Problem –Solving and Algorithms

Problem-solving and computational Skills

- The problems that computers can solve are called algorithmic problems.
- An Algorithmic problem is an instance of an abstract problem whose solutions can be computed using computers.
- A well-defined problem is a problem that is specified by the set of input and output specifications that satisfy the problem statement.
- An ill-defined problem is a problem where there are no clear specifications.

Problem-solving and computational skills

- The problem-solving process begins with the formulation of a problem statement.
- It includes all stages from problem definition to the implementation of the algorithm in the form of a programming code.
- Problem-solving is an art because the creativity and experience of the problem solver play an important role in solving problems.
- It is also considered a science because there are some standard patterns or ways of problem-solving.

Stages of Problem solving

- The stages are:

1. Problem understanding
2. Design a plan
3. Execution of a plan
4. Looking back and reviewing.

Computational Thinking

- Problem-solving has led to a new area of study called computational thinking.
- It provides a foundation of how a program can be solved.
- Computational thinking is needed because of the following two reasons:

1. It helps to clear and refine the thought process required to solve the problem.
2. Computational thinking helps to communicate the problem-solving process to another person in the form of design, diagrams and program codes.

The Elements of Computational skill

1. Decomposition
2. Pattern matching
3. Pattern generalisation and abstraction, and
4. Algorithms design

Problem-solving requires the skills to identify an algorithm with a suitable data structure and an efficient algorithm.

Programs= Data structures + Algorithms

Software Development Cycle

- Large-scale software development in large business organizations required many sophisticated steps.
- This is called the software development cycle that involves many stages that are below:

1. Understanding the problem
2. Problem analysis and specifications
3. Algorithm design
4. Implementation
5. Debugging and testing

Algorithm design strategies

- Algorithm design is the process of writing unambiguous step-by-step procedures for solving problems.
- Design strategies can lead to effective design and implementation. Some of the popular design strategies are listed below:

1. Stepwise Refinement
2. Bottom-up approach
3. Problem Reduction

Advantages and Disadvantages

No.	Advantages of Bottom-up Design	Disadvantages of Bottom-up Design
1.	Supports reuse and avoids recomputation.	More redundant work due to overlapping subproblems.
2	Testing is easy.	

No.	Advantages of Stepwise refinement	Disadvantages of Stepwise refinement
1.	The top-down design allows complex problems to be split up as subproblems and then visualise overall solutions in terms of subproblem solutions.	Overlapping subproblems gives redundancy.
2	It supports the usage of any functional, procedural, and object-oriented programming languages.	The testing function is complicated as it may depend on other functions yet to be developed.
3	It encourages abstraction.	

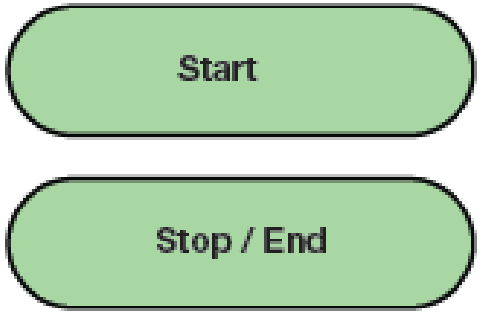
Algorithm Design

- Design strategy gives some design templates that must be communicated to the programmer.
- Communicated can be in the form of natural language, programming language, or pseudocode.
- Pseudocode is a generic way of describing or writing algorithms formally independent of any programming language or environment.
- Algorithms are built using three basic building blocks:

1. Sequence
2. Selection
3. repetition

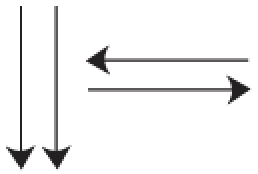
Flow charts

- A flowchart is a pictorial form representing the algorithm.
- The graphical layout of an algorithm is useful for better understanding as it helps show the algorithm’s logic.
- The flowchart illustrates the control flow from one task to another task.
- The start is the symbol used to indicate the start of the flowchart.
- It is a flattened ellipse symbol.
- The exit symbol has no exit flowlines.



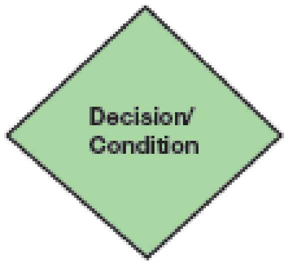
Flow charts

- Flowlines are important indicators that indicates the flow of the control.



