UNIT 07

Computational Thinking

Short introduction of Unit

Computational thinking is an essential skill that enables individuals to solve complex problems using methods that align with processes involved in computer science. This chapter begins by defining computational thinking and breaking it down into its fundamental components, decomposition, pattern recognition, abstraction and algorithms.

Q.1 Define Computational thinking and explain its significance in modern problem-solving. Provide examples to illustrate how computational thinking can be applied in different fields. 09507001

Ans. Computational Thinking (CT) is a problem-solving process that involves a set of skills and techniques to solve complex problems in a way that can be executed by a computer. This approach can be used in various fields beyond computer science, such as biology, mathematics, and even daily life.

Let's break down computational thinking into its key components:

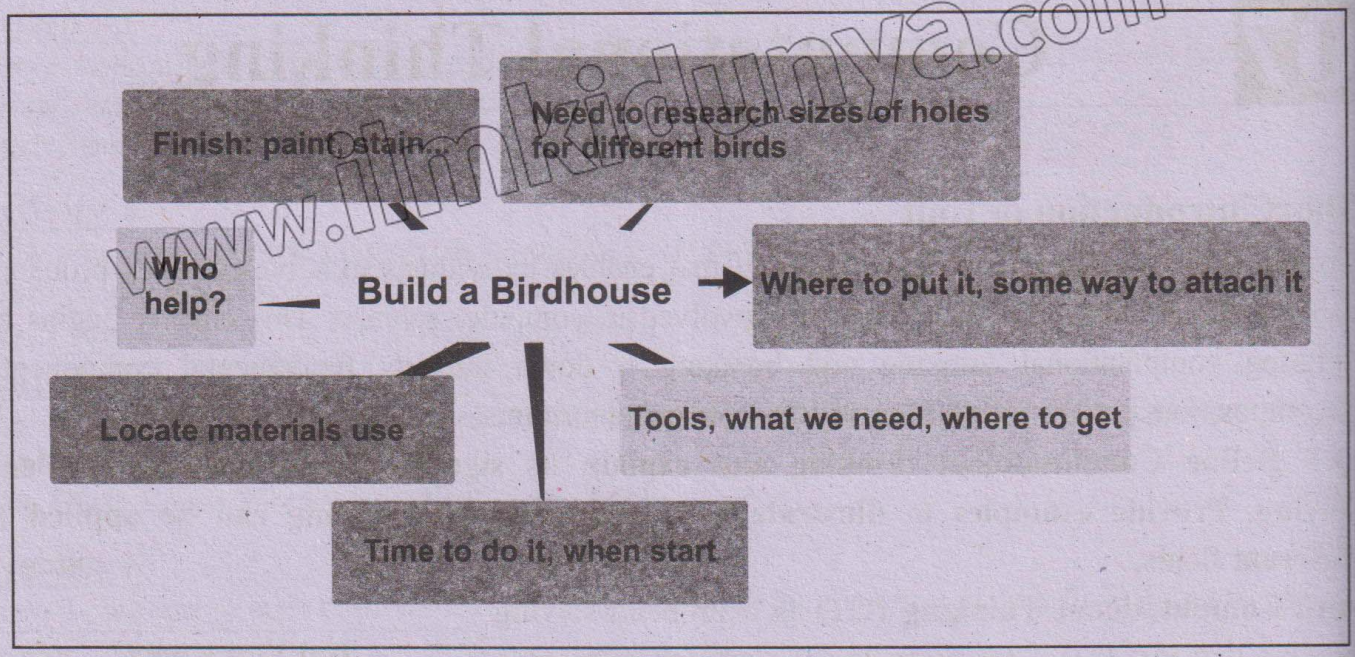
Did You Know?

Computational thinking is not limited to computer science. It is used in everyday problem solving, such as planning a trip or organizing tasks.

Decomposition

Decomposition is the method of breaking down a complicated problem into smaller, more convenient components. Decomposition is an important step in computational thinking. It involves dividing a complex problem into smaller, manageable tasks. Let's take the example of building a birdhouse. This task might look tough at first, but if we break it down, we can handle each part one at a time. Here's how we can decompose the task of building a birdhouse.

- **Design the Birdhouse:** Decide on the size, shape, and design. Sketch a plan and gather all necessary measurements.
- **Gather Materials:** List all the materials needed such as wood, nails, paint, and tools like a hammer and saw.
- **Cut the Wood:** Measure and cut the wood into the required pieces according to-the design.
- **Assemble the Pieces:** Follow the plan to assemble the pieces of wood together to form the structure of the birdhouse.
- **Paint and Decorate:** Paint the birdhouse and add any decorations to make it attractive for birds.
- **Install the Birdhouse:** Find a suitable location and securely install the birdhouse where birds can easily access it.



Q.2 Explain pattern recognition in the context of computational thinking. How does identifying patterns help in problem-solving? 09507002

Ans. Pattern recognition involves looking for similarities or patterns among and within problems. For instance, if you notice that you always forget your homework on Mondays, you might recognize a pattern and set a reminder specifically for Sundays.

Pattern recognition is an essential aspect of computational thinking. It involves identifying and understanding regularities or patterns within a set of data or problems. Let's consider the example of recognizing patterns in the areas of squares.

The upper row in Figure represents the side lengths of squares, ranging from 1 to 7. The lower row shows the corresponding areas of these squares. Here, we can observe a pattern in how the areas increase.

- Side Length 1: Area = $1^2 = 1$
- Side Length 2: Area = $2^2 = 4$ (1+3)
- Side Length 3: Area = $3^2 = 9$ (1+3+5)
- Side Length 4: Area = $4^2 = 16$ (1+3+5+7)
- Side Length 5: Area = $5^2 = 25$ (1+3+5+7+9)
- Side Length 6: Area = $6^2 = 36$ (1+3+5+7+9+11)
- Side Length 7: Area = $7^2 = 49$ (1+3+5+7+9+11+13)

Visual/Numerical Pattern

Goes up by 1

	+1	+1	+1	+1	+1	+1	
Side	1	2	3	4	5	6	7
Area	1	4	9	16	25	36	47
	+3	+5	+7	+9	+11	+13	

Goes up by consecutive odd numbers starting at 3

We can see that the area of each square can be calculated by adding consecutive odd numbers. For example, the area of a square with a side length of 3 can be found by adding the first three odd numbers: 1 + 3 + 5 = 9.

Q.3 What is an abstraction in computational thinking? Discuss its importance and provide example of how abstraction can be used to simplify complex problems. 09507003

Ans. Abstraction is a fundamental concept in problem solving, especially in computer science. It involves simplifying complex problems by breaking them down into smaller, more manageable parts, and focusing only on the essential details while ignoring the unnecessary ones. This helps in understanding, designing, and solving problems more efficiently.

Definition

Abstraction is the process of hiding the complex details while exposing only the necessary parts. It helps reduce complexity by allowing us to focus on the high-level overview without getting lost in the details.

Example: Making a Cup of Tea:

1. Boil water.
2. Add tea leaves or a tea bag.
3. Steep for a few minutes.
4. Pour into a cup and add milk/sugar if desired.

Q.4 Describe what an algorithm is and explain its role in computational thinking. Provide a detailed example of an algorithm for solving a specific problem. 09507004

Ans. An algorithm is a step-by-step collection of instructions to solve a problem or complete a task similar to following a recipe to bake a cake. An algorithm is a precise sequence of instructions that can be followed to achieve a specific goal, like a recipe or a set of directions that tells you exactly what to do and in what order:

Example: Planting a Tree: Here is a simple algorithm to plant a tree, an activity that can be very meaningful and beneficial:

1. Choose a suitable spot in your garden.
2. Dig a hole that is twice the width of the tree's root ball.
3. Place the tree in the hole, making sure it is upright.
4. Fill the hole with soil, pressing it down gently to remove air pockets.
5. Water the tree generously to help it settle.
6. Add much around the base of the tree to retain moisture
7. Water the tree regularly until it is established.

