TExES Computer Science 8–12 (241)

Practice Exam 1

Certification Prep

---

5 Hours · 100 Questions · Selected-response (multiple choice)

DIRECTIONS: Choose the best answer for each question. Leave no questions unanswered.

Time Allowed: 5 Hours · **Questions:** 100 Questions · **Format:** Selected-response (multiple choice)

DOMAIN I — TECHNOLOGY APPLICATIONS CORE

Questions 113 · ~13% of exam

Competency 001

The computer science teacher knows technology terminology and concepts; the appropriate use of hardware, software and digital files; and how to acquire, analyze and evaluate digital information.

1. Which of the following best describes the difference between RAM and ROM in a computer system?

(A) RAM is used for permanent storage of the operating system; ROM stores temporary data during processing.

(B) RAM is volatile memory used for temporary data during processing; ROM is non-volatile memory that retains data without power.

(C) RAM can only be read by the CPU; ROM can be both read and written to by the CPU.

(D) RAM is a type of secondary storage; ROM is a type of primary storage.

2. A student saves an image file as JPEG rather than PNG. What is the most significant trade-off of choosing JPEG?

(A) JPEG files cannot be opened on mobile devices.

(B) JPEG uses lossy compression, which may reduce image quality each time the file is saved.

(C) JPEG files are always larger than PNG files of the same image.

(D) JPEG files do not support color images.

3. Which of the following network types covers the largest geographic area?

(A) LAN (Local Area Network)

(B) MAN (Metropolitan Area Network)

(C) PAN (Personal Area Network)

(D) WAN (Wide Area Network)

4. A teacher wants to share a document with colleagues so that it can be opened and edited regardless of which word processing software they use. Which file format is most appropriate?

(A) .docx

(B) .rtf (Rich Text Format)

(C) .exe

(D) .png

5. Which of the following best describes an acceptable use policy (AUP) in a school setting?

(A) A legal document that gives students unlimited access to school computers
(B) A set of guidelines defining appropriate and inappropriate uses of school technology resources
(C) A contract between a software company and the school district regarding software licenses
(D) A technical document that describes the school's network hardware configuration

6. A student discovers a photograph online and uses it in a school project without checking its license. Which of the following licenses would have allowed this use without any restrictions?

(A) All Rights Reserved copyright
(B) Creative Commons Attribution-NonCommercial license
(C) Public Domain
(D) Creative Commons Attribution-NoDerivatives license

7. Which of the following is an example of unacceptable use of technology by a student?

(A) Using the school's internet to research a history project
(B) Accessing another student's online account without permission
(C) Submitting a digital portfolio of original artwork
(D) Using a school-provided IDE to complete a programming assignment

Competency 002

The computer science teacher knows how to use technology tools to solve problems, evaluate results and communicate information in a variety of formats for various audiences.

8. A computer science teacher wants students to investigate how changing the initial speed of a projectile affects its trajectory. Which type of tool is most appropriate for this activity?

(A) Word processing software
(B) A simulation or interactive virtual environment
(C) A desktop publishing application
(D) A database management system

9. A class is working on a long-term software development project. The teacher wants students to track tasks, set deadlines, and monitor who is responsible for each component. Which tool best supports this?

(A) An integrated development environment (IDE)
(B) A version control repository
(C) Project management software
(D) A code documentation generator

10. A teacher wants to evaluate student projects consistently and transparently. Which assessment tool best defines specific criteria and performance levels for grading?

(A) A standardized multiple-choice quiz
(B) A rubric
(C) A peer evaluation journal
(D) A timed coding challenge

11. A teacher is designing a website for student use. To ensure the site is accessible to students who use screen reader technology, which of the following is most important?

(A) Using high-resolution images on every page
(B) Setting background music to play automatically
(C) Providing descriptive alternative text for all images
(D) Requiring a specific web browser for access

Competency 003

The computer science teacher knows how to plan, organize, deliver and evaluate instruction that effectively utilizes current technology for teaching the Technology Applications TEKS to all students.

12. After administering a quiz on binary number conversions, a teacher finds that 70% of students answered the majority of those questions incorrectly. What is the most appropriate instructional response?

(A) Move on to the next topic, since binary conversions were already covered.
(B) Provide the class with the answer key and let them self-study.
(C) Re-teach binary conversions using a new strategy with hands-on practice before moving forward.
(D) Lower the difficulty of future assessments on number systems.

13. A computer science teacher wants to promote collaborative problem-solving during a lesson on algorithm design. Which activity best supports this goal?

(A) Assigning students individual textbook reading on algorithms
(B) Showing a lecture video and having students take notes independently
(C) Having small groups design and trace through a flowchart solution to a shared problem, then present it to the class
(D) Giving a vocabulary matching quiz on algorithm terminology

DOMAIN II — PROGRAM DESIGN AND DEVELOPMENT

Questions 1448 · ~35% of exam

Competency 004

The computer science teacher knows problem-solving strategies and different procedures for program design.

14. Which of the following correctly describes the order of phases in the Software Development Life Cycle (SDLC)?

(A) Design → Requirements → Implementation → Testing → Maintenance
(B) Requirements → Design → Implementation → Testing → Maintenance
(C) Implementation → Design → Requirements → Testing → Maintenance
(D) Testing → Requirements → Design → Implementation → Maintenance

15. A programmer approaches a large problem by first writing a high-level description, then breaking it into subproblems, then adding more detail to each subproblem until the code can be written. This approach is best described as:

(A) Black-box design
(B) Object-oriented design
(C) Top-down design with stepwise refinement
(D) Agile sprint development

16. In a flowchart, which shape is conventionally used to represent a decision point?

(A) Rectangle
(B) Oval/rounded rectangle
(C) Diamond
(D) Parallelogram

17. In software design, a "black box" component is one that:

(A) Contains intentional errors used for debugging practice
(B) Accepts defined inputs and produces defined outputs, with its internal implementation hidden from the user
(C) Cannot be modified after it is compiled
(D) Only works when connected to external hardware

18. A UML class diagram shows three classes: `Vehicle`, `Car`, and `Truck`. Arrows point from `Car` and `Truck` to `Vehicle`. What relationship does this diagram most likely represent?

(A) Composition: `Car` and `Truck` contain `Vehicle` as a component
(B) Dependency: `Car` and `Truck` depend on `Vehicle` for data
(C) Inheritance: `Car` and `Truck` are subclasses of `Vehicle`
(D) Association: `Car` and `Truck` are loosely connected to `Vehicle`

19. Which of the following design practices most directly contributes to a program being easy to maintain and expand?

(A) Writing all logic in a single large function
(B) Hardcoding specific values throughout the program
(C) Modular design with well-defined, reusable components and clear interfaces
(D) Removing all comments to reduce file size

20. A programmer creates a flowchart and reviews it with teammates before writing any code. What is the primary benefit of this step?

