

DATA STRUCTURES & ALGORITHMS

#01

Getting familiar with data structures and algorithms

|  |
| --- |
| Student Name: SAAD MAQSOOD |
| Roll Number: 22SP-017-SE Section: A |
| 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)

**Algorithm:**

Step 1: Start Program.

Step 2: Input a number: n

Step 3: Input a second to check whether it is a multiple of first number: m

Step 4: Check if n%m == 0

Step 5: is\_multiple: return true;

Step 6: else return false.

Step 7: End program.

**Output**

False

import java.util.Scanner;

public class MyMath

{

    public static void main(String[] args)

    {

        Scanner obj = new Scanner(System.in);

        System.out.print("Enter the number: ");

        int n = obj.nextInt();

        System.out.printf("Enter the number to check if it is factor of %d: ",n);

        int m = obj.nextInt();

        System.out.println();

        System.out.print(is\_multiple(n,m));

    }

    static boolean is\_multiple(int a, int b)

        {

            boolean check = false;

            if(a%b==0)

            {

                check = true;

            }

            return check;

        }

}

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.

**Algorithm**

Step 1: Start Program.

Step 2: Input a number num;

Step 3: Run from 1 up to num for checking even number.

Step 4: Set variable ev\_no = true.

Step 5: For odd value variable ev\_no will be (negation) !ev\_no and becomes false.

Step 6: For even value ev\_no will again become true.

Step 7: Return ev\_no.

Step 8: End program.

import java.util.Scanner;

public class Demo1 {

    static boolean is\_even(int k)

    {

        boolean ev\_no=true;

        for(int i=1; i<=k; ++i){

            ev\_no = !ev\_no;

        }

        return ev\_no;

    }

    public static void main(String args[])

    {

        Scanner obj = new Scanner(System.in);

        System.out.print("Enter the number to find either it is even or odd: ");

        int num = obj.nextInt();

        System.out.print("Number is even: ");

        System.out.print(is\_even(num));

    }

}

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.

**Algorithm**

Step 1: Start Program.

Step 2: Give number of inputs to variable no\_in;

Step 3: Now take input no\_in times.

Step 4: As the input is taken, perform n1 \*= n1; for squaring the number.

Step 5: Store the square to variable sqr\_sum as sqr\_sum += n1.

Step 6: As Inputs are completed Output the added squares stored in sqr\_sum.

Step 7: End program.

import java.util.Scanner;

public class task3 {

    public static void main(String args[])

    {

        Scanner obj = new Scanner(System.in);

        System.out.println("How many inputs you want to give: ");

        int no\_in = obj.nextInt();

        int sqr\_sum=0,input\_no=no\_in; // input\_no is storing it for comparison later

        while(input\_no > 0)

        {

            System.out.println("Enter the number: ");

            int n1 = obj.nextInt();

            if(n1 < no\_in)

            {

                n1 \*= n1;

                sqr\_sum += n1;

            }

            input\_no--;

        }

        System.out.printf("Sum of squares of numbers less than %d are: %d",no\_in,sqr\_sum);

        obj.close();

    }

}

Task 4

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

**Algorithm**

Step 1: Start Program.

Step 2: Give the number a to find factorial up to that number

Step 3: Initialize the variable r by 1 and i by 2;

Step 4: Multiply i by r and store it in r as r \*= i;

Step 5: Repeat Step 3 until i<=a(input);

Step 6: Return r and output it.

Step 7: End program.

import java.util.Scanner;

public class factorial {

    static int factorial(int a)

    {

        int r=1;

        for(int i=2; i<=a; ++i)

        {

            r \*= i;

        }

        return  r;

    }

    public static void main(String args[])

    {

        Scanner obj = new Scanner(System.in);

        System.out.print("Input number to find factorial: ");

        int num = obj.nextInt();

        //int result = f

        System.out.printf("The factorial of %d is: %d",num,factorial(num));

    }

}

Task 5

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

**Algorithm**

Step 1: Start Program.

Step 2: Give a String to check for Palindrome.

Step 3: Get a single character using charAt(i).

Step 4: Start getting characters from last index using length() method

Step 5: Store the characters to empty string s2.

Step 6: Repeat the above steps till first index.

Step 7: Make the both strings to lower case before comparison.

Step 8: Now compare the s2 string with original string using equals method.

Step 9: If strings are equal return true, otherwise return false.

Step 10: End program.

import java.util.Scanner;

public class palindrome {

    static boolean is\_palindrome(String a)

    {

        boolean check = false;

        String s2 = "";

        for(int i=a.length()-1; i>=0; --i){

            s2 = s2 + a.charAt(i);

        }

        if(s2.toLowerCase().equals(a.toLowerCase())){

            check = !check;

        }

        return check;

    }

    public static void main(String args[])

    {

        Scanner obj = new Scanner(System.in);

        System.out.println("Enter the string to check for palindrome.");

        String s1 = obj.nextLine();

        System.out.printf("String %s is palindrome: %b",s1,is\_palindrome(s1));

    }

}

|  |
| --- |
| **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)