**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

**package** org.example;

**import** java.util.\*;

**public** **class** ArmStrongnum {

**public** **static** **void** main(String[] args) {

Scanner sc=**new** Scanner(System.***in***);

System.***out***.println("Enter a Number");

**int** n1=sc.nextInt();

**int** result=0,rem;

**int** num2=n1;

**while** (n1>0) {

rem=n1%10;

result=result+(rem\*rem\*rem);

n1=n1/10;

}

**if** (num2==result)

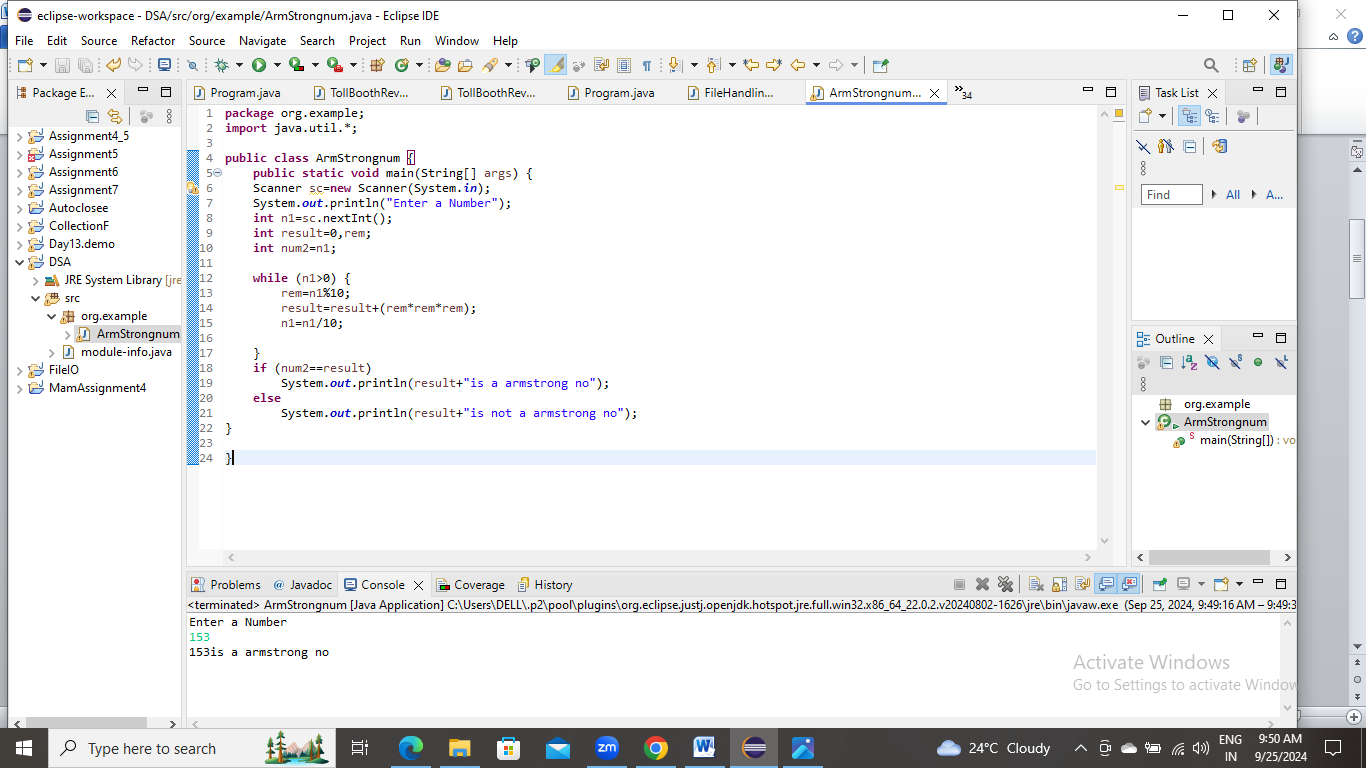
System.***out***.println(result+"is a armstrong no");

**else**

System.***out***.println(result+"is not a armstrong no");

}

}



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

class PrimeRecursion {

static int i=2;

public static void main(String[] args) {

if (isPrime(20))

System.out.println("number is prime");

else

System.out.println("number is not prime");

}

public static boolean isPrime(int n){

if (i==n/2+1)

return true;

if(n%i==0)