(A) It guarantees the final program will have no bugs.
(B) It allows structural and logical problems to be identified and corrected before time is spent coding.
(C) It is a required step before any compiler will accept the source code.
(D) It automatically generates the code from the diagram.

21. Which of the following is the best example of stepwise refinement?

(A) Writing the entire program in a single session without planning
(B) Beginning with a high-level description and progressively adding detail at each stage until code-ready specifications are complete
(C) Testing the program after every single line is written
(D) Copying reusable functions from a previous program

22. A UML sequence diagram is used during software design. What does this diagram primarily communicate?

(A) The attributes and methods of each class
(B) The file directory structure of the project
(C) The order of interactions between objects across time in a specific scenario
(D) The Big-O runtime of each method

23. Which of the following program characteristics most directly contributes to **reliability**?

(A) Using the fastest sorting algorithm available regardless of context
(B) Handling unexpected inputs gracefully rather than crashing
(C) Writing the program in a compiled language rather than an interpreted one
(D) Avoiding the use of user-defined functions

Competency 005

The computer science teacher knows procedures for software development and implementation.

24. Which of the following best describes the primary purpose of an Integrated Development Environment (IDE)?

(A) To compile only Java programs and no other languages
(B) To store source code remotely for team collaboration
(C) To provide a single, unified interface that includes a code editor, compiler or interpreter, debugger, and other developer tools
(D) To automatically generate test cases for a program

25. A student's program crashes whenever the user types a word into a field that expects a number. This type of error is best classified as a:

(A) Syntax error
(B) Logic error
(C) Runtime error
(D) Compilation error

26. Which of the following is an example of a **syntax error**?

(A) A loop that executes one too many iterations
(B) A program that produces incorrect output for certain inputs
(C) A missing closing parenthesis in a function call
(D) A program that crashes when the user divides by zero

27. A programmer is testing a function that calculates a percentage discount on a purchase. Which inputs best represent **boundary condition** testing?

(A) 25% and 50%
(B) 0% and 100%
(C) 10% and 75%
(D) 33% and 67%

28. Which of the following coding practices most improves the **readability** of source code for other developers?

(A) Using single-letter variable names to minimize typing
(B) Writing the entire program on as few lines as possible
(C) Removing all blank lines to reduce file size
(D) Using descriptive identifiers, consistent indentation, and clear inline comments

29. A development team uses version control software (such as Git) throughout a project. What is the most significant benefit of this practice?

(A) It automatically corrects logic errors in the code
(B) It allows multiple developers to track changes, work in parallel, and revert to any previous version if needed
(C) It compiles the code faster than working without it
(D) It eliminates the need for testing

30. A program is designed to calculate the area of a rectangle using user-provided length and width values. If the user enters a negative number, the program should display a helpful error message rather than crash. What technique does this represent?

(A) Stepwise refinement
(B) Input validation and error handling
(C) Encapsulation
(D) Polymorphism

31. What is the primary purpose of a **software library**?

(A) A file that stores the output data generated by a program
(B) A tool used to trace logic errors during debugging
(C) A repository where software licenses and legal documents are stored
(D) A collection of pre-written, reusable code modules that can be imported into other programs

32. A programmer wants to test a single function in isolation before integrating it into the larger program. This practice is called:

(A) Regression testing
(B) Integration testing
(C) System testing
(D) Unit testing

33. A student's program runs without crashing but consistently produces the wrong output. What type of error is most likely present?

(A) Syntax error
(B) Runtime error
(C) Logic error
(D) Compilation error

34. A program performs division using a value entered by the user. A code reviewer flags this as a potential problem. What is the most appropriate fix?

(A) Remove the division operation entirely
(B) Add a conditional check before the division to ensure the divisor is not zero
(C) Switch to a different programming language
(D) Round all values to the nearest integer before dividing

35. Which of the following best characterizes the **Agile** software development model?

(A) All requirements must be fully defined before any development begins
(B) The entire project is built in one long sequential phase before testing
(C) Development proceeds in short iterative cycles with frequent feedback, testing, and adaptation
(D) Testing only occurs after the complete product has been delivered

36. A programmer uses the comment `// This function returns the larger of two integers` above a function definition. What is the primary purpose of this comment?

(A) It tells the compiler how to optimize the function
(B) It documents the function's intent to improve readability and maintainability
(C) It causes the function to run faster
(D) It prevents other developers from modifying the function

Competency 006

The computer science teacher knows computer science terminology and concepts and the characteristics of different programming languages and paradigms.

37. Which of the following is an example of a **high-level** programming language?

(A) Binary machine code
(B) Assembly language
(C) Python
(D) Hexadecimal instruction sets

38. What is the primary difference between a **compiled** language and an **interpreted** language?

(A) Compiled languages are always easier to write than interpreted languages.
(B) In a compiled language, the entire source code is translated to machine code before execution; in an interpreted language, code is translated and executed line by line at runtime.
(C) Interpreted languages always produce programs that run faster.
(D) Compiled languages can only be used on Windows systems.

39. What does an **assembler** do?

(A) Translates source code written in a high-level language into machine code
(B) Translates assembly language mnemonics into binary machine code
(C) Executes a program line by line at runtime
(D) Combines separately compiled object files into a single executable

40. Which programming paradigm organizes a program around **objects** that encapsulate both data (attributes) and behavior (methods)?

(A) Procedural programming
(B) Functional programming
(C) Object-oriented programming
(D) Declarative programming

41. In **procedural programming**, the primary organizational unit of code is:

(A) A class
(B) An object
(C) An interface
(D) A function or procedure

42. Which of the following best describes **computational thinking**?

(A) Memorizing the syntax of multiple programming languages
(B) Running a program repeatedly until it produces the correct output
(C) Decomposing a problem into smaller parts, identifying patterns, developing abstractions, and designing step-by-step solutions
(D) Copying solutions from online sources and modifying variable names

43. A student writes a program that responds to mouse clicks and keyboard presses by triggering specific actions. This style of programming is best described as:

(A) Procedural programming
(B) Object-oriented programming
(C) Functional programming
(D) Event-driven programming

44. In software design, **abstraction** refers to:

(A) Writing code as quickly as possible without planning
(B) Translating code from one programming language to another
(C) Hiding complex implementation details and exposing only what is necessary for the user
(D) Running a program to test for runtime errors

45. Which of the following correctly distinguishes **syntax** from **semantics** in a programming language?

(A) Syntax refers to the meaning of code; semantics refers to its formatting rules.

