition: n is a positive integer
void f ( int n )
    for ( int c ← 1; c ≤ n; c ← c + 1 )
        if ( ( c / 2 ) ≠ 0 )
            print c
        end if
    end for
end f
```

Which of the following is a true statement?

(A) The procedure works correctly and prints all the odd integers from 1 up to `n`.
(B) The procedure does not work correctly; a new variable needs to be declared inside the loop to test for odd numbers.
(C) The procedure does not work correctly; the variable `c` needs to be declared before the **`for`** loop.
(D) The procedure does not work correctly; there is a logic error in the **`if`** condition, which should say `c % 2` instead of `c / 2`.

12. Consider the following pseudocode procedure.

```
void mystery ( int n )
    while ( n ≠ 1 )
        if ( ( n % 2 ) == 1 )
            n ← 3 * n + 1
        else
            n ← n / 2
        end if
        print ( n )    // print a space after the number
    end while
end mystery
```

What is printed by the call `mystery ( 6 )` ?

(A) `3 10 5 16 8 4 2`
(B) `3 10 5 16 8 4 2 1`
(C) `10 5 16 8 4 2 1`
(D) Infinitely many numbers are printed because the **`while`** loop does not terminate.

13. Consider the following pseudocode Boolean expression, where `age` and `height` are two properly declared and initialized integer variables.

```
( age > 10 ) and ( height > 36 )
```

Which of the following is an equivalent Boolean expression?

(A) `not ( ( age < 10 ) and ( height < 36 ) )`
(B) `not ( ( age ≤ 10 ) and ( height ≤ 36 ) )`
(C) `not ( ( age < 10 ) or ( height < 36 ) )`
(D) `not ( ( age ≤ 10 ) or ( height ≤ 36 ) )`

14. Consider a class `String` with the methods shown.

Procedure	Explanation
String toLowerCase ()	Converts the characters in the string to lowercase and returns the string.
String toUpperCase ()	Converts the characters in the string to uppercase and returns the string.
String substring (**int** begin, **int** end)	Returns the substring of the string from position `begin` to position `end - 1` or returns `""` if `begin` is less than zero or `begin ≥ end` or `end` is greater than the length of the string. The first character in the string is at position `0`.
int length ()	Returns the length of the string.

Which of the following pseudocode segments would produce the output `"aPPLESAUCE"` ?

(A)

```
String value ← "Applesauce"
value ← value.substring ( 0, 1 ).toLowerCase () +
        value.substring ( 1, value.length () ).toUpperCase ()
print value
```

(B)

```
String value ← "Applesauce"
value ← value.substring ( 0, 2 ).toLowerCase () +
        value.substring ( 1, value.length () ).toUpperCase ()
print value
```

(C)

```
String value ← "Applesauce"
value ← value.substring ( 0, 1 ).toUpperCase () +
        value.substring ( 1, value.length () ).toLowerCase()
print value
```

(D)

```
String value ← "Applesauce"
value ← value.toUpperCase ()
value ← value.toLowerCase ()
print value
```

15. Consider the following pseudocode procedure.

```
int mystery ( int n )
    int temp ← 0
    for ( int c ← 1; c ≤ n; c ← c + 1 )
        if ( ( c % 3 ) == 0 )
            temp ← temp + c
        end if
    end for
    return temp
end mystery
```

Which of the following best describes procedure `mystery` ?

(A) It returns a list of numbers from 1 to n.
(B) It prints every third number from 1 to n.
(C) It returns the sum of the numbers from 1 to n.
(D) It returns the sum of the multiples of 3 from 1 to n.

16. Consider the following three scenarios, which could be modeled using three data structures—dictionary/map, queue, and stack.

Scenario 1: Cars line up in a single lane at a car wash. As each driver reaches the entrance of the car wash, the driver gets out of the car. An automatic pulley moves the car through the car wash as the driver walks to the exit of the car wash to pay and pick up the car.

Scenario 2: Contestants auditioning for a reality television talent show are assigned a unique numeric ID upon completing a registration form.

Scenario 3: Tennis balls are sold in a cylindrical can that can hold a maximum of 3 balls, where each ball except the bottom one rests on top of the ball below it. The 3 balls are placed in the can one at a time through one opening at the top of the can. The 3 balls are removed from the can one at a time through the same opening.

Which of the following shows the data structures that best model the scenarios?

	Scenario 1	Scenario 2	Scenario 3
(A)	Dictionary/map	Queue	Stack
(B)	Dictionary/map	Stack	Queue
(C)	Queue	Dictionary/map	Stack
(D)	Stack	Queue	Dictionary/map

17. Consider the following pseudocode segment with integer variables, which is intended to determine the largest value among variables `a`, `b`, and `c`, and to store that value in variable `max`. The code segment does not work as intended.

