

## CHAPTER 07

# Introduction To Computational Thinking

### Complete Exercise Solutions

MCQs - Short Questions - Long Questions

9th Class | Computer Science & Entrepreneurship

[scholareye.xyz](https://scholareye.xyz)

## Multiple Choice Questions

Ticked & explained

### MCQ 1

Which of the following best defines computational thinking?

- (a) A method of solving problems using mathematical calculations only.
- (b) A problem-solving approach that employs systematic, algorithmic, and logical thinking. ✓**
- (c) A technique used exclusively in computer programming.
- (d) An approach that ignores real-world applications.

**Explanation:** Computational thinking is a broad cognitive skill. It is not just about math or exclusive to programming. It involves breaking down problems, recognizing patterns, abstracting details, and creating step-by-step solutions (algorithms) that can be applied to real-world scenarios across various disciplines.

## MCQ 2

Why is problem decomposition important in computational thinking?

(a) It simplifies problems by breaking them down into smaller, more manageable parts. ✓

(b) It complicates problems by adding more details.

(c) It eliminates the need for solving the problem.

(d) It is only useful for simple problems.

**Explanation:** Decomposition is the foundational step of computational thinking. Large, complex problems are overwhelming. By breaking them into smaller sub-problems, each part becomes easier to understand, solve, and test individually.

## MCQ 3

Pattern recognition involves:

(a) Finding and using similarities within problems ✓

(b) Ignoring repetitive elements

(c) Breaking problems into smaller pieces

(d) Writing detailed algorithms

**Explanation:** Pattern recognition is about observing trends, similarities, or repeating sequences. Once a pattern is identified, you can apply the same solution to similar instances, making problem-solving more efficient.

**MCQ 4**

**Which term refers to the process of ignoring the details to focus on the main idea?**

(a) Decomposition

(b) Pattern recognition

**(c) Abstraction** ✓

(d) Algorithm design

**Explanation:** Abstraction is the process of filtering out unnecessary information (details) to focus on the core, essential concepts. This helps in creating a general model of the problem without getting bogged down by complexities.

**MCQ 5**

**Which of the following is a principle of computational thinking?**

(a) Ignoring problem understanding

**(b) Problem simplification** ✓

(c) Avoiding solution design

(d) Implementing random solutions

**Explanation:** Problem simplification (which involves decomposition and abstraction) is a core principle of computational thinking. The other options (ignoring understanding, avoiding design, implementing random solutions) are counterintuitive to logical problem-solving.

**MCQ 6**

**Algorithms are:**

- (a) Lists of data
- (b) Graphical representations
- (c) Step-by-step instructions for solving a problem ✓**
- (d) Repetitive patterns

**Explanation:** An algorithm is a precise, unambiguous, and finite sequence of step-by-step instructions designed to solve a specific problem or perform a computation.

## MCQ 7

Which of the following is the first step in problem-solving according to computational thinking?

(a) Writing the solution

(b) Understanding the problem ✓

(c) Designing a flowchart

(d) Selecting a solution

**Explanation:** Before you can design a solution, write an algorithm, or code, you must completely understand the problem. This involves identifying inputs, outputs, constraints, and what a successful solution looks like.

## MCQ 8

Flowcharts are used to:

(a) Code a program

(b) Represent algorithms graphically ✓

(c) Solve mathematical equations

(d) Identify patterns

**Explanation:** Flowcharts use standard geometric symbols connected by arrows to visually represent the flow of control, decisions, and steps of an algorithm, making the logic easy to visualize.

**MCQ 9**

**Pseudocode is:**

(a) A type of flowchart

**(b) A high-level description of an algorithm using plain language** ✓

(c) A programming language

(d) A debugging tool

**Explanation:** Pseudocode uses structured, plain English (or another human language) to describe the logic of an algorithm. It is not a real programming language, so it doesn't have strict syntax rules, making it easy to write and convert to actual code later.