This algorithm gives clear instructions on how to plant a tree, making it easy to follow for anyone.

Q.5 Explain the Principles of Computational Thinking in details: 09507005

Ans. Computational thinking involves several key principles that guide the process of problem-solving in a structured manner.

Problem Understanding

Understanding a problem involves identifying the core issue, defining the requirements, and setting the objectives. Understanding the problem is the first and most important step in problem-solving, especially in computational thinking. This involves thoroughly analyzing the problem to identify its key components and requirements before attempting to find a solution.

Importance of Problem Understanding

- **Clarity and Focus:** By fully understanding the problem, you gain clarity on what needs to be solved. This helps you focus on the right aspects without getting distracted by irrelevant details.
- **Defining Goals:** Proper understanding of the problem allows you to define clear and achievable goals. You can determine what the final outcome should look like and set specific objectives to reach that outcome.
- **Efficient Solutions:** When you understand the problem well, you can devise more efficient and effective solutions. You can choose the best methods and tools to address the problem, saving time and resources.
- **Avoiding Mistakes:** By thoroughly understanding the problem, you can avoid common pitfalls and mistakes. Misunderstanding the problem can lead to incorrect solutions and wasted effort.

Example: Building a School Website:

Imagine you are asked to build a website for your school. Before jumping into coding, you need to understand the problem:

1. **Identify Requirements:** What features does the website need? For example, pages for news, events, class schedules, and contact information.
2. **User Needs:** Who will use the website? Students, teachers, parents? Understanding your audience helps in designing user-friendly interfaces.
3. **Technical Constraints:** What resources and tools are available? Do you have access to a web server and the necessary software?

By understanding these aspects, you can plan and build a website that meets the needs of your school community.

Problem Simplification

Simplifying a problem involves breaking it down into smaller, more manageable sub-problems. Example: To design a website, break down the tasks into designing the layout, creating content, and coding the functionality.

Solution Selection and Design

Choosing the best solution involves evaluating different approaches and selecting the most efficient one. Designing the solution requires creating a detailed plan or algorithm.

Q.6 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. 09507006

Ans. Algorithm design methods provide a range of tools and techniques to tackle various computational problems effectively. Each method has its strengths and weaknesses, making it suitable for different types of problems. Understanding different methods allows one to choose the most appropriate approach for a given problem, leading to more efficient and elegant solutions. Let's discuss two of these methods.

Flowcharts

Flowcharts are visual representations of the steps in a process or system, Flowcharts depicted using different symbols connected by arrows. They are widely used in various fields, including computer science, engineering, and business, to model processes, design systems, and communicate complex workflows clearly and effectively.

Importance of Flowcharts

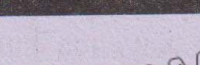
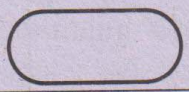
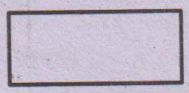
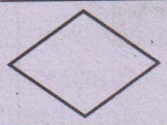
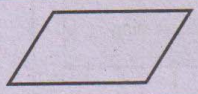
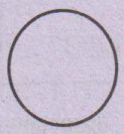
- **Clarity:** Flowcharts provide a clear and concise way to represent processes, making them easier to understand at a glance.
- **Communication:** These are excellent tools for communicating complex processes to a wide audience, ensuring everyone has a common understanding.
- **Problem Solving:** Flowcharts help identify bottlenecks and inefficiencies in a process, aiding in problem-solving and optimization.
- **Documentation:** They serve as essential documentation for systems and processes, which is useful for training and reference purpose.

Did you know?

The first standardized flowchart symbols were developed in **1947** by the **American National Standard Institute (ANSI).**

Flowchart Symbols

Flowchart symbols are visual representations used to illustrate the steps and flow of a process or system.

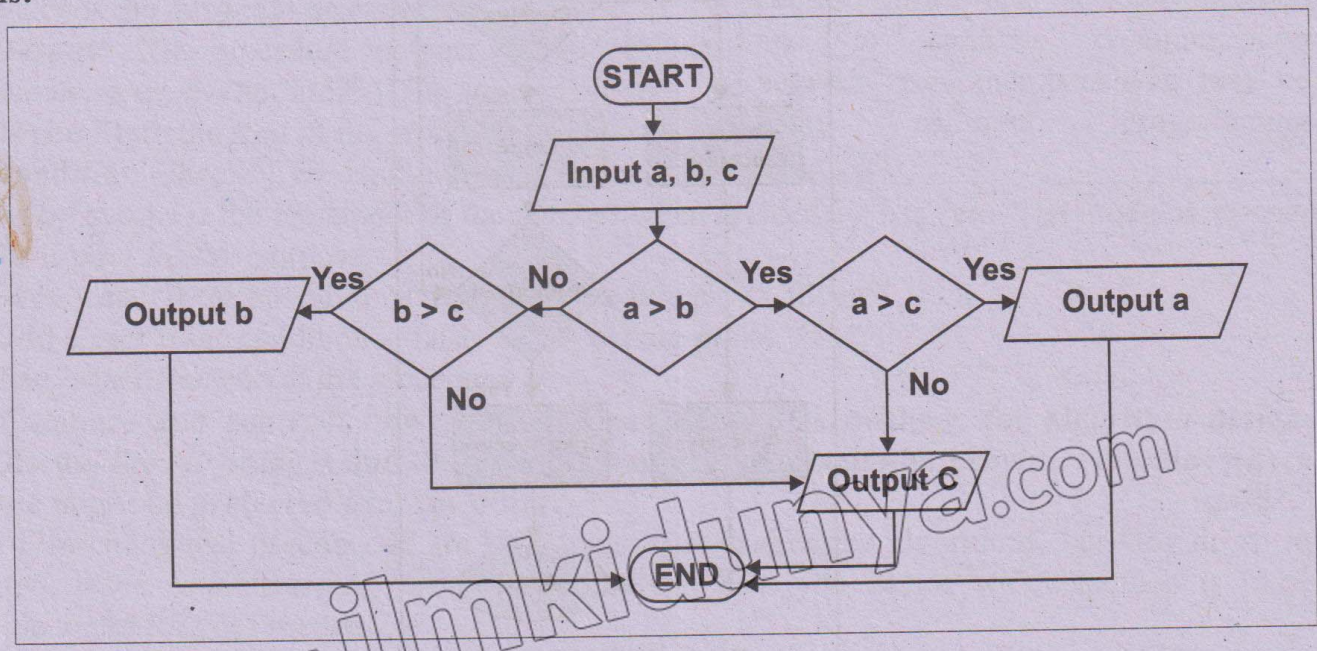
Symbol	Name	Description
	Flow line	This symbol shows the flow of data, information or process in the flowchart.
	Terminal	An oval shape symbol that represents the Start and End of a flowchart.
	Process	A Rectangle shape symbol is used to represent the process or action taken or shows all the calculations.
	Decision	A diamond represents a decision symbol used for comparison or a decision. It changes the flow of control and decides a particular path to be followed.
	Input/output	A parallelogram represents either input or output operation regardless of the input or output method.
	Connector	A small circle represents a connector symbol and is used to join various parts of a flow chart. Connectors are used when flow chart is very large and complex and the numbers inside them identify their links. Control is transferred from one connector to another with the same number in a program.