```
max ← -1
if ( a > b )
    if ( a > c )
        max ← a
    end if
else
    if ( b > c )
        max ← b
    else
        max ← c
    end if
end if
```

Which of the following test cases can be used to demonstrate that the code segment does not work as intended?

(A)

```
a ← 20
b ← 15
c ← 15
```

(B)

```
a ← 20
b ← 15
c ← 25
```

(C)

```
a ← 20
b ← 25
c ← 25
```

(D)

```
a ← 20
b ← 25
c ← 30
```

18. Huffman coding assigns unique variable-length codes to input values based on the frequency of occurrence of each value. Frequently occurring values are assigned codes that contain fewer bits than values that occur less frequently, which are assigned codes that contain more bits. Which of the following best describes an appropriate use of Huffman coding?

    (A) Decryption
    (B) Efficient sorting
    (C) Lossless compression
    (D) Lossy compression

19. Which of the following spreadsheet functions would be most useful for detecting improbably high or low values that have become part of a data set as a result of data entry errors?

    (A) A function that averages numeric values in a column or row
    (B) A function that counts the values in a column or row
    (C) A function that rounds a numeric value
    (D) A function that sorts values in a column or row

20. A computer simulation is created to simulate the growth of a certain plant species in different conditions. Which of the following actions could be used to validate the model used in the simulation?

    (A) Express the simulation software using both recursive and iterative algorithms. Compare the results of the recursive algorithm to those of the iterative algorithm.
    (B) Perform real-world experiments on the plant species' growth in different environments. Compare the experimental results to the results provided by the simulation.
    (C) Remove any unnecessary details from the model. Compare the running times of the original simulation and the simplified simulation.
    (D) Run the simulation software on multiple devices. Compare the results obtained from each of the devices.

21. Consider the following assumptions about electronic storage for the text in all the books of a university's libraries.

    - The libraries at the university collectively contain 3 million books.
    - A book contains an average of 400 pages.
    - A page contains an average of 50 lines.
    - A line on a page contains an average of 10 words.
    - A word contains an average of 5 letters/characters.
    - A letter/character is represented by 1 byte.

    Based on the given assumptions, which of the following is the unit in which the electronic storage required for the text in all the books of the university's libraries would best be measured?

    (A) Megabyte (2^{20} or approximately 10^{6} bytes)
    (B) Gigabyte (2^{30} or approximately 10^{9} bytes)
    (C) Terabyte (2^{40} or approximately 10^{12} bytes)
    (D) Petabyte (2^{50} or approximately 10^{15} bytes)

22. Which of the following is an example of the use of a device on the Internet of Things (IoT) ?

    (A) A car alerts a driver that it is about to hit an object.
    (B) A hiker uses a GPS watch to keep track of her position.
    (C) A refrigerator orders milk from an online delivery service when the milk in the refrigerator is almost gone.
    (D) A runner uses a watch with optical sensors to monitor his heart rate.

23. What is the role of the compiler in the process of developing executable software?

    (A) Managing specification files created as part of the development process
    (B) Running and testing the executable created by the programmer
    (C) Tracking older versions of the software in case an error is found and the software needs to be reverted to an earlier form
    (D) Translating a program written in an abstract, high-level language into a program with the same behavior expressed in machine code

24. Which of the following best describes a Web server?

    (A) A computer system that delivers Web pages to clients
    (B) A computer system that determines the shortest path between two computers over the Internet
    (C) A computer system running software that provides a user-friendly interface for creating Web pages
    (D) A computer system that translates domain names to IP addresses

25. Which pillar of cybersecurity is compromised when someone logs into a system using a stolen login and password?

    (A) Authentication
    (B) Confidentiality
    (C) Integrity
    (D) Nonrepudiation

Answers

1. Options (A), (B), and (C) are correct. All three choices are possible uses of the application.

   Choice (A): Using the application to combine a transcript of an unrecorded speech with existing recordings of an individual would create an audio reconstruction of the unrecorded speech.

   Choice (B): Using the application to combine false statements with existing recordings of an individual would create an audio file in which the individual seems to be uttering the false statements.

   Choice (C): The application could be used to create voice commands that could fool voice activation systems.

