The Problem Solving Process

A computer program is a finite set of precise instructions written in a programming language. The design of a computer program involves:

- The Problem-Solving Phase
- The Implementation Phase

The Problem Solving Phase

1.Define the problem
2.Analyse the problem
3.Find a solution to the problem
4.Evaluate alternate solutions
5.Represent the most efficient solution as an algorithm
6.Test the correctness of algorithm

1. Define the problem

In defining the problem there must be no ambiguity. The objectives of the problem must be specified: output, input, processing and in some cases the storage.

2. Analyse the problem

In this stage, you need to identify the inputs to be used, outputs required, processing that needs to be done and values to be stored (if any). Input is the data needed to solve the problem, the processing instructions manipulate the input data (e.g. mathematical operations, selecting instructions) and the output instructions allow information to be displayed to the screen.

The Defining Diagram/Input-Processing-Output (IPO) Chart

This is a table with columns representing the input, output and processing activities required to solve the problem.

Example: Write the steps to read a number, find its square and print the square of a number.

1. Identify Input: (number)
   [keywords: read, input, accept, given]

2. Identify Process: (read number, calculate the square, print the square)
   [What must I do with the inputs to produce the desired output?]

3. Identify Output: (print the square of the number)
   [key words: print, display, output, produce]

IPO

Input	**Processing**	**Output**
Number eg num	Read num sq ← num * num Print sq	sq

3. Find a solution to the problem

Example: Read a number, find its square and print the square of the number.

Initial Solution

Get the number, store the number in a variable called num

Square the number by multiply the number by itself

Place the result of the square into a variable called sq

Print sq

Important Terms:

Variable: The name given to a temporary storage location for data used in a program.

- When a new value is placed into a variable, the previous value is replaced by the new one.
- Variable names should reflect the kind of data being stored.
- E.g. Sum = x+y and not Square = x+y.
- Variable names should begin with an alphabetic character.
- Initialization of variables: Variables that are used as counters or used to store totals should always be assigned an initial value of 0 before they are incremented.

Constant: Represents a memory location where a fixed item of data are stored. Eg VAT = 17.5%

4. Evaluate alternate solutions

Points to consider:

1. Can you derive the result differently?
2. Can you make the solution more general?
3. Can you use the solution to solve another problem?
4. Can you reduce the number of steps and still maintain the logic?
5. Can you make the solution more robust? Would it work properly if incorrect data were entered?

5. Represent the most efficient solution as an algorithm

The most efficient solution does not only mean the solution which results in the best use of memory, or the shortest solution. The most efficient solution should be:

- maintainable
- memory efficient
- robust

Algorithm: A sequence of instructions which represent the solution to a problem in a finite number of steps.

NB. Algorithms may be expressed as narrative, pseudocode or flowcharts .

Features of algorithms: They must be:

- Correct: Accept all inputs (even invalid inputs) and output a correct answer or meaningful message.
- Clear: The algorithm should be easy to read and understand.
- Easy to implement: Translation to a programming language should be easy.

- Efficient: The algorithm should facilitate a program which executes quickly.
- Finite: There must be an end point for the algorithm.

6. Test the correctness of algorithm

Trace Tables:

A table in which you write the values of the variables in your algorithm. Test data are used to check each statement of the algorithm to see if the algorithm is producing the correct results.

Dry Run

num	sq	Output
5	25	25

Important Terms:

- **Dry Run**: A manual traversal of the logic and correctness of a program.
- **Testing**: The process of checking the logic and correctness of a program.
- **Test Data**: Dummy data used to check the logic and correctness of a program before end-users operate it.
- **Live Data**: Data previously processed by the system, that is used to check the logic and correctness of a program before end-users operate it.
- **Program Trace**: A software traversal of the logic and correctness of a program.