Q.7 Draw a flow chart to display the large one out of three given unequal numbers. 09507007

OR

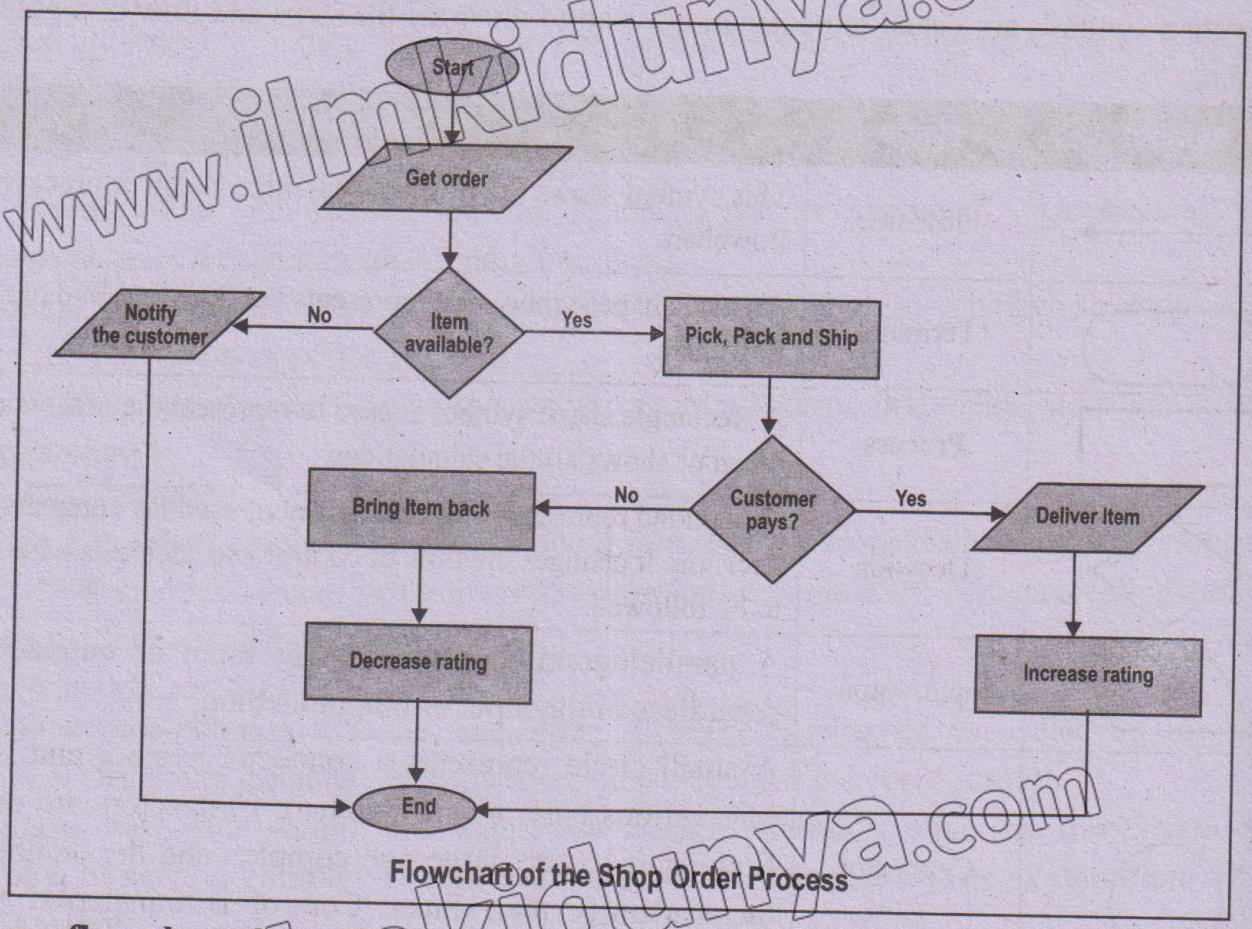
Draw a flowchart to find largest of three numbers.

Ans:



Q.8 Draw a flow chart to show the process of order from a shop. 09507008

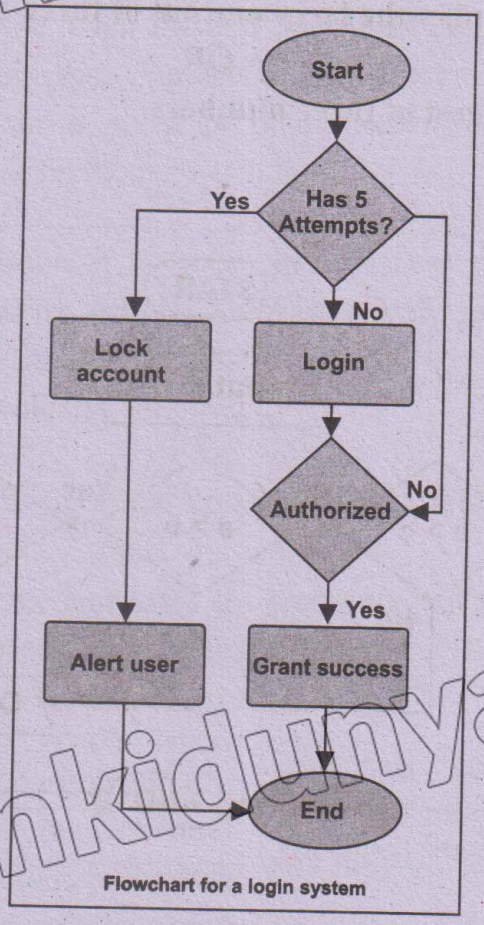
Ans.



Flowchart of the Shop Order Process

Q.9 Draw a flow chart that will show login process to a system. 09507009

Ans.



Flowchart for a login system

Q.10 Provide a detailed explanation of pseudocode, along with relevant examples to illustrate its use effectively. 09507010

Ans. Pseudocode is a method of representing an algorithm using simple and informal language that is easy to understand. It combines the structure of programming clarity with the readability of plain English, making it a useful tool for planning and explaining algorithms. Pseudocode is not actual code that can be run on a computer, but rather a way to describe the steps of an algorithm in a manner that is easy to follow..

Example-1

Determining whether a number is even or odd is a fundamental task in programming and computer science. An even number is divisible by 2 without any remainder, whereas an odd number has a remainder of 1 when divided by 2. Below is the pseudocode for this process, followed by an explanation.

Algorithm 1

Pseudocode for determining if a number is even or odd.

1. Procedure Check Even Odd(number)
2. Input: number (The number to be checked)
3. Output: "Even" if number is even, "Odd" if number is odd
4. Begin
5. if (number % 2 == 0) then
6. print "Even"
7. else
8. print "Odd"
9. End if
10. End

Explanation

1. **Procedure Declaration:** The pseudocode begins with the declaration of the procedure 'Check Even Odd' which takes a single input, 'number'.
2. **Input:** The procedure accepts a variable 'number' which is the integer to be checked.
3. **Output:** The procedure outputs "Even" if the number is even, and "Odd" if the number is odd.
4. **Begin:** Mark the start of the procedure.
5. **Condition Check:** The condition 'if (number % 2 == 0)' checks if the remainder of the number when divided by 2 is zero. The modulus operator '%' is used for this purpose.
6. **Even Case:** If the condition is true, the procedure prints "Even".
7. **Odd Case:** If the condition is false, the procedure prints "Odd".
8. **End:** Marks the end of the procedure.

Did you know?

Pseudocode is often used in software development before writing the actual code to ensure that the logic is sound and to facilitate communication between team members who may be using different programming languages.

Q.11 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. 09507011

Ans. Flowcharts and pseudocode are both tools used to describe algorithms, but they do so in different ways. Understanding their differences can help you decide which method is more suitable to use for your scenario.