**MCQ 10**

**Dry running a flowchart involves:**

(a) Writing the code in a programming language

**(b) Testing the flowchart with sample data** ✓

(c) Converting the flowchart into pseudocode

(d) Ignoring the flowchart details

**Explanation:** A dry run (or desk check) is the manual process of stepping through the flowchart or algorithm using specific sample inputs to trace the execution and verify the logic without using a computer.

## Short Questions

Model answers

### Q 1

#### **Define computational thinking.**

Computational thinking is a systematic problem-solving process that involves formulating problems and their solutions in a way that a computer can effectively execute. It utilizes core skills such as decomposition, pattern recognition, abstraction, and algorithm design.

### Q 2

#### **What is decomposition in computational thinking?**

Decomposition is the process of breaking down a complex problem or system into smaller, more manageable parts or sub-problems that are easier to understand, solve, and test.

**Q 3****Explain pattern recognition with an example.**

Pattern recognition is the process of identifying similarities, trends, or repeating elements within a problem or across different problems. *Example:* In a program that calculates the area of different shapes, you might notice that the process for validating user input (ensuring it is a positive number) is a repeating pattern. This pattern can be recognized and handled with a single reusable function rather than rewriting the code for every shape.

**Q 4****Describe abstraction and its importance in problem-solving.**

Abstraction is the process of filtering out unnecessary details to focus on the essential, high-level aspects of a problem. It is important because it reduces complexity, allowing the problem-solver to focus on the core logic and design without being distracted by irrelevant details.

**Q 5****What is an algorithm?**

An algorithm is a precise, step-by-step set of instructions or rules designed to solve a specific problem or perform a computation. It must be finite (it must end), unambiguous, and effective.

**Q 6****How does problem understanding help in computational thinking?**

Problem understanding is the foundation of computational thinking. It helps identify the exact inputs, expected outputs, constraints, and core requirements. Without a clear understanding, subsequent steps like decomposition and algorithm design will be flawed and misdirected.

**Q 7****What are flowcharts and how are they used?**

Flowcharts are graphical representations of an algorithm or process. They use standard geometric symbols (e.g., ovals for start/end, rectangles for processes, diamonds for decisions) connected by arrows to show the flow of control. They are used to visualize, design, and communicate the logic of a process clearly before coding.

**Q 8****Explain the purpose of pseudocode.**

The purpose of pseudocode is to describe the logic of an algorithm in a structured, human-readable language (like English) that is independent of any specific programming language syntax. It acts as a bridge between the conceptual design (flowchart) and the actual code.

**Q 9****How do you differentiate between flowcharts and pseudocode?**

Flowcharts are visual and graphical, using symbols and arrows to represent logic, making them excellent for visualizing overall flow but harder to write for very complex logic. Pseudocode is text-based, using structured plain language, making it easier to write detailed logic and convert into actual code, but it lacks the immediate visual clarity of a flowchart.

**Q 10****What is a dry run and why is it important?**

A dry run (or desk checking) is the manual process of stepping through an algorithm or flowchart with sample input data to trace the execution. It is important because it helps identify logic errors, infinite loops, or incorrect conditions before the algorithm is translated into actual code, saving time and debugging effort.

**Q 11**

**Describe LARP and its significance in learning algorithms.**

In the context of learning algorithms, LARP stands for **L**ogic, **A**lgorithm, **R**epresentation, and **P**seudocode. It is a structured pedagogical approach. Its significance is that it forces learners to separate logical thinking from coding syntax. By mastering Logic, then Algorithms, then Representation (Flowcharting), and finally Pseudocode, students build a deep, fundamental understanding of how algorithms work, making them better programmers.

**Q 12**

**List and explain two debugging techniques.**

**1. Print Statement Debugging:** Inserting print statements into the code to output variable values at specific points to trace execution and find where the logic goes wrong.

**2. Dry Run/Desk Checking:** Manually tracing the code or flowchart with specific inputs to simulate execution and identify logical flaws.