2. Option (D) is correct. The Creative Commons Attribution (CC BY) license allows anyone to use, revise, and distribute the code, even commercially, as long as the original author is credited properly.

3. Option (C) is correct. A DDoS attack is a DoS attack that originates from many different sources.

4. Option (A) is correct. Of the entries in the list, block-based programming language has the highest level of abstraction, logic gate has the lowest level of abstraction, and the other two entries are in between. Machine language is closer to hardware than is assembly language, so assembly language has a higher level of abstraction than does machine language.

5. Option (C) is correct. An analysis of the information in the table shows that: (i) when neither of the two dice is a 6, the score increases by the sum of the values of the two dice; (ii) when both of the dice are a 6, the score becomes 0; and (iii) when one but not both of the two dice is a 6, the score remains the same. Only the code segment in choice (C) is compatible with these observations—the first `return` statement in option (C) corresponds to case (i) above; the second `return` statement corresponds to case (ii); and the third `return` statement corresponds to case (iii).

6. Option (C) is correct. Since $231 = 14 \times 16 + 7$, the decimal number 231_{10} is equivalent to the hexadecimal number $E7_{16}$.

7. Option (A) is correct. The algorithm solves the traveling salesman problem using a brute-force approach, which enumerates all possible trips and identifies the least expensive one. The number of possible orderings of the clients' cities is factorial in the number of cities, so the running time of the algorithm is factorial.

8. Option (B) is correct. At the end of the first iteration, the value of `pos` is `1` and the array is unchanged. At the end of the second iteration, the value of `pos` is `0` and the array is `{1, 2, 5, 3, 4}`. At the end of the third iteration, the value of `pos` is `1` and the array is `{1, 2, 5, 3, 4}`. At the end of the fourth iteration, the value of `pos` is `2` and the array is `{1, 2, 5, 3, 4}`. At the end of the fifth iteration, the value of `pos` is `3` and the array is `{1, 2, 5, 3, 4}`. At the end of the sixth iteration, the value of `pos` is `2` and the array is `{1, 2, 3, 5, 4}`.

9. Option (A) is correct. When the execution of the code reaches the missing statement, the value of `n` is greater than or equal to `6`. When `n ≥ 6`, the value of `f ( n )` is the product of consecutive integers from `5` to `n`, which is the product of the consecutive integers from `n` down to `5`, which is `n` times the product of the consecutive integers from `n - 1` down to `5`, which is `n` times `f ( n - 1 )`. Therefore, when `n ≥ 6`, we have the recurrence relation `f ( n ) = n * f ( n - 1 )`.

10. Option (C) is correct. At the beginning of each iteration of the **`while`** loop, the value of `max` is the largest value in the subarray `numList[0..i-1]`. The missing statement is `numList[i] > max`, which is equivalent to saying that `numList[i]` is greater than all the elements in the subarray `numList[0..i-1]`. If the comparison is true, the value of `numList[i]` is assigned to `max`; If not, the value of `max` is unchanged.

11. Option (D) is correct. The procedure does not work correctly—as is, the procedure does not print anything when `n` is `1` and prints all consecutive integers from `2` to `n` when `n ≥ 2`. To fix the procedure, the integer division operator `/` in the **`if`** condition needs to be replaced with the modulus (remainder) operator `%`.

12. Option (B) is correct. The **`if-else`** statement checks whether the value of `n` is an odd integer; if `n` is odd, it replaces it with one more than its triple; if `n` is even, it halves it. Since the initial value of `n` is `6`, an even number, the procedure prints `3`. Since `3` is an odd number, the procedure prints `10`. Since `10` is an even number, the procedure prints `5`. Since `5` is an odd number, the procedure prints `16`. Since `16` is an even number, the procedure prints `8`. Since `8` is an even number, the procedure prints `4`. Since `4` is an even number, the procedure prints `2`. Since `2` is an even number, the procedure prints `1`. Since the value of `n` is now `1`, the execution exits the **`while`** loop and the procedure finishes executing.

13. Option (D) is correct. The Boolean expression `( age > 10 )` **`and`** `( height > 36 )` is true exactly when both `age` is greater than `10` and `height` is greater than `36`. The same Boolean expression is false exactly either when `age` is less than or equal to `10` or when `height` is less than or equal to `36`, which is the argument of **`not`** `()` in choice (D). Another approach to solve this question is to use Boolean logic rules. According to one of De Morgan's laws, if `p` and `q` represent Boolean variables, the Boolean expression `p` **`and`** `q` is equivalent to the Boolean expression **`not`** `( p )` **`or not`** `( q ) )`. If `p` represents the Boolean expression `age > 10` and if `q` represents the Boolean expression `height > 36`, then **`not`** `( p )` represents `age ≤ 10` and **`not`** `( q )` represents `height ≤ 36`.