Pseudocode	Flowcharts
• Pseudocode uses plain language and structured format to describe the steps of an algorithm. • It is read like a story, with each step is written out sequentially. • Pseudocode communicates the steps in a detailed, narrative-like format. • It is particularly useful for documenting algorithms in a way that can be easily converted into actual code in any programming language.	• Flowcharts use graphical symbols and arrows to represent the flow of an algorithm. • It is like watching a movie, where each symbol (such as rectangles, diamonds, and ovals) represents a different type of action or decision, and arrows indicate the connection and direction of the flow. • Flowchart communicates the process in a visual format, which can be more easy for understanding the overall flow and structure. • They are useful for identifying the steps and decisions in an algorithm at a glance.

Example:

Algorithm: Presents the pseudocode for checking a valid username and password.

1. Procedure Check Credentials (username, password)
2. Input: username, password
3. Output: Validity message
4. Begin
5. validUsername = "user123" (Replace with the actual valid username)
6. validPassword = "pass123" (Replace with the actual valid password}
7. if (username = = validUsername) then
8. if (password == valid Password) then
9. print "Login successful"
10. else
11. print "Invalid password"
12. end if
13. else
14. print "Invalid username"
15. end if.
16. End

Q.12 Describe Basic Algorithm Design and Evaluation Techniques. 09507012

OR

Provide a detailed explanation of the fundamental techniques used in algorithm design and evaluation.

Ans. Techniques to essential algorithms are essential to understand how efficiently they solve problems. In this section, we will explore different techniques for evaluating algorithms, focusing on their time and space complexities.

Did you know?

Some algorithms can perform the same task much faster than others. For example, sorting a list of 100 items might take one algorithm 1 second and another algorithm 10 seconds.

1. Time Complexity

Time Complexity measures how fast or slow an algorithm performs. It shows how the running time of an algorithm changes as the size of the input increases.

Here's an easy way to understand it:

Imagine you have a list of names, and you want to find a specific name. If you have 10 names, it might only take a few seconds to look through the list. But what if you have 100 names? Or 1,000 names? The time it takes to find the name increases as the list gets longer. Time complexity helps us understand this increase.

2. Space Complexity

Space complexity measures the amount of memory an algorithm uses relative to input size. It is essential to consider both the memory required for the input and any extra memory used by the algorithm.

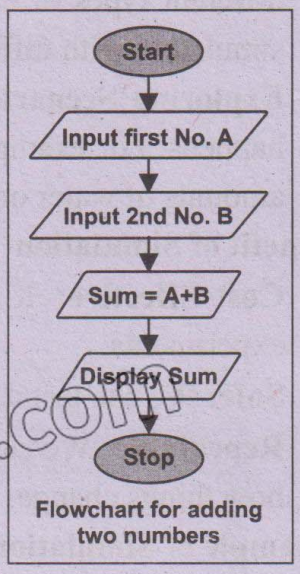
Q.13 Explain the concept of a dry run in the context of both flowcharts and pseudocode.

OR

How does performing a dry run help in validating the correctness of an algorithm? 09507013

Dry Run

A dry run involves manually going through the algorithm with sample data to identify any errors.



Flowchart for adding two numbers

Dry Run of a Flowchart

A dry run of a flowchart involves manually walking through the flowchart step-by-step to understand how the algorithm works without using a computer. This helps identify any logical errors and understand the flow of control.

Example: Calculating the Sum of Two Numbers: Consider the flowchart given in figure for adding two numbers:

Steps to dry run this flowchart:

1. Start
2. Input the first number (e.g., 3)
3. Input the second number (e.g., 5)
4. Add the two numbers (3 + 5 = 8)
5. Output the result (8)
6. Stop

Dry Run of Pseudocode

A dry run of pseudocode involves manually simulating the execution of the pseudocode line-by-line.

This helps in verifying the logic and correctness of the algorithm.

Did you know?

Dry running your code or algorithm helps catching errors early in the development process, saving time and effort.

Example: Finding the Maximum of Two Numbers

Consider the pseudocode for finding the maximum of two numbers:

Algorithm: Find Maximum

1. Input: num1, num2
2. if num1> num2 then
3. max = num1
4. else
5. max = num2
6. end if
7. Output: max

Step to dry run this pseudocode:

1. Input num1 and num2 (e.g., 10 and 15)
2. Check if num1 > num2 (10>15: False)

3. Since the condition is False, max = num2 (max = 15)
4. Output max (15)

Q.14 Explain the basic concept of Simulation. Also describe its benefits with suitable examples. 09507014

Ans. Simulation is use of computer programs to create a model of a real-world process or system. This helps us understand how things work by testing different ideas or algorithms without needing to try them out in real life.

Use of Simulation

1. **Testing Algorithms:** We can use simulation to see how well an algorithm works with different types of data. For example, if we want to test a new way to sort numbers, we can simulate it with different sets of numbers to see how fast it is.
2. **Exploring Scenarios:** Simulation allows us to create many different situations to see what happens. For example, in a science experiment about plant growth, we can simulate different amounts of water or sunlight to find out which conditions help plants grow best.

Benefit of Simulation

- **Cost-Effective:** It is often cheaper and faster to run simulations than to conduct real experiments.
- **Safe:** We can test dangerous situations, like a fire in a building, without putting anyone at risk.
- **Repeatable:** We can run the same simulation multiple times with different settings to observe how things change.

Example of Simulation

1. **Weather Forecasting:** Meteorologists use simulations to predict the weather. They input data about temperature, humidity, and wind speed into a computer model to see how the weather might change over the next few days.
2. **Traffic Flow:** City planners can simulate traffic to see how changes to roads or traffic lights might affect the flow of cars. This helps them design better roads and reduce traffic jams.

Q.15 What is LARP? Discuss its importance in learning and practicing algorithms. 09507015

Ans. LARP stands for **Logic of Algorithms for Resolution of Problems.** It is a fun and interactive way to learn how algorithms work by actually running them and seeing the results. Think of it as a playground where you can experiment with different algorithms and understand how they process data.

Did you know?

For the latest versions and updates of **LARP** software, check trusted educational and coding platforms, or search for "**LARP** software download" on your favorite search engine.

Why is LARP Important?