## Long Questions

[Detailed answers](#)

### Q 1

**Write an algorithm to assign a grade based on the marks obtained by a student. The grading system follows these criteria:**

|              |    |
|--------------|----|
| 90 and above | A+ |
| 80 to 89     | A  |
| 70 to 79     | B  |
| 60 to 69     | C  |
| Below 60     | F  |

## Algorithm:

Step 1: Start

Step 2: Read marks

Step 3: If marks  $\geq 90$ ,  
then Grade = 'A+'

Step 4: Else if marks  $\geq 80$   
and marks  $\leq 89$ ,  
then Grade = 'A'

Step 5: Else if marks  $\geq 70$   
and marks  $\leq 79$ ,  
then Grade = 'B'

Step 6: Else if marks  $\geq 60$   
and marks  $\leq 69$ ,  
then Grade = 'C'

Step 7: Else if marks  $< 60$ ,  
then Grade = 'F'

Step 8: Print Grade

Step 9: Stop

## Q 2

**Explain how you would use algorithm design methods, such as flowcharts and pseudocode, to solve a complex computational problem. Illustrate your explanation with a detailed example.**

**Explanation:** Solving a complex problem begins with understanding the problem (Computational Thinking). Then, we break it down (Decomposition). Once the sub-problems are understood, we design algorithms for each part. Flowcharts are used to visually map out the control flow (decision points, loops) providing a high-level overview. Pseudocode is then written to flesh out the logic in plain language, which makes it easier to translate into actual code.

**Example:** Calculating the average of all even numbers in a list.

**Flowchart Logic:** Start -> Initialize sum=0, count=0 -> Loop through list -> Check if number is even -> Yes: add to sum, increment count -> No: skip -> After loop, check if count > 0 -> Calculate average = sum/count -> Print average -> End.

### Pseudocode:

```
SET sum = 0
SET count = 0
FOR each number IN list DO
    IF number MOD 2 == 0 THEN
        sum = sum + number
        count = count + 1
    END IF
END FOR
IF count > 0 THEN
    average = sum / count
    PRINT average
ELSE
    PRINT "No even numbers"
END IF
```

### Q 3

**Define computational thinking and explain its significance in modern problem-solving. Provide examples to illustrate how computational thinking can be applied in different fields.**

**Definition:** Computational thinking is a problem-solving process that involves expressing problems and their solutions in ways that a computer could execute. It encompasses skills like decomposition, pattern recognition, abstraction, and algorithm design.

**Significance:** It equips individuals with a systematic approach to tackle complex problems, fostering logical reasoning, creativity, and efficiency. It is not just for computer scientists but is a fundamental skill for everyone in the digital age.

**Examples in different fields:**

- *Biology:* Sequencing DNA (Pattern recognition, Decomposition).
- *Business:* Optimizing supply chains (Abstraction, Algorithm design).
- *Healthcare:* Diagnosing diseases based on symptoms (Pattern recognition).

**Q 4**

**Discuss the concept of decomposition in computational thinking. Why is it important?**

Decomposition is the process of taking a complex problem and breaking it into smaller, more manageable sub-problems.

**Importance:** It makes complex problems solvable. Smaller problems are easier to understand, design algorithms for, test, and debug. It also allows for parallel work (different people can work on different sub-problems) and code reusability (solving similar sub-problems with the same modules).

**Q 5**

**Explain pattern recognition in the context of computational thinking. How does identifying patterns help in problem-solving?**

Pattern recognition is identifying similarities, trends, or repeating sequences within a problem or across different problems.

**How it helps:** It allows problem-solvers to leverage previous solutions. If a new problem shares a pattern with a solved problem, the existing solution or algorithm can be adapted, saving time and effort. It helps in generalizing solutions, making them reusable.

**Q 6**

**What is an abstraction in computational thinking? Discuss its importance and provide examples of how abstraction can be used to simplify complex problems.**

Abstraction is the process of removing unnecessary details to focus on the essential features of a problem.

**Importance:** It reduces complexity, making it easier to conceptualize and solve problems. It allows us to build models and use them without needing to understand every underlying detail.