(B) Syntax refers to the grammar and structural rules of the language; semantics refers to the meaning and intended behavior of the code.

(C) Both syntax and semantics describe only how code is indented and spaced.

(D) Semantics refers to execution speed; syntax refers to memory usage.

46. Consider the following recursive function.

```
int mystery(int n)
    if n ≤ 1
        return 1
    end if
    return n * mystery(n - 1)
```

Which of the following iterative segments produces the same result as `mystery(5)`?

(A) result ← 1; for i ← 5 to 1 step −1: result ← result + i

(B) result ← 1; for i ← 1 to 5: result ← result ∗ i

(C) result ← 0; for i ← 1 to 5: result ← result ∗ i

(D) result ← 5; for i ← 4 to 1 step −1: result ← result + i

47. Which of the following characteristics is unique to **object-oriented programming** compared to procedural programming?

(A) Support for loops and conditional statements

(B) Support for functions and procedures

(C) Support for encapsulation and inheritance

(D) Support for variables and data types

48. An interpreted language is generally easier to **debug interactively** than a compiled language because:

(A) Interpreted languages produce faster-running programs

(B) Errors are reported at the specific line where they occur during execution, allowing immediate feedback

(C) Interpreted code cannot produce logic errors

(D) The interpreter automatically fixes errors without user input

DOMAIN III — PROGRAMMING LANGUAGE TOPICS

Questions 4986 · ~38% of exam

Competency 007

The computer science teacher correctly and efficiently uses data types, data structures and functions in the development of code.

49. What is the decimal equivalent of the binary number **1011**?

(A) 9
(B) 10
(C) 11
(D) 13

50. What is the hexadecimal equivalent of the decimal number **255**?

(A) EE
(B) FE
(C) EF
(D) FF

51. Consider the following pseudocode. What are the two values printed?

```
int x ← 17
int y ← 5
print x / y
print x % y
```

(Note: On the TExES exam, `/` indicates integer division and `%` indicates modulus.)

(A) 3 and 1
(B) 3 and 2
(C) 3.4 and 2
(D) 2 and 2

52. Which of the following correctly describes the difference between a **local variable** and a **global variable**?

(A) A local variable is accessible throughout the entire program; a global variable is only accessible within one function.
(B) A local variable is accessible only within the block or function where it is declared; a global variable is accessible throughout the entire program.
(C) A local variable stores only integer values; a global variable stores only string values.
(D) There is no meaningful difference between local and global variables.

53. In most programming languages, casting the floating-point value `3.9` to an integer results in:

(A) 4 (rounded to the nearest integer)
(B) 3 (truncated toward zero)
(C) An error is generated
(D) The value is unchanged

54. Consider the following pseudocode. What values are printed?

```
procedure swap (pass-by-reference int a, pass-by-reference int b)
    int temp ← a
    a ← b
    b ← temp
end procedure

int x ← 10
int y ← 20
swap(x, y)
print x
print y
```

(A) 10 and 20
(B) 20 and 10
(C) 10 and 10
(D) 20 and 20

55. Which of the following data structures follows the **Last-In, First-Out (LIFO)** principle?

(A) Queue
(B) Linked list
(C) Stack
(D) Binary search tree

56. Which of the following data structures follows the **First-In, First-Out (FIFO)** principle?

(A) Stack
(B) Queue
(C) Binary search tree
(D) Graph

57. A programmer needs to store student names that must always be accessible in **alphabetical order**, and names will be **frequently inserted and deleted**. Which data structure best meets these requirements?

(A) A one-dimensional array
(B) A stack
(C) A queue
(D) A binary search tree

58. Consider the following pseudocode. What are the two printed values?

```
String s ← "Hello, World!"
print length(s)
print substring(s, 0, 5)
```

(Assume `length` returns the character count, and `substring(s, start, end)` returns characters from index `start` up to but not including index `end`. Indexing begins at 0.)

(A) 13 and "Hello"
(B) 12 and "Hello,"
(C) 13 and "Hello,"
(D) 12 and "Hello"

59. Which of the following is an advantage of a **linked list** compared to an array?

(A) Linked lists allow random access to elements by index in O(1) time.
(B) Linked lists store elements in contiguous memory locations.
(C) Linked lists allow efficient insertion and deletion of elements without shifting other elements.
(D) Linked lists always use less total memory than an array of the same size.

60. Consider the following pseudocode. What value is printed?

```
int[] A ← {5, 3, 8, 1, 9}
print A[2]
```

(Assume zero-based indexing.)

(A) 3
(B) 5
(C) 8
(D) 1

61. A `Stack S` supports `push(x)`, `pop()`, and `peek()`. Consider the following pseudocode. What two values are printed?

```
Stack S ← new Stack()
S.push(10)
S.push(20)
S.push(30)
print S.peek()
S.pop()
print S.peek()
```

(A) 10 and 20
(B) 30 and 10
(C) 30 and 20
(D) 20 and 10

62. Which of the following best describes the difference between a **primitive** data type and a **reference** data type?

(A) Primitive types store the actual value directly in memory; reference types store a memory address that points to where the value is stored.
(B) Primitive types can only hold integer values; reference types can hold any data.
(C) Reference types are always faster to access than primitive types.
(D) There is no meaningful difference in how they are stored in memory.

Competency 008

The computer science teacher correctly and efficiently uses statements and control structures in the development of code.

63. Consider the Boolean expression: `(x != 0) AND (10 / x > 2)`. If `x = 0`, what happens when the language uses **short-circuit evaluation**?

(A) A division-by-zero runtime error occurs because both operands are always evaluated
(B) The expression evaluates to `true`
(C) The expression evaluates to `false` without evaluating the second condition
(D) The expression cannot be compiled

64. Consider the following pseudocode. What is printed?

```
int x ← 5
if (x > 3 and x < 10)
    print "In range"
else
    print "Out of range"
end if
```

(A) "In range"
(B) "Out of range"
(C) Nothing is printed
(D) A runtime error is produced

65. What is the output of the following pseudocode?

```
int count ← 0
for (int i ← 1; i ≤ 5; i ← i + 1)
    count ← count + i
end for
print count
```

(A) 5
(B) 10
(C) 15
(D) 25

66. Consider the following pseudocode. How many times is "Hello" printed?

```
int i ← 1
while (i ≤ 4)
    print "Hello"
    i ← i + 1
end while
```

(A) 3
(B) 4
(C) 5
(D) The loop runs indefinitely

67. What is the key behavioral difference between a `while` loop and a `do-while` loop?

(A) A `while` loop always executes at least once; a `do-while` loop may not execute at all.
(B) A `do-while` loop always executes at least once; a `while` loop may not execute at all if the condition is initially false.
(C) A `while` loop can only iterate a fixed number of times; a `do-while` loop has no limit.
(D) There is no behavioral difference; they are always interchangeable.