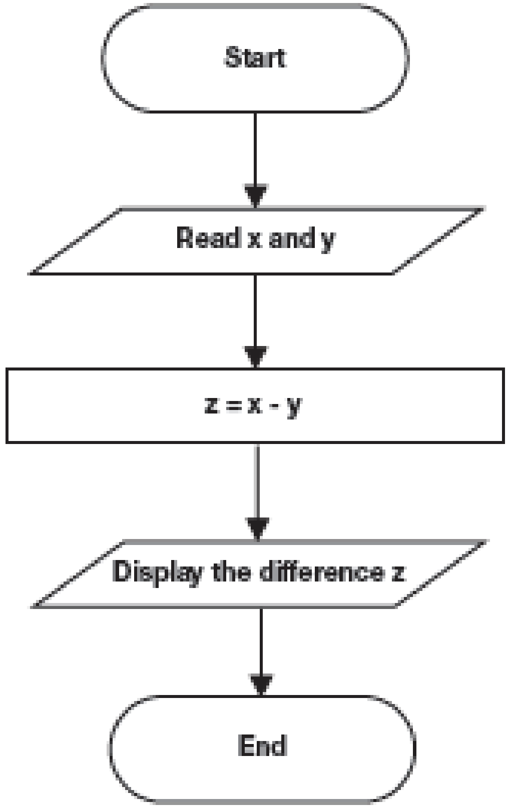
- The process symbol has one entrance and one exit. The process can be calculations or file operations.

- The symbol diamond indicates the decision symbol. It has one input and two output symbols.



Algorithm for Subtraction of Two Numbers.

Algorithm	Pseudocode
1. Start	Begin
2. Get two numbers as input and store them into variables x and y	x = input() y = input()
3. Subtract the values of numbers in the variables x & y and store the result in the variable z	z = x - y
4. Print z	print(z)
5. End	End



Analysis of Algorithms

- Algorithmic Complexity theory is a branch of algorithm study that analysis algorithms.
- Algorithm analysis tries to measure the computer resources such as time and space. There are respectively called time and space complexity.
- Algorithm Analysis

Thee are two types of analysis:

1. Mathematical analysis of algorithms
2. Empirical program analysis

Mathematical Framework for Algorithm Analysis

- The framework for complexity analysis is given below:

1. Measure the input size. i.e., the input data length.
2. Measure the running time using either a step count or a count basic operations.

Elementary, or basic operations are primitive and often implemented directly with less effort. Some of the basic operations are listed below.

1. Assignment operations
2. Comparison Operations
3. Arithmetic operations
4. Logical operations

Illustration of Complexity Analysis of an algorithm

Apply complexity analysis for the following algorithm segment.

```
Algorithm add(x,y)
z = x + y
m = x – y
k = x * y
r = x / y
return z,m,k,r
```

Step No	Algorithm segment	Frequency of execution	Cost
1.	Algorithm add(x,y)	-	(As this is a non-executable statement)
2.	z = x + y	1	c_1
3.	m = x – y	1	c_2
4.	k = x * y	1	c_3
5.	r = x / y	1	c_4
6.	return z,m,k,r	1	c_5

So, the total run time cost $f(n) = f(n) = c_1 + c_2 + c_3 + c_4 + c_5 \approx c$ (As the sum of all constants yields another constant). Therefore, f(n) = c. This is a constant time algorithm.

Efficiency of the Algorithm

- The final step is to identify the rate of growth.
- This step determined the algorithm’s performance when the input size is scaled higher.
- Scaling means changing the input size to a larger value and carrying out the measurement of the behaviour of the algorithms.
- The algorithm behaviour can be determined only after observing the algorithm’s performance when the input data are scaled high.
- The analysis is useful to classify the problems into sets of problems that share common characteristics; they are polynomial algorithms and exponential algorithms.

Complexity of algorithms

- The linear algorithms are listed below:

No.	Complexity	Name
1.	$O(1)$	Constant time
2.	$O(\log n)$	Logarithmic
3.	$O(n)$	Linear time
4.	$O(n^2)$	Quadratic time
5.	$O(n^3)$	Cubic time
6.	$O(2^n)$	Exponential time
7.	$O(n!)$	Exponential time

- The other type of algorithm is called exponential time. These algorithms are whose operations are bounded by constants raised to the power of a constant.