**Examples:** Driving a car (you use the steering wheel, gas, brakes, but ignore the complex engineering of the engine). In programming, using a function like `sort()` without knowing the specific sorting algorithm used inside.

## Q 7

**Describe what an algorithm is and explain its role in computational thinking. Provide a detailed example of an algorithm for solving a specific problem, and draw the corresponding flowchart.**

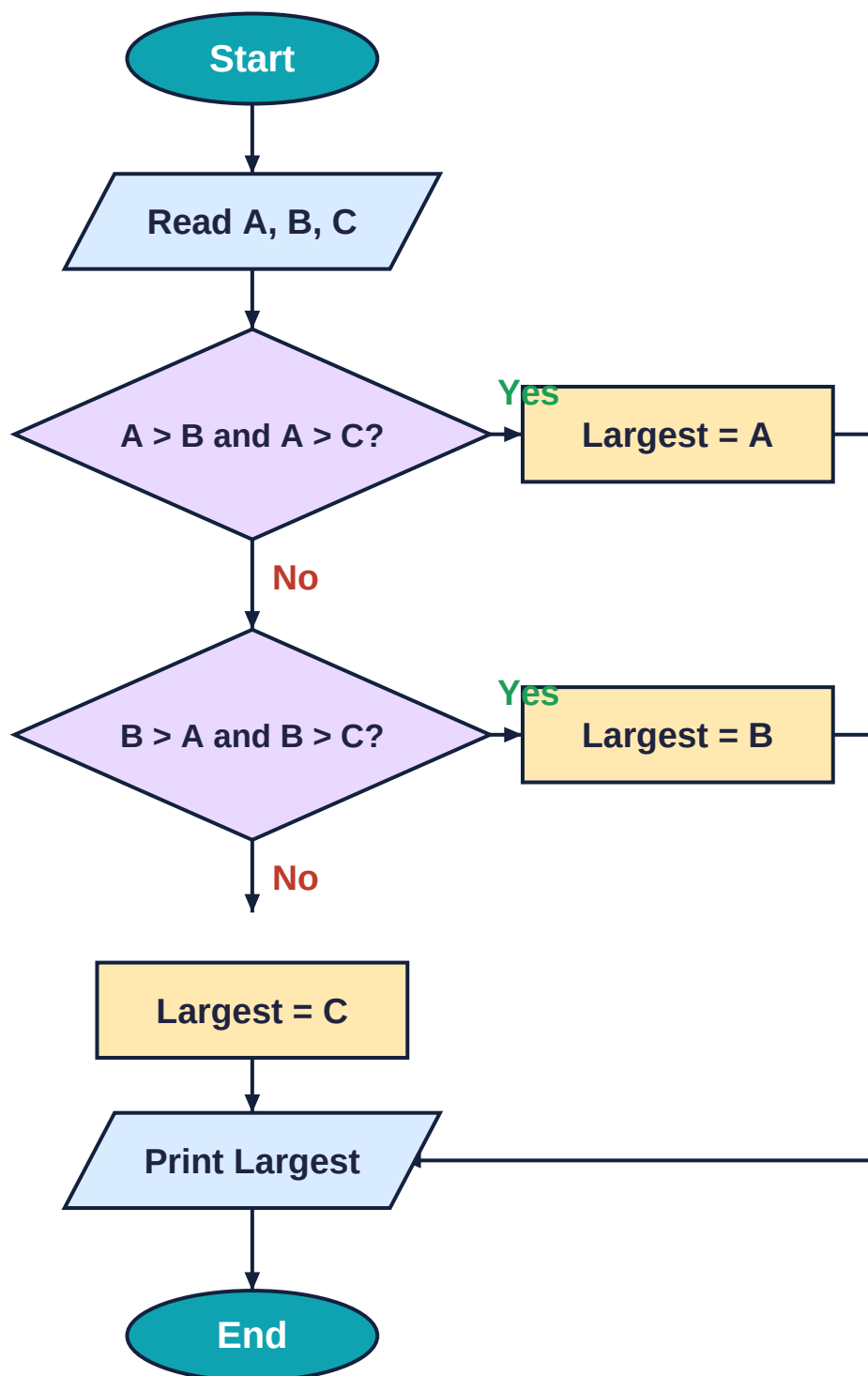
**Description & Role:** An algorithm is a precise, step-by-step set of instructions designed to solve a specific problem. Algorithms are the actionable output of computational thinking. They are the precise steps that can be translated into code.