68. In object-oriented programming, which principle describes bundling data and the methods that operate on that data within a single class, and restricting direct access to internal data from outside the class?

(A) Inheritance
(B) Polymorphism
(C) Abstraction
(D) Encapsulation

69. A class `Animal` defines a method `makeSound()`. Subclasses `Dog` and `Cat` each override `makeSound()` to produce different sounds. When `makeSound()` is called on a variable of type `Animal` that holds a `Dog` object, the dog's version runs. What OOP concept does this illustrate?

(A) Encapsulation
(B) Abstraction
(C) Composition
(D) Polymorphism

70. Which of the following correctly describes the relationship between a parent class and a child class in **inheritance**?

(A) The child class inherits no attributes or methods from the parent class.
(B) The child class inherits attributes and methods from the parent class, and may add new ones or override inherited ones.
(C) The parent class must always be abstract and cannot be instantiated.
(D) A child class can inherit from at most one parent class in all object-oriented languages.

71. Consider the following pseudocode. What is printed? Assume the language evaluates left to right and **short-circuits** `or`.

```
int x ← 2
int y ← 0
if (y == 0 or x / y > 1)
    x ← x + 1
end if
print x
```

(A) 2
(B) 3
(C) A runtime error occurs due to division by zero
(D) The result is undefined

72. What is the difference between method **overloading** and method **overriding**?

(A) Overloading redefines a parent class method in a subclass; overriding defines multiple methods with the same name but different parameters in the same class.
(B) Overloading defines multiple methods with the same name but different parameter lists within the same class; overriding redefines a parent class method in a subclass.
(C) Overloading and overriding are the same concept with different names.
(D) Overriding applies only to constructors; overloading applies to all other methods.

73. What does an **abstract class** provide that a regular (concrete) class cannot?

(A) The ability to create objects directly using the `new` keyword
(B) The ability to inherit from multiple parent classes simultaneously
(C) Faster execution speed due to optimized compilation
(D) A template that defines some methods and declares others without implementation, requiring subclasses to provide those implementations

74. Consider the following pseudocode. What is the output?

```
int x ← 10
repeat
    print x
    x ← x - 3
until (x ≤ 1)
```

(A) 10, 7, 4
(B) 10, 7, 4, 1
(C) 10, 7
(D) 7, 4, 1

75. What is the result of evaluating the expression `3 + 4 * 2 - 1`?

(A) 13
(B) 10
(C) 14
(D) 9

Competency 009

The computer science teacher knows how to construct, compare and analyze various algorithms.

76. A **linear search** is performed on an unsorted array of 1,000 elements to find a target value that is the last element. What is the number of comparisons made?

(A) 1
(B) 10
(C) 500
(D) 1,000

77. **Binary search** requires which precondition on the input data?

(A) The array must contain only integer values.
(B) The array must be sorted in ascending or descending order.
(C) The array must contain an even number of elements.
(D) The array must be stored as a linked list.

78. What is the Big-O time complexity of **binary search** in the worst case?

(A) O(1)
(B) O(log n)
(C) O(n)
(D) $O(n^2)$

79. What is the Big-O time complexity of **selection sort** in the worst case?

(A) O(n)
(B) O(n log n)
(C) $O(n^2)$
(D) O(log n)

80. Which sorting algorithm works by dividing the array into two halves, recursively sorting each half, and then **merging** the two sorted halves back together?

(A) Bubble sort
(B) Selection sort
(C) Insertion sort
(D) Merge sort

81. Consider the following pseudocode function:

```
int mystery(int n)
    if (n ≤ 1)
        return n
    else
        return mystery(n - 1) + mystery(n - 2)
    end if
end mystery
```

What value does `mystery(5)` return?

(A) 3
(B) 5
(C) 8
(D) 13

82. Which of the following best describes the difference between an **iterative** and a **recursive** solution?

(A) An iterative solution calls itself repeatedly; a recursive solution uses a loop.
(B) A recursive solution calls itself with a modified argument until a base case is reached; an iterative solution uses loops to repeat steps.
(C) Recursive solutions are always more efficient than iterative solutions.
(D) Iterative solutions can only be used with array data structures.

83. Which sorting algorithm works by repeatedly examining **adjacent pairs** of elements and swapping them if they are out of order, continuing until no swaps are needed?

(A) Selection sort
(B) Insertion sort
(C) Bubble sort
(D) Merge sort

84. Consider the following pseudocode, which implements a sorting algorithm:

```
for (int j ← 0; j < n - 1; j ← j + 1)
    int minIndex ← j
    for (int i ← j + 1; i < n; i ← i + 1)
        if (A[i] < A[minIndex])
            minIndex ← i
        end if
    end for
    if (minIndex ≠ j)
        swap(A[minIndex], A[j])
    end if
end for
```

This algorithm is best described as:

(A) Bubble sort
(B) Insertion sort
(C) Selection sort
(D) Merge sort

85. What is the **worst-case** time complexity of **quicksort**?

(A) O(n log n)
(B) O(n)
(C) O(n^2)
(D) O(log n)

86. A recursive function has been written but runs forever without stopping. Which component is most likely missing or incorrect?

(A) The recursive call
(B) The return type declaration
(C) The base case
(D) The function name

DOMAIN IV — SPECIALIZED TOPICS

Questions 87100 · ~14% of exam

Competency 010

The computer science teacher knows discrete mathematics topics relevant to computer science.

87. Which of the following truth tables correctly represents the result of **P AND Q** for all combinations of truth values?

P	Q	P AND Q
T	T	**?**
T	F	**?**
F	T	**?**
F	F	**?**

(A) T, T, T, T
(B) T, F, F, F
(C) F, T, T, F
(D) T, T, F, F

88. Using **De Morgan's Law**, which of the following is logically equivalent to `NOT (A OR B)`?

(A) `NOT A OR NOT B`
(B) `NOT A AND NOT B`
(C) `A AND B`
(D) `NOT A OR B`

89. A student states: *"If it is raining, then I will carry an umbrella."* Which of the following is the **contrapositive** of this statement?