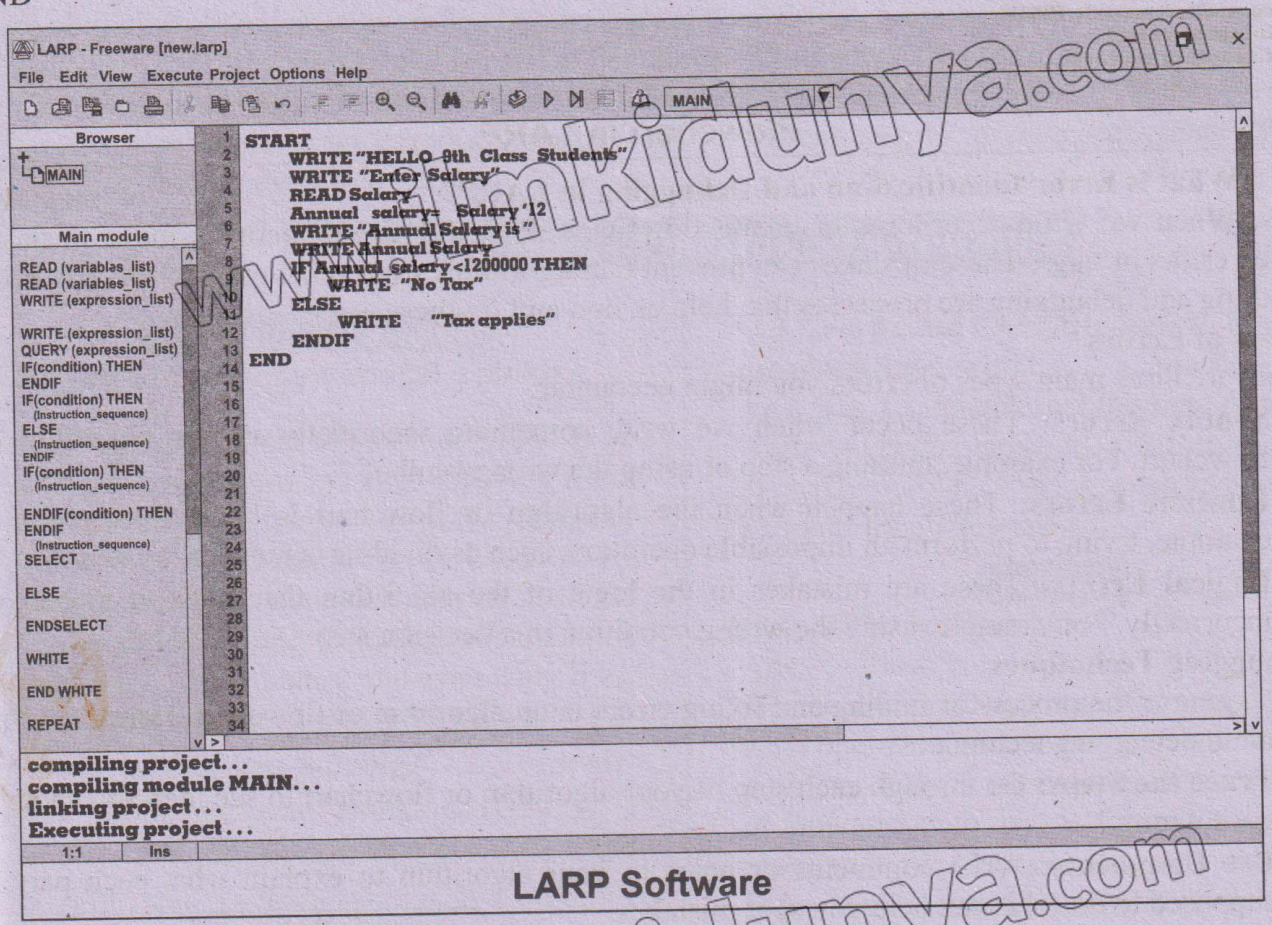
LARP helps you:

- Understand how algorithms work. For instance, which illustrates an algorithm designed to determine the applicability of tax on the annual salary of a person.
- See the effect of different inputs on the output.
- Practice writing and improving your own algorithms.

Writing Algorithms

Writing algorithms using LARP involves a structured and simplified approach to developing logical solutions for computational problems. LARP employs a clear syntax that begins with a START command and ends with an END command, ensuring that each step of the algorithm is easy to follow. Within this framework, instructions are provided in a straightforward manner, such as using WRITE to display messages, READ to input values, and conditional statements like IF...THEN...ELSE to handle decision-making processes. Here's an example of a simple algorithm to check if a number is even or odd.

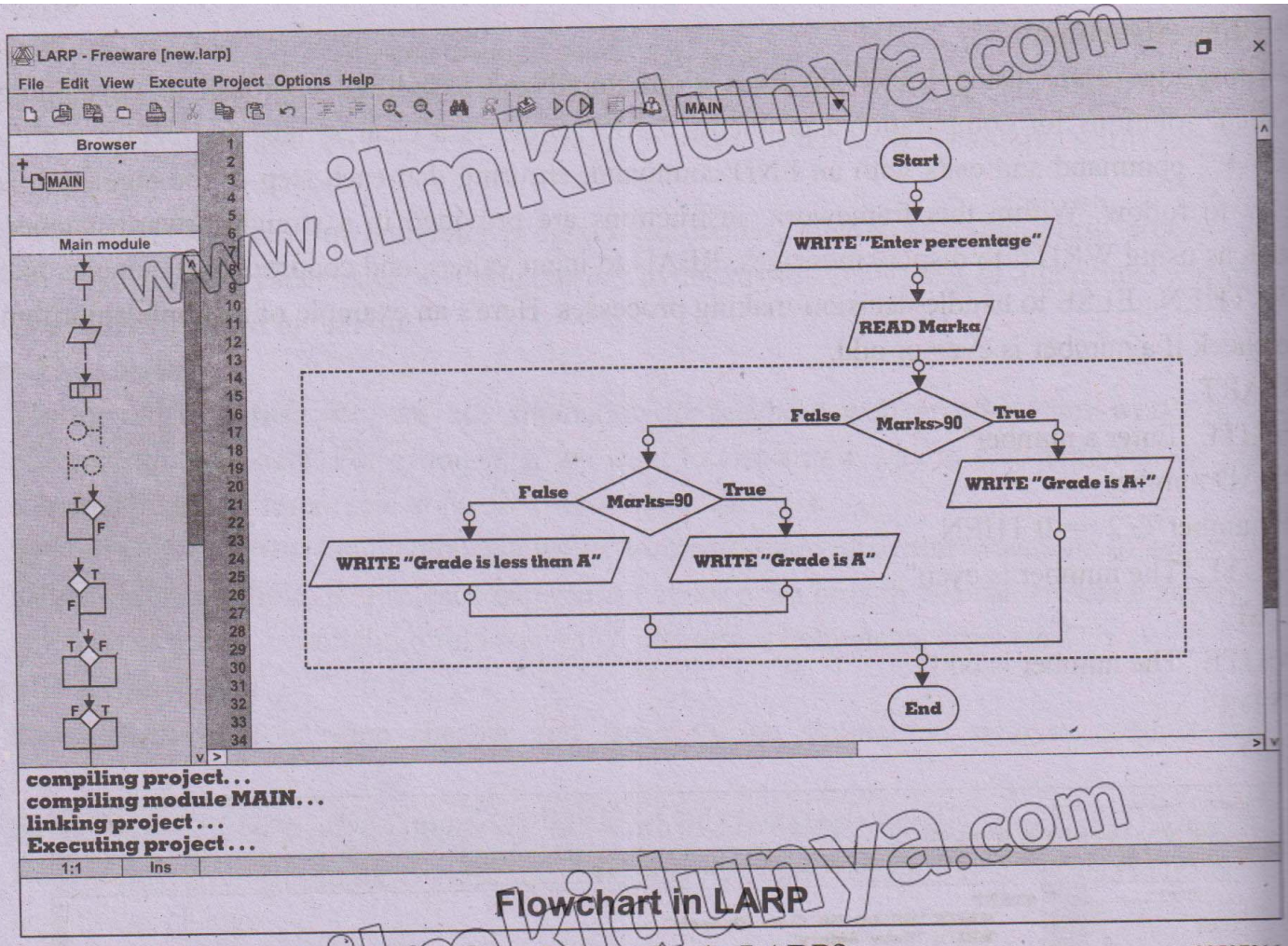
```
START
WRITE "Enter a number"
READ number
IF number % 2 == 0 THEN
WRITE "The number is even"
ELSE
WRITE "The number is odd"
ENDIF
END
```



LARP Software

Drawing Flowcharts in LARP:

Drawing flowcharts in LARP involves visually representing the algorithm's steps using standard flowchart symbols such as rectangles for process, diamonds for decisions, and parallelogram for input/output operations.



Flowchart in LARP

Q.16 What is Error Identification and Debugging in LARP? 09507016

Ans. When we write algorithms or create flowcharts in LARP, we sometimes make mistakes called errors or bugs. These mistakes can prevent our algorithms from functioning correctly. Error handling and debugging are processes that help us find and fix these errors.