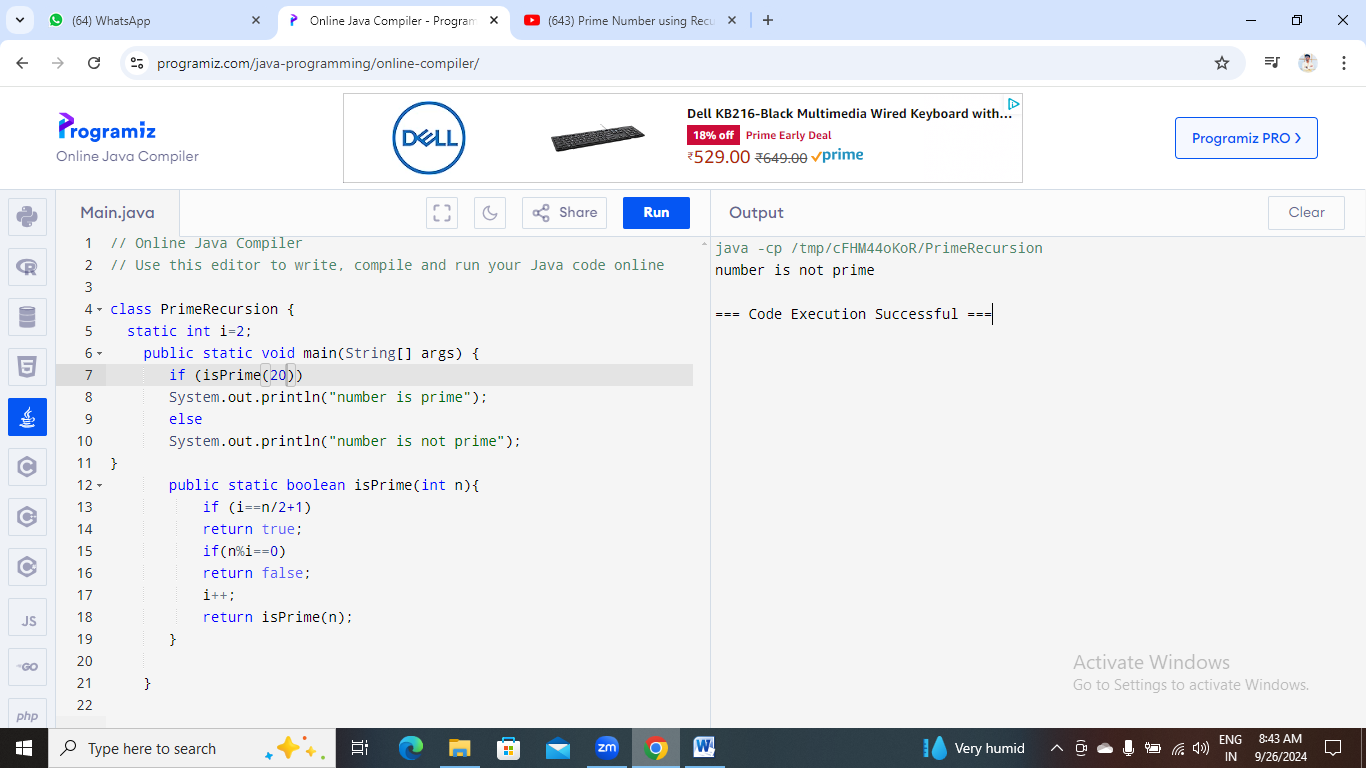
return false;

i++;

return isPrime(n);

}

}



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 Factorial1{

static int fact(int n){

if(n<=1)

return n;

else

return n\*fact(n-1);