(A) If I carry an umbrella, then it is raining.
(B) If it is not raining, then I will not carry an umbrella.
(C) If I do not carry an umbrella, then it is not raining.
(D) If I do not carry an umbrella, then it is raining.

90. How many different ways can a committee of **4 students** be selected from a group of **10** students, if the order of selection does not matter?

(A) 40
(B) 210
(C) 5,040
(D) 10,000

91. Which of the following correctly simplifies the Boolean expression `A AND (A OR B)` using Boolean algebra?

(A) `A OR B`
(B) `A AND B`
(C) `A`
(D) `B`

Competency 011

The computer science teacher knows digital forensics topics.

92. What is the primary goal of **digital forensics** in a cybersecurity context?

(A) To prevent viruses from entering a network in real time
(B) To recover and investigate digital evidence from devices, often in connection with a crime or security incident
(C) To design and implement secure software applications
(D) To encrypt all sensitive data before it is transmitted over a network

93. A company's computers are attacked by malicious software that **replicates itself and spreads to other computers automatically**, without requiring any user action. This is best described as a:

(A) Trojan horse
(B) Phishing attack
(C) Worm
(D) Rootkit

94. Which of the following actions is a critical first step in a proper digital forensics investigation?

(A) Reinstalling the operating system on the affected device to eliminate malware
(B) Permanently deleting potentially compromised files to protect user privacy
(C) Preserving and documenting the original state of digital evidence before any analysis begins
(D) Installing new security software to replace potentially compromised programs

Competency 012

The computer science teacher knows robotics topics.

95. A student wants to program a robot to navigate a course while **avoiding obstacles**. Which type of sensor is most directly useful for detecting objects in the robot's path?

(A) Light sensor
(B) Temperature sensor
(C) Sound sensor
(D) Distance or proximity sensor

96. A robotics team designs a robot to complete a specific task. After building an initial prototype, they test it, identify problems, and make improvements before building the next version. Which phase of the engineering design process does this best represent?

(A) Initial ideation
(B) Final production
(C) Evaluate and refine
(D) Documentation only

Competency 013

The computer science teacher knows game and mobile application development topics.

97. In game development, what term describes the detection and handling of when **two game objects occupy the same space** on the screen?

(A) Sprite rendering
(B) Frame rate control
(C) Collision detection
(D) Event handling

98. What is the result of the expression `(int) 7.9 + (int) 2.1` in a language where casting to `int` truncates the decimal portion?

(A) 10.0
(B) 10
(C) 9
(D) 11

99. In a 2D game coordinate system, how does the **y-axis** typically behave compared to standard mathematical Cartesian coordinates?

(A) The y-axis increases upward, exactly as in standard mathematics.
(B) The y-axis increases downward, with the origin located at the top-left corner of the screen.
(C) The y-axis is horizontal; the x-axis is vertical.
(D) The coordinate system is polar rather than Cartesian.

100. During which phase of game development are the game's **rules, objectives, win/lose conditions, and player interactions** primarily defined?

(A) Art and graphics design
(B) Game design
(C) Programming and coding
(D) Quality assurance testing

ANSWER KEY — Practice Exam 1

Use this key to score by domain and identify areas needing review.

#	Answer	Domain	Competency
1	B	I	001
2	B	I	001
3	D	I	001
4	B	I	001
5	B	I	001
6	C	I	001
7	B	I	001
8	B	I	002
9	C	I	002
10	B	I	002
11	C	I	002
12	C	I	003
13	C	I	003
14	B	II	004
15	C	II	004
16	C	II	004
17	B	II	004
18	C	II	004
19	C	II	004
20	B	II	004
21	B	II	004
22	C	II	004
23	B	II	004
24	C	II	005
25	C	II	005
26	C	II	005
27	B	II	005
28	D	II	005
29	B	II	005
30	B	II	005
31	D	II	005
32	D	II	005
33	C	II	005
34	B	II	005
35	C	II	005
36	B	II	005
37	C	II	006
38	B	II	006
39	B	II	006
40	C	II	006
41	D	II	006
42	C	II	006
43	D	II	006
44	C	II	006
45	B	II	006
46	B	III	009
47	C	II	006
48	B	II	006

#	Answer	Domain	Competency
49	C	III	007
50	D	III	007
51	B	III	007
52	B	III	007
53	B	III	007
54	B	III	007
55	C	III	007
56	B	III	007
57	D	III	007
58	A	III	007
59	C	III	007
60	C	III	007
61	C	III	007
62	A	III	007
63	C	III	008
64	A	III	008
65	C	III	008
66	B	III	008
67	B	III	008
68	D	III	008
69	D	III	008
70	B	III	008
71	B	III	008
72	B	III	008
73	D	III	008
74	A	III	008
75	B	III	008
76	D	III	009
77	B	III	009
78	B	III	009
79	C	III	009
80	D	III	009
81	B	III	009
82	B	III	009
83	C	III	009
84	C	III	009
85	C	III	009
86	C	III	009
87	B	IV	010
88	B	IV	010
89	C	IV	010
90	B	IV	010
91	C	IV	010
92	B	IV	011
93	C	IV	011
94	C	IV	011
95	D	IV	012
96	C	IV	012
97	C	IV	013
98	C	III	007
99	B	IV	013
100	B	IV	013

ANSWER EXPLANATIONS — Practice Exam 1

Brief rationale for each correct answer.

1. An AUP is a school policy document that communicates expectations for responsible use of technology resources, not a legal contract with a vendor or a technical network document. Teachers must understand AUPs to model and enforce appropriate digital citizenship.

2. RAM (Random Access Memory) is volatile—it loses its contents when power is removed—and is used for active program data; ROM (Read-Only Memory) retains its data without power and stores firmware.

3. A WAN (Wide Area Network) spans the largest geographic area—potentially global—connecting multiple LANs and MANs across cities, countries, or continents. The Internet is the most prominent example of a WAN.

4. RTF (Rich Text Format) is an open, widely supported format that preserves basic formatting (bold, fonts, etc.) and can be opened by virtually all word processors across operating systems. Proprietary formats like .docx may not be openable without specific software.

5. Accessing another student's online account without permission is a clear violation of privacy and an example of unauthorized access, which is prohibited by school AUPs and potentially by law. Downloading music from an approved site or looking up information are generally acceptable uses.

6. Public Domain works are not protected by copyright (either because protection expired or was never held), so they may be used freely without attribution or permission. Creative Commons, Fair Use, and similar frameworks have different conditions and restrictions.

