**C-DAC Mumbai Date 25/09/2024**

**Subject: Algorithm and Data Structure**

**Assignment 1**

**Solve the assignment with following thing to be added in each question.**

-Program

-Flow chart

-Explanation

-Output

-Time and Space complexity

1. Armstrong Number

Problem: Write a Java program to check if a given number is an Armstrong number.

Test Cases:

Input: 153

Output: true

Input: 123

Output: false

import java.util.\*;

class Q1 {  //armstrong number

public static void main (String args[]) {

    Scanner sc = new Scanner (System.in);

    int a,b, d, sum = 0;

    System.out.println("Enter a number");

    b = sc.nextInt();

    a = b;

    while (b > 0)

    {

        d = b % 10;

        sum = sum+(d\*d\*d);

        b = b / 10;

    }

    if (a  == sum)

        System.out.println(true);

    else

        System.out.println(false);

    }

}

2. Prime Number

Problem: Write a Java program to check if a given number is prime.

Test Cases:

Input: 29

Output: true

Input: 15

Output: false

import java.util.\*;

class Q2 //prime

{

    public static void main (String args[] ) {

    System.out.println("Take any number of your choice to check prime: ");

    Scanner sc = new Scanner (System.in);

    int num = sc.nextInt();

    boolean flag = false; //set default value of boolean as false

    for (int i=2; i <=num/2; i++) {

    if (num % i == 0) {

    flag = true;

    break;

    }

    }

    if (!flag)

    System.out.println(true);

    else

    System.out.println(false);

    }

}

3. Factorial

Problem: Write a Java program to compute the factorial of a given number.

Test Cases:

Input: 5

Output: 120

Input: 0

Output: 1

class Factorial{

static int fact (int n)

{

if (n <=1)

return 1;

else

return n\* fact(n-1);

}

public static void main(String [] args)

{

System.out.println(fact(5));

}

}

4. Fibonacci Series

Problem: Write a Java program to print the first n numbers in the Fibonacci series.

Test Cases:

Input: n = 5

Output: [0, 1, 1, 2, 3]

Input: n = 8

Output: [0, 1, 1, 2, 3, 5, 8, 13]

//Finite loop

class Recursion5{

static int fib(int n){

if (n <= 1)

{

return n;

}

return fib(n-1)+fib(n-2);

}

public static void main(String args[])

{

int num=100;

for(int i=0;i<=num;i++)

{

System.out.print(fib(i)+" ");

}

}

}

5. Find GCD

Problem: Write a Java program to find the Greatest Common Divisor (GCD) of two numbers.

Test Cases:

Input: a = 54, b = 24

Output: 6

Input: a = 17, b = 13

Output: 1

public class Q5 //Calculate GCD

{

    public static int euclideanGCD(int a, int b) {

        while (b != 0) {

            int temp = a;

            a = b;              //swapping but b % b

            b = temp % b;

        }

        return a;

    }

    public static void main(String[] args) {

        int num1 = 54;      // input constraints

        int num2 = 24;

        int gcd = euclideanGCD(num1, num2);

        System.out.println("GCD of " + num1 + " and " + num2 + " is: " + gcd);

    }

}

6. Find Square Root

Problem: Write a Java program to find the square root of a given number (using integer approximation).

Test Cases:

Input: x = 16

Output: 4

Input: x = 27

Output: 5

import java.util.\*;

class Q6  {

//Square root

    public static void main (String args[]) {

    Scanner sc = new Scanner (System.in);

    System.out.println("Enter the number whose square root you want: ");

    double x = sc.nextDouble();

    double ans = (int) Math.sqrt(x);    //narrowing conversion - data type

    System.out.println(ans);

    }

}

7. Find Repeated Characters in a String

Problem: Write a Java program to find all repeated characters in a string.

Test Cases:

Input: "programming"

Output: ['r', 'g', 'm']

Input: "hello"

Output: ['l']

import java.util.Scanner;

public class Q7 {   //show repeat characters (only lowercase)

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a string: ");

        String input = scanner.nextLine();

        int[] charCount = new int[50]; // Assuming only lowercase letters

        for (int i = 0; i < input.length(); i++) {

            charCount[input.charAt(i) - 'a']++;

        }

        System.out.print("Repeated characters: ");

        for (int i = 0; i < charCount.length; i++) {

            if (charCount[i] > 1) {

                System.out.print((char) ('a' + i) + ", ");

            }

        }

    }

}

8. First Non-Repeated Character

Problem: Write a Java program to find the first non-repeated character in a string.

Test Cases:

Input: "stress"

Output: 't'

Input: "aabbcc"

Output: null

import java.util.Scanner;

import Java.util.\*;

class NonRepeatingChar

{

    public static void main(String args[])

    {

        Scanner s=new Scanner(System.in);

        System.out.println("Input a String:");

        String word = s.nextLine();

        boolean flag = true;

        for(char i :word.toCharArray())

        {

            if (word.indexOf(i) == word.lastIndexOf(i))

            {

                System.out.println("The first character which is not repeating is: "+ i);

                flag = false;

                break;

            }

        }

        if(flag== true){

            System.out.println("There is no non-repeating character in the input string");

}

     }

}

9. Integer Palindrome

Problem: Write a Java program to check if a given integer is a palindrome.

Test Cases:

Input: 121

Output: true

Input: -121

Output: false

import java.util.Scanner;

public class Q9 {   //palindrome

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a word: ");

        String input = scanner.nextLine();

        boolean isPalindrome = true;

        int left = 0;

        int right = input.length() - 1;

        while (left < right) {

            if (input.charAt(left) != input.charAt(right)) {

                isPalindrome = false;

                break;

            }

            left++;

            right--;

        }

        if (isPalindrome) {

            System.out.println(input + " is a palindrome.");

        } else {

            System.out.println(input + " is not a palindrome.");

        }

    }

}

10. Leap Year

Problem: Write a Java program to check if a given year is a leap year.

Test Cases:

Input: 2020

Output: true

Input: 1900

Output: false

import java.util.\*;

class Q10 {

    public static void main(String args[]) {

        Scanner sc = new Scanner(System.in);

        System.out.println("Enter the year you want to choose: ");

        int year = sc.nextInt();

        if (year % 4 == 0 && year %100 != 0 || year % 400 == 0)

        {

            System.out.println(true);

        }

        else

        {

                System.out.println(false);

        }

    }

}