Types of Errors

There are three main types of errors you might encounter:

- **Syntax Errors:** These occur when we write something incorrectly in our algorithm or flowchart. For example, missing a step or using the wrong symbol.
- **Runtime Errors:** These happen when the algorithm or flowchart is being executed. For example, trying to perform an impossible operation, such as dividing a number by zero.
- **Logical Errors:** These are mistakes in the logic of the algorithm that cause it to behave incorrectly. For example, using the wrong condition in a decision step.

Debugging Techniques

Debugging is the process of finding and fixing errors in an algorithm or flowchart. Here are some common debugging techniques:

- **Trace the Steps:** Go through each step of your algorithm or flowchart to see identity where it goes wrong.
- **Use Comments:** Write comments or notes in your algorithm to explain what each part is supposed to do. This can help you spot mistakes.
- **Check Conditions:** Ensure that all conditions in decision steps are correct.
- **Simplify the Problem:** Break down the algorithm into smaller parts and test each part separately.

Did you know?

Always read error messages carefully. They often tell you exactly where the problem is.

Common Error Message in LARP

Here are some common error messages you might see in LARP and what they mean:

- **Missing Step:** You probably forgot to include an important step in your algorithm.
- **Undefined Variable:** You are using a variable that hasn't been defined yet.
- **Invalid Operation:** You are trying to perform an operation that is not allowed, like dividing by zero.

Did you know?

The term **"debugging"** comes from an actual bug-a moth that was found causing problems in an early computer. The moth was removed, and the process was called **"debugging"**

Q.17 How does LARP enhance the understanding and application of computational thinking principles? Provide a scenario where LARP can be used to improve an algorithm. 09507017

Ans. LARP (Logic of Algorithms for Problem Resolution) improves knowledge and implementation of computational thinking principles by taking an immersive, interactive, and collaborative approach to problem solving. Participants gain hands-on experience decomposing complicated issues, identifying patterns, abstracting pertinent information, and designing successful algorithms through role-playing situations that replicate real-world challenges. LARP makes abstract topics tangible by requiring participants to actively create and test solutions while receiving quick feedback, which encourages incremental improvements. This strategy focuses on collaboration and adaptation, reflecting the dynamic nature of computational problem solving.

Consider a case in which LARP is used to develop a sorting algorithm for organizing volumes in a library based on height. Participants assume roles such as "librarian," "books," and "observers." The librarian employs an initial algorithm, such as bubble sort, to arrange participants (books) according to their displayed height.

Topic Wise Short Questions (Additional)

Computational Thinking

Q.1 Introduce Computational Thinking. 09507018

Ans. Computational Thinking (CT) is a problem-solving process that involves a set of skills and techniques to solve complex problems in a way that a can be executed by a computer. This approach can be used in various fields beyond computer science, such as biology, mathematics, and even daily life.

Q.2 How Computational Thinking is helpful in Problem-Solving? 09507019

Ans. Computational Thinking (CT) is a problem-solving process that involves a set of skills and techniques to solve complex problems in a way that a can be executed by a computer. This approach can be used in various fields beyond computer science, such as biology, mathematics, and even daily life.

Q.3 Can we use Computational Thinking for planning a trip? 09507020

Ans. Yes, Computational thinking is not limited to computer science. It is used in everyday problem solving, such as planning a trip or organizing tasks.

Flowchart

Q.4 What is Flowchart?

Ans. Flowchart is a diagrammatic representation of an algorithm. It describes what operations are required to solve a given problem. Flowchart illustrates the sequence of operations to be performed to solve a problem in the form of a diagram. 09507021

Q.5 Which requirements must determine by the developer while creating a flowchart? 09507022

Ans. The flowchart developer must determine the following requirements for

the given problem or algorithm before drawing a flowchart.

- Start of the flowchart
- Input to the flowchart
- Type of processing required
- Decision to be taken
- Output of the operation
- End of the flowchart

Q.6 Which software tools are required for designing flowchart? 09507023

Ans. For flowchart designing, different software tools are available to design the flowcharts. Some of the famous tools are Microsoft Visio and LARP software.

- **Microsoft Visio**
- **LARP (Logics of Algorithms and Resolution of Problems)**

Q.7 How to create flowchart while using Microsoft Visio? 09507024

Ans. Following steps will be followe while creating flowchart.

- Start Microsoft Visio
- Click on the category of Flowchart
- Double-click the Basic Flowchart
- For each step, you wish to design in th process, drag a relevant flowchar symbol and place itonto your drawing.
- Connect the flowchart shapes b holding the mouse pointer over the firs symbol, and thenreleasing it on the to th other symbol you wish to connect to.
- To place text into a shape, select it, an then type.

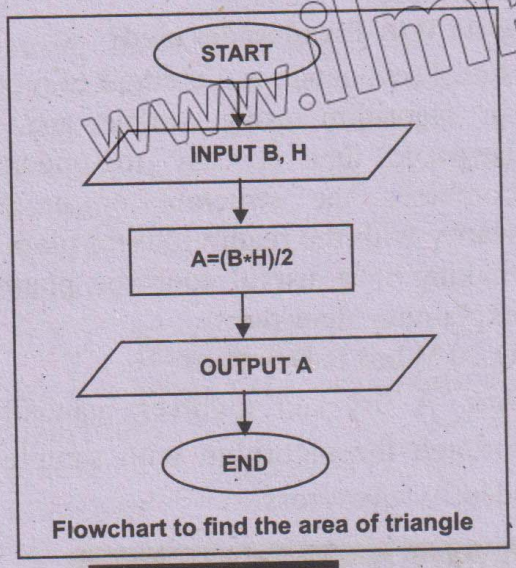
Q.8 Draw the symbols of flowchart with descriptions. 09507025

Ans. Flowchart symbols

Symbol	Name	Description
	Flow line	This symbol shows the flow of data, information or processin the flowchart.
	Terminal	An oval shape symbol that represents the Start and End of aflowchart.
	Process	A Rectangle shape symbol is used to represent the processor action taken or shows all the calculations.
	Decision	A diamond represents a decision symbol used for comparison or a decision. It changes the flow of control and decides a particular path to be followed.
	Input/output	A parallelogram represents either input or output operation regardless of the input or output method.
	Connector	A small circle represents a connector symbol and is used to join various parts of a flow chart. Connectors are used when flow chart is very large and complex and the numbers inside them identify their links. Control is transferred from one connector to another with the same number in a program.

Q.9 Draw a flowchart to find the area of triangle when the lengths of height and base aregiven. 09507026

Ans.



Flowchart to find the area of triangle

Properties of CT

Q.10 What are the Properties of Computational Thinking? 09507027

Ans. Following are the basic properties of Computational Thinking:

- **Decomposition**
- **Abstraction**
- **Pattern Recognition**
- **Algorithm Design**

Q.11 How pattern recognition works? 09507028

Ans. Pattern recognition involves looking for similarities or patterns among and within problems. For instance, if you notice that you always forget your homework on Mondays, you might recognize a pattern and set a reminder specifically for Sundays. Pattern recognition is an essential aspect of computational thinking.

Q.12 Give an example of Abstraction. 09507029

Ans. Making a Cup of Tea - **High-level Steps:** 1. Boil water. 2. Add tea leaves or a tea bag. 3. Steep for a few minutes. 4. Pour into a cup and add milk/sugar if desired.

Q.13 Give an example of Algorithm. 09507030

Ans. To find the sum, product and average of five given numbers.

Step 1. Start
Step 2. Input numbers, a,b,c,d,e
Step 3. Set sum to a+b+c+d+e
Step 4. Set product to a×b×c×d×e
Step 5. Set average to $\frac{a+b+c+d+e}{5}$
Step 6. Output sum, product, average
Step 7. End

Q.14 What are the principles of Computational Thinking? 09507031

Ans. Computational thinking involves several key principles that guide the process of problem-solving in a structured manner.