7. JPEG uses lossy compression, which permanently discards some image data to achieve smaller file sizes; repeated saves can degrade quality further. PNG uses lossless compression, preserving all original pixel data at the cost of larger file sizes.

8. A rubric explicitly lists the criteria being assessed and defines what performance looks like at each level, giving both teacher and student a shared, transparent standard. Other tools like quizzes or peer journals lack this structured, criteria-based scoring dimension.

9. Providing descriptive alt text for images ensures screen readers can convey the image's meaning to users with visual impairments, making the site compliant with web accessibility standards (WCAG). Good alt text describes the image's purpose, not just its appearance.

10. A simulation or interactive virtual environment lets students systematically vary initial conditions and observe outcomes, supporting inquiry-based learning about dynamic systems. Static diagrams or quizzes cannot replicate this cause-and-effect exploration.

11. Project management software (e.g., Trello, Jira, Asana) provides tools for assigning tasks, tracking progress, managing deadlines, and visualizing workflow—exactly what a long-term collaborative coding project requires. IDEs, forums, and grade books do not provide these project coordination features.

12. When 70% of students miss the same concept, the appropriate response is to re-teach using a different instructional approach before advancing, not to move forward or simply provide answers. Formative assessment data should drive instructional decisions.

13. Having students collaboratively design and trace through a flowchart engages higher-order thinking (analysis and creation) and promotes discussion of problem-solving strategies, then reinforces learning by requiring students to explain their solution to peers. This reflects active, inquiry-based pedagogy.

14. A black-box component exposes only its interface (inputs and outputs) and hides its internal implementation, supporting abstraction and modularity. This concept is central to software engineering and allows components to be replaced or updated without affecting callers.

15. Modular design with well-defined interfaces isolates changes to individual components, making the program easier to test, debug, and extend without breaking other parts. Monolithic or tightly-coupled designs

are far harder to modify reliably.

16. Top-down design begins with a broad description of the problem and repeatedly breaks it into smaller, more detailed sub-tasks until each piece is small enough to code directly (stepwise refinement). This contrasts with bottom-up design, which builds small pieces first and assembles them.

17. A reliable program handles unexpected or invalid inputs gracefully—with informative error messages or safe fallback behavior—rather than crashing. Input validation and robust error handling are the primary techniques that enable this quality.

18. The diamond shape in a standard flowchart represents a decision point (a conditional branch with two or more outgoing paths), while rectangles represent process steps and ovals represent start/end points. This convention is widely used in algorithm design documentation.

19. In UML, an arrow from `Car` to `Vehicle` and from `Truck` to `Vehicle` with open arrowheads represents inheritance (generalization), indicating that `Car` and `Truck` are specialized subclasses of the more general `Vehicle` class. This is the standard UML notation for "is-a" relationships.

20. Reviewing a flowchart or design document before coding allows the team to catch structural flaws, missing cases, and logical errors cheaply—when they can be fixed with an eraser rather than a code rewrite. This is the value of design reviews in professional software development.

21. Stepwise refinement starts with an abstract, high-level description of a solution and systematically adds detail at each step until the description is specific enough to translate directly into code. This top-down process helps manage complexity in large programs.

22. A UML sequence diagram shows the chronological order of messages exchanged between objects, revealing how a particular scenario unfolds over time. Class diagrams show static structure; use-case diagrams show system-actor interactions.

23. The classic Software Development Life Cycle (SDLC) phases proceed in order: Requirements $\rightarrow$ Design $\rightarrow$ Implementation $\rightarrow$ Testing $\rightarrow$ Maintenance. Skipping or reordering phases—especially doing implementation before design—leads to costly rework.

24. A syntax error is a violation of the language's grammatical rules—such as a missing closing parenthesis—that prevents the code from being parsed or compiled. Logic and runtime errors occur after the code is syntactically valid.

25. An IDE integrates editor, compiler/interpreter, debugger, and related tools into one environment, increasing developer productivity by reducing context switching. Individual tools like text editors or standalone compilers provide only part of this functionality.

26. A logic error occurs when the code is syntactically correct and runs without crashing but produces incorrect results due to a flaw in the algorithm or conditional logic. Syntax errors prevent compilation; runtime errors cause crashes during execution.

27. Boundary values—the edges of the valid input range—are where off-by-one errors and edge-case bugs most commonly lurk, making 0% and 100% the most critical test cases for a discount function. This is the principle behind boundary-value analysis in software testing.

28. Descriptive identifiers make the purpose of variables and functions self-evident; consistent indentation reveals program structure visually; and inline comments explain non-obvious logic. Together these practices reduce the cognitive load for anyone reading or maintaining the code.

29. Comments that explain the purpose and behavior of a function serve as inline documentation, helping future readers—including the original developer—understand the code without having to trace through the implementation. This directly improves long-term maintainability.

30. Version control systems like Git maintain a complete history of every change, support branching for parallel development, and allow the team to roll back to any prior state if a bug is introduced. Without version control, coordinating changes across multiple developers becomes error-prone.

31. Unit testing isolates and tests a single function or module independently from the rest of the program,

making it easier to pinpoint the source of a failure. Integration testing checks how components interact; system testing evaluates the whole program.

32. A software library is a collection of pre-written, tested, reusable code (functions, classes, modules) that developers can import and call rather than writing from scratch. This promotes the DRY (Don't Repeat Yourself) principle and accelerates development.

33. A runtime error occurs when syntactically valid code encounters an unexpected condition during execution—such as receiving a string where a number is required—causing the program to crash. This differs from a compile-time syntax error or a logic error that produces wrong output silently.

34. Input validation checks that user-supplied values meet required constraints (e.g., positive numbers only) before the program uses them; error handling catches and responds to unexpected situations gracefully. Together they prevent crashes and incorrect calculations when users provide bad input.

35. Agile development proceeds in short iterations (sprints), with working software delivered frequently and requirements refined based on ongoing feedback from stakeholders. This contrasts with the Waterfall model, which completes all requirements before moving to design, then design before coding, etc.

36. The safest fix for a potential division-by-zero error is to add a guard condition that checks whether the divisor is zero before performing the division. This prevents the runtime error without changing the program's behavior for valid inputs.

37. Object-oriented programming introduces encapsulation (bundling data with methods) and inheritance (deriving new classes from existing ones), which are not features of procedural programming. These mechanisms support code reuse and modeling of real-world relationships.

38. Syntax defines the grammatical rules of how code must be written (e.g., punctuation, structure), while semantics defines what that code means and how it behaves when executed.