14. Option (A) is correct. To transform `"Applesauce"` into `"aPPLESAUCE"`, the character at position `0` needs to change to lowercase and the characters from position `1` through the end of the string need to change to uppercase. The string that represents the character at position `0` is `value.substring ( 0, 1 )`. The string that represents the characters at positions `1` and higher is `value.substring ( 1, value.length() )`.

15. Option (D) is correct. The **`for`** loop iterates over the values of `c` from `1` to `n`, but the value of `temp` increases by `c` only when `c` is a multiple of `3`.

16. Option (C) is correct. In scenario 1, the cars are washed in first-in first-out order, which characterizes the functionality of a queue. In scenario 2, the unique numeric IDs represent a mapping function, which is best supported by a dictionary/map data structure. In scenario 3, the balls are inserted in and removed from the can in first-in last-out order, which characterizes the functionality of a stack.

17. Option (B) is correct. Since 20 is greater than 15, the outer if condition evaluates to true and then the condition a > c is evaluated. Since 20 is not greater than 25, the condition a > c evaluates to false and max is not updated. The result is that the value of max at the end of the code segment is -1 instead of 25.

18. Option (C) is correct. Huffman coding is appropriately used for lossless compression because the original values can be recovered from the uniquely assigned codes with no loss of data.

19. Option (D) is correct. A function that sorts values in a column or row will facilitate detecting extreme values in that column or row.

20. Option (B) is correct. The model is validated by comparing the simulation results with the real-world data.

21. Option (C) is correct. The size of the storage needed is approximately $3 \times 10^6 \times 4 \times 10^2 \times 5 \times 10^1 \times 10^1 \times 5 \times 1$ bytes, which is equivalent to 3×10^{12} bytes. This number is best expressed in terabytes.

22. Option (C) is correct. An Internet-connected smart refrigerator that can track its milk inventory would be able to communicate with online delivery services and order milk shipments whenever the milk inventory is low.

23. Option (D) is correct. A compiler translates a program written in a high-level programming language into low-level instructions that can be read and executed by the computer.

24. Option (A) is correct. A Web server is a server that provides content to clients by using Hypertext Transfer Protocol (HTTP).

25. Option (A) is correct. Authentication is the process of confirming a valid identification. Successful use of false credentials results in incorrect identification of the user.

Understanding Question Types

The *Praxis*® assessments include a variety of question types: constructed response (for which you write a response of your own); selected response, for which you select one or more answers from a list of choices or make another kind of selection (e.g., by selecting a sentence in a text or by selecting part of a graphic); and numeric entry, for which you enter a numeric value in an answer field. You may be familiar with these question formats from taking other standardized tests. If not, familiarize yourself with them so you don't spend time during the test figuring out how to answer them.

Understanding Selected-Response and Numeric-Entry Questions

For most questions, you respond by selecting an oval to select a single answer from a list of answer choices.

However, interactive question types may also ask you to respond by:

- Selecting more than one choice from a list of choices.
- Typing in a numeric-entry box. When the answer is a number, you may be asked to enter a numerical answer. Some questions may have more than one entry box to enter a response. Numeric-entry questions typically appear on mathematics-related tests.
- Selecting parts of a graphic. In some questions, you will select your answers by selecting a location (or locations) on a graphic such as a map or chart, as opposed to choosing your answer from a list.
- Selecting sentences. In questions with reading passages, you may be asked to choose your answers by selecting a sentence (or sentences) within the reading passage.
- Dragging and dropping answer choices into targets on the screen. You may be asked to select answers from a list of choices and to drag your answers to the appropriate location in a table, paragraph of text or graphic.
- Selecting answer choices from a drop-down menu. You may be asked to choose answers by selecting choices from a drop-down menu (e.g., to complete a sentence).

Remember that with every question you will get clear instructions.

Understanding Constructed-Response Questions

Some tests include constructed-response questions, which require you to demonstrate your knowledge in a subject area by writing your own response to topics. Essays and short-answer questions are types of constructed-response questions.