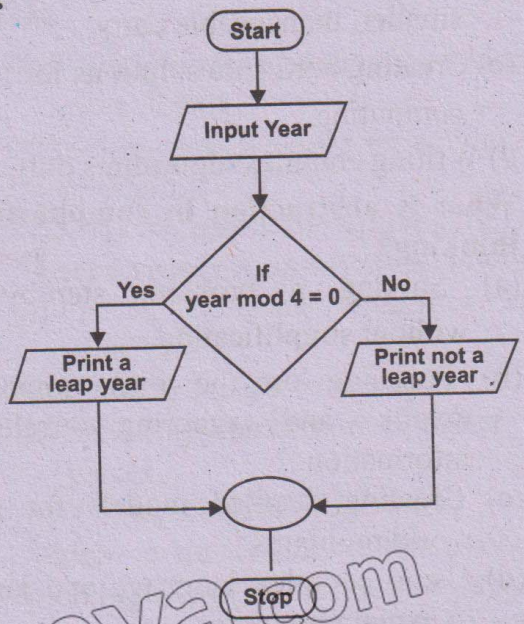
- **Problem Understanding**
- **Problem Simplification**
- **Solution Selection and Design**

Q.15 How Communication is important in Flowcharts? 09507031

Ans. They are excellent tools for communicating complex processes to a wide audience, ensuring everyone has a common understanding.

Q.16 Draw a flowchart that Input a year and determine whether it is a leap year or not. 09507032

Ans.



Pseudocodes & Dry Run

Q.17 Define Pseudocode. 09507033

Ans. Pseudocode is a method of representing an algorithm using simple and informal

language that is easy to understand. It combines the structure of programming clarity with the readability of plain English, making it a useful tool for planning and explaining algorithms.

Q.18 Give an example of Pseudocode. 09507034

Ans. Pseudocode for determining if a number is even or odd.

```
1. Procedure Check Even Odd(number)
2. Input: number (The number to be checked)
3. Output: "Even" if number is even, "Odd" if number is odd
4. Begin
5. if (number % 2 == 0) then
6. print "Even"
7. else
8. print "Odd"
9. End if
10. End
```

Q.19 Why Pseudocodes used? 09507035

Ans. Pseudocode is a method of representing an algorithm using simple and informal language that is easy to understand. It combines the structure of programming clarity with the readibili9ty of plain English, making it a useful tool for planning and explaining algorithms.

Q.20 What is Dry Run? 09507036

Ans. A dry run involves manually going through the algorithm with sample data to identify any errors.

Topic Wise Multiple Choice Questions (Additional)

Q.1 Choose the correct option

Computational & Their Properties

1. Which of the following best describes computational thinking? 09507037

(a) The process of performing mathematical calculations

(b) Breaking down a problem into smaller, manageable parts

(c) Creating hardware solutions for computing

d) Writing complex algorithms only

2. What is abstraction in computational thinking? 09507038

(a) Solving a problem step-by-step without simplification

(b) Focusing on the most important details and ignoring irrelevant information

(c) Creating detailed models for real-world problems

(d) Using complex language and jargon to explain problems

3. Which of the following is an example of pattern recognition in computational thinking? 09507039

(a) Designing a new algorithm for every problem

(b) Identifying trends in data to predict future outcomes

(c) Writing a program in multiple programming languages

(d) Solving problems without breaking them into smaller parts

4. Which step in computational thinking involves creating a sequence of steps to solve a problem? 09507040

(a) Abstraction

(b) Algorithmic design

(c) Decomposition

(d) Pattern recognition

5. What is decomposition in computational thinking? 09507041

(a) Creating a set of instructions to follow

(b) Breaking a complex problem into smaller, easier-to-solve parts

(c) Identifying patterns and similarities between problems

(d) Ignoring unnecessary details and focusing only on key aspects

6. **How many properties of computational thinking are available?**
(a) 1 (b) 2 09507042
(c) 3 (d) 4

7. **Breaking down the larger problems into smaller/ manageable ones and working on them one by one is called.**
(a) Abstraction 09507043
(b) Decomposition
(c) Algorithm design
(d) Pattern Recognition

Flowcharts

8. **Which is a graphical representation of an algorithm?** 09507044
(a) Matrix (b) Graph
(c) Flowchart (d) Solution

9. **Which symbol in the flowchart is used to either start or end the flowchart?** 09507045
(a) Terminal (b) Connector
(c) Process (d) Decision

10. **The diamond symbol represents the:** 09507047
(a) Input/output (b) Decision making
(c) Processing (d) Remarks

11. **In flow charts symbol ◇ is used to show a:** 09507048
(a) Solution (b) Decision making
(c) Verification (d) Test data

12. **Which technique has drawn a pictorial representation of the solution?** 09507049
(a) Prototype (b) Pseudo
(c) Debugging (d) Testing

13. **What is the purpose of a parallelogram shape in flowcharting?** 09507050
(a) Decision (b) Connector
(c) Input/output (d) Start or End

14. **What is the purpose of an oval shape symbol in flowcharting?** 09507051
(a) Decision (b) Connector
(c) Process (d) Start or End

Answer Key

1	b	2	b	3	b	4	b	5	b	6	d	7	b	8	c	9	a	10	b
11	b	12	a	13	c	14	d												

Solved Exercise

Choose the correct option.

1. **Which of the following best defines computational thinking?** 09507052
(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.

2. **Why is problem decomposition important in computational thinking?** 09507053
(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.

3. **Pattern recognition involves:** 09507054
(a) Finding and using similarities within problems
(b) Ignoring repetitive elements
(c) Breaking problems into smaller pieces
(d) Writing detailed algorithms

4. **Which terms refers to the process of ignoring the detail to focus on the main idea?** 09507055
(a) Decomposition
(b) Pattern recognition
(c) Abstraction
(d) Algorithm design

5. **Which of the following is a principle of computational thinking?** 09507056
   (a) Ignoring problem understanding
   (b) Problem simplification
   (c) Avoiding solution design
   (d) Implementing random solutions
6. **Algorithms are:** 09507057
   (a) Lists of data
   (b) Graphical representations
   (c) Step-by-step instructions for solving a problem
   (d) Repetitive patterns
7. **Which of the following is the first step in problem-solving according to computational thinking?** 09507058
   (a) Writing the solution
   (b) Understanding the problem
   (c) Designing a flowchart
   (d) Selecting a solution
8. **Flowcharts are used to:** 09507059
   (a) Code a program
   (b) Represent algorithms graphically
   (c) Solve mathematical equations
   (d) Identify patterns
9. **Pseudocode is:** 09507060
   (a) A type of flowchart
   (b) A high-level description of an algorithm using plain language
   (c) A programming language
   (d) A debugging tool
10. **Dry running a flowchart involves:** 09507061
   (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

Answer Key

1	b	2	a	3	a	4	c	5	b	6	c	7	b	8	b	9	b	10	b

Short Questions

Q.1 Define computational thinking. 09507062

Ans. Short Question No. 1

Q.2 What is decomposition in computational thinking?

Ans. Decomposition is an important step in computational thinking. It involves dividing a complex problem into smaller, manageable tasks. Let's take the example of building a birdhouse. This task might look tough at first, but if we break it down, we can handle each part one at a time.

Q.3 Explain pattern recognition with an example. 09507063

Ans. Short Question No. 11

Q.4 Describe abstraction and its importance in problem-solving. 09507064

Ans. Abstraction is a fundamental concept in problem solving, especially in computer science. It involves simplifying complex problems by breaking them down into smaller, more manageable parts, and focusing only on the essential details while ignoring the unnecessary ones.

