

Boys' High School and College



Estd. 1861

Prayagraj

Session 2020-2021

Round 7

Subject : Computer Science

Class : XI

Worksheet 4





**Topic : Implementation of Algorithm for
problem solving**

Introduction to Algorithm

Before writing a computer program that solves a problem you need two things

- ❖ A good understanding of the problem.
- ❖ A plan how to solve it.

What is an Algorithm ?

-  Any computing problem can be solved by executing a sequence of instructions in a specific order
-  An Algorithm is such a sequence of instructions that leads to a predictable result
-  Algorithm is developed before actual coding is done.
-  It is written using English language so that it is understandable by non programmers

An algorithm is a procedure for solving a problem in terms of the actions to be executed and the order in which those actions are to be executed. An algorithm is merely the sequence of steps taken to solve a problem. The steps are normally "sequence," "selection," "iteration," and a case-type statement.



Example of an everyday algorithm

Get ready for school

Step 1: Get up

Step 2: Take a shower

Step 3: Get dressed

Step 4: Eat breakfast

Step 5: Go to the school

Tips for writing a good algorithm

- ❖ A good algorithm should produce the correct output for any set of legal inputs.
- ❖ A good algorithm should execute efficiently with the fewest number of steps as possible.
- ❖ A good algorithm should be designed in such a way that others will be able to understand it and modify it to specify solutions to additional problems

What is Pseudo code?

Pseudo code is an artificial and informal language that helps programmers develop algorithms.

Pseudo code is a "text-based" detail (algorithmic) design tool. The rules of Pseudo code are reasonably straightforward.

How to write a Pseudo code?

- 1) Arrange the sequence of tasks and write accordingly.
- 2) Start with the statement of Pseudo code which establishes the main goal or the aim.
- 3) The way if-else, for, while loops are indented in a program, indent the statements likewise, as it helps to comprehend the decision control and execution mechanism. They also improve readability to a great extent.

Example

```
if "1"  
    print response  
    "I am case 1"
```

```
if "2"  
    print response  
    "I am case 2"
```

- 4) Use appropriate naming conventions. The human tendency follows what we see. If a programmer goes through a Pseudo code, his approach will be the same as per it, so the naming must be simple and distinct.
- 5) Use appropriate sentence casing as camelCase for methods , upper case for constants and lower case for variables.
- 6) Elaborate everything which is going to happen in the actual code. Don't make the pseudo code abstract.
- 7) Use standard programming structures such as 'if then' , 'for' , 'while' , 'cases' the way we use it in programming .
- 8) Check whether all the sections of a pseudo code is complete, finite and clear to understand and comprehend.
- 9) Don't write the pseudo code in a complete programmatic manner. It is necessary to be simple to understand even for a layman or

client , hence don't incorporate too many technical terms.

Some Keywords That Should be Used And Additional Points

- For looping and selection, The keywords that are to be used include Do While; Do Until...Enddo; While Endwhile is acceptable. Also, Loop endloop is also VERY good and is language independent. Case...EndCase; If...Endif; Call ... with (parameters); Call; Return;
- Always use scope terminators for loops and iteration.
- As verbs, use the words Generate, Compute, Process, etc. Words such as set, reset, increment, compute, calculate, add, sum,

multiply, ... print, display, input, output, edit, test, etc. with careful indentation tend to foster desirable pseudo code. Also, using words such as Set and Initialize, when assigning values to variables is also desirable.

- Do not include data declarations in your pseudo code.
- But do cite variables that are initialized as part of the declarations. E.g. "initialize count to zero" is a good entry.



Example 1 - Two add two numbers

Step 1: Start the algorithm.

Step 2: Declare variables num1, num2 and sum

Step 3 :Read or input values for num1 and num2

Step 4: Add num1 and num2 and assign the result to sum

Step 5: Print the output

Step 6: End the algorithm.



Example 2 - Magic Number

Algorithm for finding out a number is Magic or not.

A number is said to be Magic if the eventual sum of digits is 1. Eg 82,19,100.

Step 1: Start the algorithm.

Step 2: Initialize sumOfDigits variable value to zero.
It represents the sum of digits of a given input number

Step 3: Create a copy of the inputNumber (original number) by storing its value in the variable number .

Step 4: Using while loop

a. Continue the loop till the sumOfDigits does not become a single digit or number is not equal to zero

b. Get the right most digit of variable number by using the $(\text{number} \% 10)$ and add its value to the variable sumOfDigits

Step 5: Check whether variable sumOfDigits is equal to 1

- a. If both are equal i.e 1 is equal to sumOfDigits then the inputNumber is Magic
- b. Otherwise, the inputNumber is not Magic number

Step 6: End the algorithm