For example, an essay question might present you with a topic and ask you to discuss the extent to which you agree or disagree with the opinion stated. You must support your position with specific reasons and examples from your own experience, observations, or reading.

Review a few sample essay topics:

- *Brown* v. *Board of Education of Topeka*

  "We come then to the question presented: Does segregation of children in public schools solely on the basis of race, even though the physical facilities and other 'tangible' factors may be equal, deprive the children of the minority group of equal educational opportunities? We believe that it does."

  A. What legal doctrine or principle, established in *Plessy* v. *Ferguson* (1896), did the Supreme Court reverse when it issued the 1954 ruling quoted above?
  B. What was the rationale given by the justices for their 1954 ruling?

- *In his self-analysis, Mr. Payton says that the better-performing students say small-group work is boring and that they learn more working alone or only with students like themselves. Assume that Mr. Payton wants to continue using cooperative learning groups because he believes they have value for all students.*

  - Describe **TWO** strategies he could use to address the concerns of the students who have complained.
  - Explain how each strategy suggested could provide an opportunity to improve the functioning of cooperative learning groups. Base your response on principles of effective instructional strategies.

- *"Minimum-wage jobs are a ticket to nowhere. They are boring and repetitive and teach employees little or nothing of value. Minimum-wage employers take advantage of people because they need a job."*

  - Discuss the extent to which you agree or disagree with this opinion. Support your views with specific reasons and examples from your own experience, observations, or reading.

Keep these things in mind when you respond to a constructed-response question:

1. **Answer the question accurately.** Analyze what each part of the question is asking you to do. If the question asks you to describe or discuss, you should provide more than just a list.
2. **Answer the question completely.** If a question asks you to do three distinct things in your response, you should cover all three things for the best score. Otherwise, no matter how well you write, you will not be awarded full credit.
3. **Answer the question that is asked.** Do not change the question or challenge the basis of the question. You will receive no credit or a low score if you answer another question or if you state, for example, that there is no possible answer.
4. **Give a thorough and detailed response.** You must demonstrate that you have a thorough understanding of the subject matter. However, your response should be straightforward and not filled with unnecessary information.
5. **Take notes on scratch paper** so that you don't miss any details. Then you'll be sure to have all the information you need to answer the question.

Reread your response. Check that you have written what you thought you wrote. Be sure not to leave sentences unfinished or omit clarifying information.

General Assistance For The Test

Praxis® Interactive Practice Test

This full-length *Praxis*® practice test lets you practice answering one set of authentic test questions in an environment that simulates the computer-delivered test.

- Timed just like the real test
- Correct answers with detailed explanations
- Practice test results for each content category

ETS provides a free interactive practice test with each test registration. You can learn more here.

Doing Your Best

Strategy and Success Tips

Effective *Praxis* test preparation doesn't just happen. You'll want to set clear goals and deadlines for yourself along the way. Learn from the experts. Get practical tips to help you navigate your Praxis test and make the best use of your time. Learn more at Strategy and Tips for Taking a *Praxis* Test.

Develop Your Study Plan

Planning your study time is important to help ensure that you review all content areas covered on the test. View a sample plan and learn how to create your own. Learn more at Develop a Study Plan.

Helpful Links

Ready to Register – How to register and the information you need to know to do so.

Disability Accommodations – Testing accommodations are available for test takers who meet ETS requirements.

PLNE Accommodations (ESL) – If English is not your primary language, you may be eligible for extended testing time.

What To Expect on Test Day – Knowing what to expect on test day can make you feel more at ease.

Getting Your Scores – Find out where and when you will receive your test scores.

State Requirements – Learn which tests your state requires you to take.

Other Praxis Tests – Learn about other *Praxis* tests and how to prepare for them.

To search for the *Praxis* test prep resources that meet your specific needs, visit:

www.ets.org/praxis/testprep

To purchase official test prep made by the creators of the *Praxis* tests, visit the ETS Store:

www.ets.org/praxis/store

www.ets.org