39. A compiler translates the entire source program into machine code before any execution begins, producing a standalone executable; an interpreter processes and executes code one statement at a time at runtime. This difference affects performance, portability, and the debugging experience.

40. Computational thinking encompasses decomposition, pattern recognition, abstraction, and algorithm design—a problem-solving framework applicable well beyond programming.

41. Event-driven programming structures program flow around events such as user actions (mouse clicks, key presses) rather than a sequential top-down execution. GUIs and interactive applications are the primary use case for this paradigm.

42. Python is a high-level language with syntax that abstracts away hardware details, making it readable and portable. Assembly and machine code are low-level; SQL is a domain-specific query language, not a general-purpose high-level language.

43. In procedural programming, code is organized into named functions (or procedures) that perform specific tasks and can be called by name. Classes and objects belong to OOP; events belong to event-driven programming.

44. Object-oriented programming (OOP) organizes code around objects that combine state (data) and behavior (methods), supporting encapsulation, inheritance, and polymorphism. Procedural programming organizes code around functions; functional programming treats computation as the evaluation of mathematical functions.

45. Because an interpreter executes code line by line, it can immediately report an error at the exact line that caused it, providing instant feedback during interactive development. Compiled languages report errors before execution, but runtime errors in compiled code may be harder to trace to source lines.

46. The recursive function computes n! (factorial). mystery(5) = 120. The equivalent iterative loop initializes result to 1 and multiplies each integer from 1 to 5.

47. Abstraction in software design means hiding complex implementation details behind a simple interface, allowing the user or programmer to work at a higher conceptual level without needing to understand the

internal mechanics. It is one of the four pillars of computational thinking.

48. An assembler is a program that translates human-readable assembly language mnemonics (like `MOV`, `ADD`) into the corresponding binary machine code instructions for a specific CPU architecture. It differs from a compiler, which translates a high-level language.

49. `peek()` returns the top element without removing it, so after pushing 10, 20, and 30, `peek()` returns 30; then `pop()` removes 30, making 20 the new top, so the second `peek()` returns 20. Stacks follow Last-In, First-Out (LIFO) ordering.

50. A binary search tree (BST) maintains the invariant that left-subtree values are smaller and right-subtree values are larger, enabling $O(\log n)$ average-case search and keeping elements in sorted order during traversal. Arrays, hash maps, and stacks do not inherently maintain sorted order.

51. With integer division, $17/5 = 3$ (the fractional part is discarded), and the modulo operator gives the remainder: $17\%5 = 2$.

52. Casting a float to an integer in most languages truncates toward zero, dropping the decimal portion without rounding, so 3.9 becomes 3. This behavior is important to distinguish from the `round()` or `floor()` functions.

53. Passing by reference means the function operates on the actual variables, not copies, so the swap inside the procedure changes the original `x` and `y`. After the call, `x` holds 20 and `y` holds 10, which are printed in that order.

54. A local variable's scope is limited to the function or block where it is declared, preventing unintended side effects elsewhere; a global variable is accessible from anywhere in the program, which can make code harder to debug. Scope management is essential for writing maintainable programs.

55. Linked lists store elements as nodes with pointers, so insertion and deletion only require updating a few pointers rather than shifting all subsequent elements as arrays require. However, linked lists sacrifice constant-time random access.

56. A queue follows First-In, First-Out (FIFO) ordering—elements are enqueued at the back and dequeued from the front, just like a waiting line. Stacks use LIFO; arrays and trees are general-purpose structures without inherent ordering policies.

57. Each hex digit represents 4 bits; $255 = 16 \times 15 + 15 = \texttt{FF}_{16}$. Recognizing that $255 = 2^8 - 1$ (all eight bits set) is a useful shortcut for binary/hex conversion.

58. Primitive types (e.g., `int`, `bool`) store their actual value directly in a variable's memory location, while reference types store a memory address pointing to an object elsewhere in the heap. This distinction affects assignment behavior (copy vs. shared reference) and is fundamental to understanding variable semantics.

59. Converting binary 1011 to decimal: $1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0 = 8 + 0 + 2 + 1 = 11$.

60. A stack follows LIFO—the last item pushed is the first to be popped—making it the correct structure for scenarios like undo operations, expression evaluation, and call stacks. Queues follow FIFO, not LIFO.

61. Zero-based indexing means the first element is at index 0; index 2 therefore refers to the third element of the array {5, 3, 8, 1, 9}, which is 8. Confusing zero-based with one-based indexing is a common source of off-by-one errors.

62. The string `Ёello, World!¨` has 13 characters (including the comma, space, and exclamation mark), so `length(s)` returns 13. `substring(s, 0, 5)` with zero-based indexing and exclusive end returns characters at indices 0–4, which is `Ёello¨`.

63. With short-circuit evaluation, when the first operand of AND is false (x != 0 is false when x = 0), the second operand is not evaluated, preventing the division-by-zero error.

64. $10\%3 = 1$ (the remainder when 10 is divided by 3) and $10/3 = 3$ (integer division discards the remainder). The print statements output the modulo result first, then the integer quotient.

65. The loop sums integers from 1 to 5: $1 + 2 + 3 + 4 + 5 = 15$, which is stored in `count` and then printed.

This is a classic loop-tracing problem testing accumulator pattern recognition.

66. Short-circuit evaluation of `OR` means that if the first operand (`y == 0`) is true, the second (`x / y > 1`) is never evaluated, preventing the division-by-zero runtime error. Because the condition is true, the branch executes and `x` becomes 3.

67. Overloading defines multiple methods sharing the same name but with different parameter signatures within one class, resolved at compile time; overriding redefines an inherited method in a subclass to change its behavior, resolved at runtime. Both are key OOP concepts but serve different purposes.

68. Encapsulation is the OOP principle of bundling data (attributes) and the methods that operate on that data together inside a class, restricting direct external access to internal state. This promotes data integrity and modular design.

69. An abstract class can contain both fully implemented methods and abstract methods (declared but not implemented), forcing subclasses to provide concrete implementations. A concrete class must implement all methods itself and cannot require subclasses to do so.

70. Operator precedence dictates that multiplication is performed before addition or subtraction: $4 \times 2 = 8$, then $3 + 8 - 1 = 10$. Misapplying left-to-right evaluation without respecting precedence would give the wrong answer.

71. A `do-while` loop executes its body first, then checks the condition, guaranteeing at least one execution. A `while` loop evaluates the condition before the first iteration, so if the condition is initially false the body never runs.