Q.5 What is an algorithm? 09507065

Ans. An algorithm is a step-by-step collection of instructions to solve a problem or complete a task similar to following a recipe to bake a cake.

Q.6 How does problem understanding help in computational thinking? 09507066

Ans. Understanding a problem involves identifying the core issue, defining the requirements, and setting the objectives. Understanding the problem is the first and most important step in problem-solving, especially in computational thinking.

OR

Ans. Short Question No. 2

Q.7 What are flowcharts and how are they used? 09507067

Ans. Short Question No. 4

Q.8 Explain the purpose of pseudocode.

Ans: Short Question No. 17 09507068

Q.9 How do you differentiate between flowcharts and pseudocode? 09507069

Ans. Here's the comparison between **flowcharts** and **pseudocode** in a tabular form:

Flowcharts	Pseudocode
Graphical symbols (rectangles, diamonds, etc.)	Text-based, resembles programming language syntax
Visually represents the sequence of operations and decision-making process	Describes the algorithm in a structured, human-readable format
High-level view, easy to follow at a glance	More detailed, step-by-step instructions
Used in presentations, documentation, and for explaining logic visually	Used by developers to outline algorithms before coding
Non-programmers or visual learners	Programmers or those familiar with algorithm design
Simpler for small algorithms, harder for complex ones	Flexible for complex algorithms, but can become lengthy

Q.10 What is a dry run and why is it important? 09507070

Ans: Short Question No. 20

Q.11 Describe LARP and its significance in learning algorithms. 09507071

Ans: LARP stands for Logic of Algorithms for resolution of Problems. It is a fun and interactive way to learn how algorithms work by actually running them and seeing the results. Think of it as a playground where you can experiment with different algorithms and understand how they process data.

Q.12 List and explain two debugging techniques: 09507072

Ans: Debugging is the process of finding and fixing errors in an algorithm or flowchart. Here are some common debugging techniques:

- **Trace the Steps:** Go through each step of your algorithm or flowchart to see identity where it goes wrong.
- **Use Comments:** Write comments or notes in your algorithm to explain what each part is supposed to do. This can help you spot mistakes.

Long Questions

1. **Write an algorithm to assign a grade based on the marks obtained by a student. The grading system follows these criteria:** 09507073

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

Ans.

Start

Input: Marks obtained by the student (let's call it marks).

If marks >= 90, assign grade as **A+**.

Else if 80 <= marks < 90, assign grade as **A**.

Else if 70 <= marks < 80, assign grade as **B**.

Else if 60 <= marks < 70, assign grade as **C**.

Else, assign grade as **F**.

Output: Display the assigned grade.

End

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. 0950704

Ans. See Long Question No. 6

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.

Ans. See Long Question No. 1

4. Discuss the concept of decomposition in computational thinking. Why is it important? 0950705

Ans. Decomposition is a key notion in computational thinking that entails dividing a large problem or system into smaller, more manageable components. This technique simplifies the problem, making it easier to comprehend, evaluate, and resolve. Decomposition, by focusing on individual components, aids in the identification of patterns, streamlines the problem-solving process, and ensures clarity. It is especially significant in software development, where huge projects can be broken down into modules such as user interface design, backend development, and database management. This modular approach not only improves knowledge, but it also increases efficiency because different components can be worked on concurrently by different teams or people. Furthermore, deconstruction makes solutions to smaller sub-problems more reusable, as they can often be applied to other projects or settings. It also promotes scalability by allowing systems to evolve or adapt through changes to individual components rather than redesigning the entire system. Overall, decomposition is an effective technique for structured problem solving, modularity, and efficiency in computational thinking and beyond.

5. Explain pattern recognition in the context of computational thinking. How does identifying patterns help in problem-solving? 0950706

Ans. See Long Question No. 2

6. What is an abstraction in computational thinking? Discuss its importance and provide example of how abstraction can be used to simplify complex problems. 0950707

Ans. See Long Question No. 3

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. 0950708

Ans. See Long Question No. 4

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. 0950709

Ans. See Long Question No. 11

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? 09507080

Ans. See Long Question No. 13

10. What is LARP? Discuss its importance in learning and practicing algorithms. 09507081

Ans. See Long Question No. 15

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. 09507082

Ans. See Long Question No. 17

Activities

Activity 1

09507083

Decompose a Task

Think of a complex task you do regularly, like organizing school event or cooking a meal. Break it down into smaller, manageable parts. Write down each step and discuss with your classmates how decomposition makes the task easier to handle.

Sample: Complex Task: Cooking a Meal

1. **Plan the Menu**: Decide what dishes to prepare based on available ingredients, dietary preferences, and time constraints.
2. **Gather Ingredients**: Check the pantry, make a shopping list for missing items, and purchase them.
3. **Prepare Ingredients**: Wash, peel, chop, or marinate as required for the chosen recipes.
4. **Cook Each Dish**: Follow the recipe steps for each dish, such as sautéing, boiling, or baking.
5. **Set the Table**: Arrange plates, utensils, and any additional items like condiments.
6. **Serve the Meal**: Present the dishes in serving bowls or plates and ensure everything is ready to eat.
7. **Clean Up**: Wash dishes, wipe surfaces, and store leftovers.

Ans. Class Work\ Lab Work\ Practical Work

Activity 2

09507084

Create a table with side lengths from 1 to 10. Calculate the areas of the square using the pattern of adding consecutive odd numbers. Verify your results by squaring the side lengths and see if the pattern holds.

Ans. Sample table

Side Length	Area (Odd Numbers)	Area (Squared)
1	1	1
2	4	4
3	9	9
4	16	16
5	25	25
6	36	36
7	49	49
8	64	64

Side Length	Area (Odd Numbers)	Area (Squared)
9	81	81
10	100	100

Activity 3

09507085

Let's create an algorithm! Think of something you do every day, like brushing your teeth or packing your school bag. Write down the steps you follow, one by one. Share you algorithm with your class and see if your friends can follow it!

Ans. Class Work\ Lab Work\ Practical Work

Activity 4

09507086

Outline an algorithm for applying to the Board of Intermediate and Secondary Education (BISE) for 9th Grade Examination.

Algorithm Challenge

Work in pairs to create an algorithm for a common task, such as making a sandwich or getting ready for school. Write down each step clearly, then exchange algorithms with another pair. Follow their algorithm exactly as written and see if you can complete the task.

Ans. Class Work\ Lab Work\ Practical Work

Activity 5

09507087

Create a flowchart for a daily routine activity, such as getting ready for school. Include decision points like choosing what to wear based on the weather.

Ans. Class Work\ Lab Work\ Practical Work

Activity 6

09507088

Draw a flowchart for selecting the school cricket team. The team can have a maximum of 11 players, and each player must have parental permission.

Ans. Class Work\ Lab Work\ Practical Work

Activity 7

09507089

Create Your Own Pseudocode: Divide the students into small groups and assign each group a different simple problem, such as finding the maximum number in a list or calculating the factorial of a number. Ask them to write the pseudocode for their assigned problem and then present it to the class.

Ans. Class Work\ Lab Work\ Practical Work

Activity 8

09507090

Think of a simple task, like finding the largest number in a list. Write down the steps you would take to complete this task. Now, imagine the list has 10 numbers, then 100 numbers. How do the steps change?

Ans. Class Work\ Lab Work\ Practical Work

Activity 9

09507091

Create a simple flowchart in LARP that calculates the average of three numbers. Introduce a syntax error, a runtime error, and a logical error in your flowchart. Then try to fix them using the debugging techniques we discussed.

Ans. Class Work\ Lab Work\ Practical Work