**Example:** Finding the largest number among three numbers (A, B, C).

## Algorithm:

1. Start
2. Read A, B, C
3. If  $A > B$  and  $A > C$ ,  
then Largest = A
4. Else if  $B > A$  and  $B > C$ ,  
then Largest = B
5. Else, Largest = C
6. Print Largest
7. Stop

## Flowchart:



**Flowchart Representation (Textual description):** Oval Start  
-> Parallelogram Read A, B, C -> Diamond  $A > B$  and  $A > C$ ?  
-> Yes -> Rectangle Largest = A -> Diamond  $B > A$  and  $B > C$ ?  
-> Yes -> Rectangle Largest = B -> No -> Rectangle Largest = C -> Parallelogram Print Largest -> Oval End.

### Q 8

**Compare and contrast flowcharts and pseudocode as methods for algorithm design. Discuss the advantages and disadvantages of each method, and provide examples where one might be preferred over the other.**

**Flowcharts:** Visual, use standard symbols. *Advantages:* Easy to understand overall flow, great for visual learners, clearly shows branching and looping. *Disadvantages:* Hard to modify for complex logic, takes up a lot of space, drawing tools required.

**Pseudocode:** Text-based, plain language. *Advantages:* Easy to write and modify, focuses purely on logic, easy to translate to code. *Disadvantages:* No visual representation, might be ambiguous if not written strictly.

**Preference:** Flowcharts are preferred for presenting logic to non-technical stakeholders or for simpler, high-level overviews. Pseudocode is preferred by programmers during the detailed design phase before coding.

## Q 9

**Explain the concept of a dry run in the context of both flowcharts and pseudocode. How does performing a dry run help in validating the correctness of an algorithm?**

A dry run (or desk checking) is the manual process of stepping through an algorithm or flowchart with sample input data to trace the execution.

**Validation:** It validates correctness by acting as a manual simulation. It ensures all variables are initialized correctly, loops terminate, and conditional branches lead to the expected outputs. It catches logical errors (e.g., using `>` instead of `>=`) before the expense of coding and debugging.

## Q 10

**What is LARP? Discuss its importance in learning and practicing algorithms.**

In this educational context, LARP stands for **L**ogic, **A**lgorithm, **R**epresentation, and **P**seudocode. It is a structured pedagogical approach.

**Importance:** It forces learners to separate logical thinking from coding syntax. By step-by-step mastering Logic, then Algorithms, then Representation (Flowcharting), and finally Pseudocode, students build a deep, fundamental understanding of how algorithms work, making them better programmers.

## Q 11

**How does LARP enhance the understanding and application of computational thinking principles? Provide a scenario where LARP can be used to improve an algorithm.**

LARP enhances understanding by breaking down the abstract concepts of Computational Thinking (decomposition, abstraction, etc.) into a concrete, actionable cycle.

**Scenario:** A student wants to build a weather prediction app. Instead of jumping into code, they use LARP:

- **Logic:** Understand that weather depends on temperature, humidity, pressure.
- **Algorithm:** Design steps to collect data, apply a rule (if humidity > 80% and pressure < 1000mb, predict rain), and output prediction.
- **Representation:** Draw a flowchart showing data input, decision diamonds, and output.
- **Pseudocode:** Write the plain language steps.

By doing this, they improve the algorithm iteratively before writing a single line of actual programming code.



# THANK YOU!

**Keep thinking. Keep solving. Keep growing.**

Chapter 07: Introduction To Computational Thinking  
9th Class Computer Science & Entrepreneurship

**[scholareye.xyz](https://scholareye.xyz)**

More notes, MCQs & solutions available online.