72. The while loop condition is `i <= 4` and `i` starts at 1, so the loop body executes for $i = 1, 2, 3, 4$—exactly 4 iterations—printing `"Hello"` each time. Incrementing `i` inside the loop ensures termination.

73. Polymorphism allows a single method call—`makeSound()`—to invoke different implementations depending on the actual runtime type of the object (Dog vs. Cat). This "many forms" behavior enables flexible, extensible code through a shared interface.

74. Starting at $x = 10$, the loop prints x then subtracts 3 each iteration: 10, 7, 4. After printing 4, x becomes 1, which satisfies $x \leq 1$ so the loop terminates before printing 1. The `repeat-until` loop checks the condition *after* the body.

75. In inheritance, a child (subclass) automatically acquires all attributes and methods of its parent (superclass) and may add new members or override inherited methods to specialize behavior. This "is-a" relationship enables code reuse and polymorphism.

76. Linear search examines each element one by one; in the worst case (target not present or at the last position), it inspects all n elements, giving $O(n)$ worst-case time. For 1,000 elements, that worst case is 1,000 comparisons.

77. Recursion solves a problem by having a function call itself with a simpler version of the problem until reaching a base case, whereas iteration uses explicit loop constructs (`for`/`while`) to repeat steps. Both strategies can solve the same problems but differ in structure and sometimes efficiency.

78. Binary search halves the search space with each comparison, yielding a worst-case of $O(\log n)$ comparisons. This logarithmic growth makes it dramatically faster than linear search for large sorted datasets.

79. Quicksort's worst-case occurs when the chosen pivot is always the smallest or largest element (e.g., already-sorted data with a bad pivot strategy), causing $O(n^2)$ comparisons. Its average-case is $O(n \log n)$, which is why it remains popular in practice.

80. Merge sort uses the divide-and-conquer strategy: it recursively splits the array in half, sorts each half, and merges the sorted halves. This guarantees $O(n \log n)$ time in all cases, unlike quicksort which degrades to $O(n^2)$ in the worst case.

81. The `mystery` function computes the Fibonacci sequence: `mystery(1) = 1`, `mystery(2) = mystery(1) + mystery(0) = 1`, `mystery(3) = 2`, `mystery(4) = 3`, `mystery(5) = 5`. Recognizing this classic recursive pattern is essential for algorithm tracing.

82. Binary search works by comparing the target to the middle element and eliminating half of the remaining elements on each step; this only works correctly if the array is already sorted. Applying binary search to an unsorted array produces undefined results.

83. Selection sort always performs $\Theta(n^2)$ comparisons because it scans the unsorted portion to find the minimum on every pass, regardless of the input's initial order. This makes its best, average, and worst cases all $O(n^2)$.

84. Every recursive function needs a base case—a condition that stops the recursion without another recursive call. Without a reachable base case, the function recurses indefinitely, eventually causing a stack overflow.

85. Bubble sort repeatedly compares adjacent pairs and swaps them if they are out of order, "bubbling" larger values toward the end of the array on each pass. Insertion sort builds a sorted subarray from left to right; selection sort finds the minimum on each pass.

86. The pseudocode finds the minimum element in the unsorted portion of the array on each outer loop iteration and swaps it to its correct position—this is the defining behavior of selection sort. Bubble sort compares adjacent elements; insertion sort shifts elements to insert each new item.

87. The AND operation returns true only when both operands are true; for the four combinations of P and Q, only T AND T yields T, giving the truth table T, F, F, F. This is the defining characteristic of logical conjunction.

88. De Morgan's Law states that `NOT (A OR B)` is logically equivalent to `NOT A AND NOT B`. Applying the law: the negation of a disjunction equals the conjunction of the negations.

89. By the absorption law of Boolean algebra, A AND $(A$ OR $B) = A$: if A is true the whole expression is true regardless of B; if A is false the whole expression is false regardless of B. This simplification is a standard Boolean algebra identity.

90. The number of ways to choose 4 from 10 is $\binom{10}{4} = \frac{10!}{4!\cdot 6!} = \frac{10\times 9\times 8\times 7}{4\times 3\times 2\times 1} = 210$. Combinations are used when order does not matter, as is the case when forming a committee.

91. The contrapositive of "If P then Q" is "If NOT Q then NOT P," and it is logically equivalent to the original statement. "If I do not carry an umbrella, then it is not raining" correctly applies the contrapositive to the given conditional.

92. Digital forensics focuses on recovering, preserving, and analyzing digital evidence from devices and networks, often to support legal proceedings or incident response. It is distinct from cybersecurity defense (prevention) or penetration testing (active probing).

93. A worm is self-replicating malware that spreads across networks without needing to attach itself to a host program, distinguishing it from a virus (which requires a host file) or ransomware (which encrypts data for extortion). Identifying malware types is part of cybersecurity literacy for CS educators.

94. Preserving the original state of evidence (e.g., creating a forensic image) before any analysis prevents accidental or intentional alteration that would compromise the integrity and admissibility of the evidence. This step is foundational to chain-of-custody principles.

95. A distance or proximity sensor allows a robot to detect how far away an obstacle is, enabling it to stop or change course before a collision. Cameras can also be used for obstacle detection but require more complex image-processing logic.

96. The engineering design process involves iterative cycles of evaluate-and-refine: after building and testing an initial prototype, the team analyzes what did not meet requirements and makes targeted improvements. This reflects the iterative, empirical nature of robotics and engineering design.

97. Collision detection is the computational process of determining when two or more game objects (sprites, hitboxes, meshes) overlap or touch, which triggers in-game responses like damage, bouncing, or stopping movement. It is a fundamental mechanic in virtually all 2D and 3D games.

98. Casting 7.9 to int truncates to 7, and casting 2.1 to int truncates to 2. The sum is 7 + 2 = 9.

99. Game design is the phase in which rules, objectives, win/loss conditions, and core mechanics are established—the conceptual blueprint before any coding or art assets are created. Implementation, testing, and deployment follow design.

100. In most 2D screen coordinate systems (and game engines), the origin (0, 0) is at the top-left corner and the y-axis increases downward, opposite to the Cartesian convention used in mathematics. Understanding this prevents common bugs in 2D game positioning code.

DOMAIN SCORE TRACKER

Domain	Questions	My Score	Percentage
I — Technology Applications Core	13 questions	___/13	__%
II — Program Design & Development	35 questions	___/35	__%
III — Programming Language Topics	38 questions	___/38	__%
IV — Specialized Topics	14 questions	___/14	__%
TOTAL	**100 questions**	**___/100**	**__%**