

DATA STRUCTURES & ALGORITHMS

#01

Getting familiar with data structures and algorithms

|  |
| --- |
| Student Name: |
| Roll Number: Section: |
| Work submitted on: |

|  |  |  |  |
| --- | --- | --- | --- |
| **Maximum Marks** | **Performance** | **Viva** | **Total** |
| **Marks Obtained** |  |  |  |
| **Remarks (if any)** |  | | |
|  | | | |
| **Experiment evaluated by** | | | |
| Instructor Name: | | | |
| Signature: | | | |

|  |
| --- |
| **Theory** |

**Data Structures**

Data may be organized in many different ways; the logical or mathematical representation of a particular organization of data is called a *data structure*. The choice of a particular data model depends on two considerations. First, it must be rich enough in structure to mirror the actual relationships of data in the real world. On the other hand, the structure should be simple enough that one can effectively process the data when necessary. Some of the widely used data structures are:

* Array
* Structure
* Linked List
* Stack
* Queue
* Tree
* Graph

**Data Structure Operations**

The data appearing in data structures are processed by means of certain operations. Some of the most frequently used operations are:

* Traversing
* Searching
* Inserting
* Deleting
* Sorting
* Merging

**Algorithm**

An algorithm is a well-defined list of steps for solving a particular problem. The problem can be a calculation, data processing, automated reasoning task etc. Algorithms tell the programmers how to code the program. A good algorithm suggests an efficient way of performing one or more of the operations mentioned above. Algorithm writing is a process and is executed after the problem domain is well-defined. That is, we should know the problem domain, for which we are designing a solution.

|  |
| --- |
| **Examples** |

**Example 1:**

**Algorithm 1**: This algorithm receives two integers A and B from user, adds them

and writes the sum in C.

**Step 1.** Read: A, B

**Step 2.** Set C=A+B

**Step 3**. Write: C

**Step 4.** Exit

**Code:**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

int a, b, c;

Scanner input = new Scanner(System.in);

System.out.print("Enter 1st Number: ");

a = input.nextInt();

System.out.print("Enter 2nd Number: ");

b = input.nextInt();

c = a + b;

System.out.println("Sum is " + c);

input.close();

}

}**Example 2:**

**Algorithm 2**: This algorithm receives an integer N from user and calculates its

multiplication table up to 10. The variable K is used as a counter.

**Step 1.** [Initialize] Set K:=1

**Step 2.** Read: N

**Step 3**. Repeat Step 4 and 5 while K<=10:

**Step 4.** Write: N, K, N\*K

**Step 5.** K=K+1

[End of Step 3 loop]

**Step 6.** Exit

**Code:**

import java.util.Scanner;

public class Main {

public static void main(String[] args) {

int n, k;

Scanner input = new Scanner(System.in);

System.out.print("Enter Number: ");

n = input.nextInt();

k = 1;

while (k <= 10) {

System.out.println(n + "x" + k + "=" + (n \* k));

k = k + 1;

}

input.close();

}

}

|  |
| --- |
| **Do It Yourself** |

Task 1

Write algorithm and Java program in which create a class with name MyMath and create a method is\_multiple(n,m), that takes two integer values and returns T rue if n is a multiple of m, that is, n = m × i for some integer i, and F alse otherwise.

Function Calling

is\_multiply (20 ,3)

Output

False

Task 2

Write algorithm and Java program in which create a class with name demo and create a method, is\_even(k), that takes an integer value and returns T rue if k is even, and F alse otherwise. However, your function cannot use the multiplication, modulus, or division operators.

Task 3

Write algorithm and Java program that takes n integers input and returns the sum of the squares of all the positive integers smaller than n.

Task 4

Write an algorithm and a java program to calculate factorial of a given number.

Task 5

Write algorithm and Java program to find out whether the given String is Palindrome or not.

|  |
| --- |
| **Viva** |

Answer True or False for these statements:

1. Array is a data structure operation. \_\_\_\_\_\_ (True/False)
2. Stacks, queue and trees are types of data structures. \_\_\_\_\_\_ (True/False)
3. An algorithm written for C++ will be completely different than the one for any other language. \_\_\_\_\_\_ (True/False)
4. Traversing means visiting each element of a data structure in order to perform a certain operation. \_\_\_\_\_\_ (True/False)
5. Choice of a data structure for a particular scenario depends on the nature of data. \_\_\_\_\_\_ (True/False)