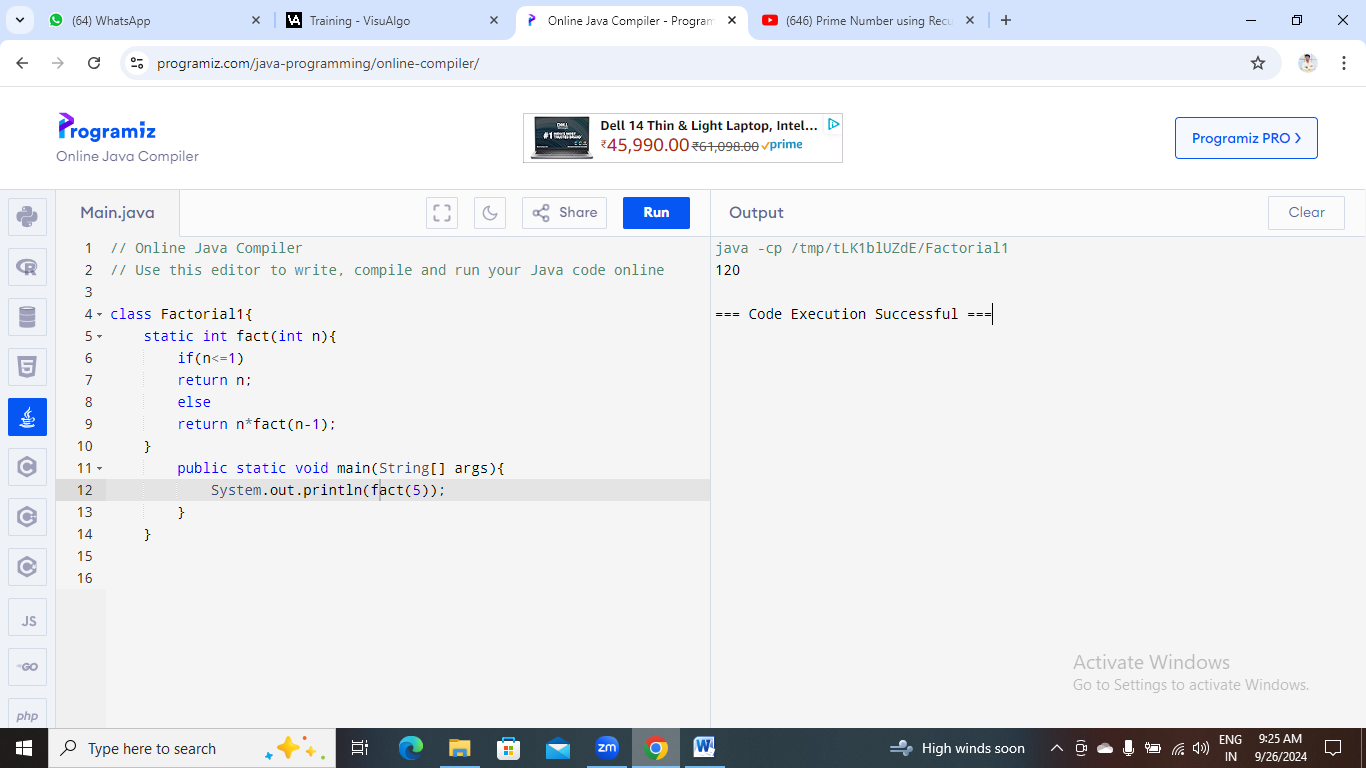
}

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]

class Fibonacci1{

public 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=10;

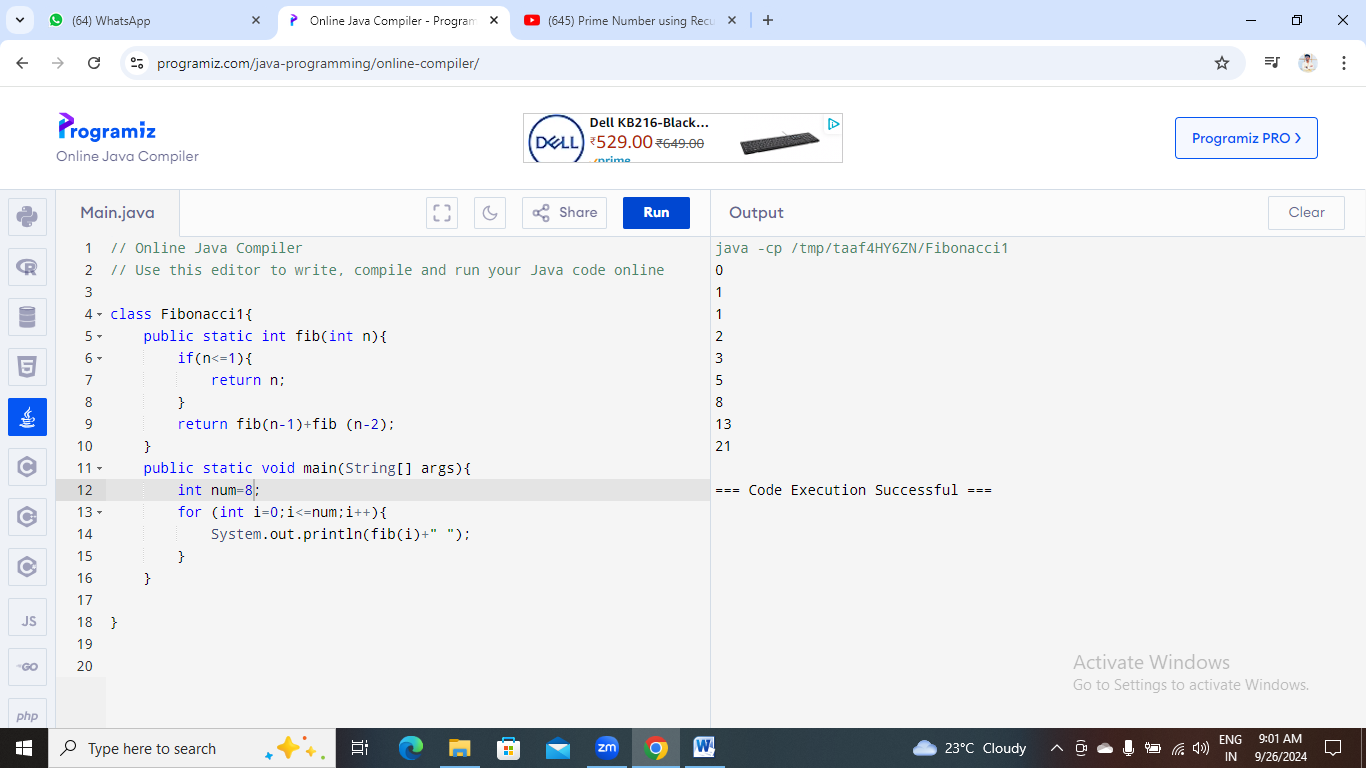
for (int i=0;i<=num;i++){

System.out.println(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 GCD {

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

if (b == 0) {

return a;

}

return findGCD(b, a % b);

}

public static void main(String[] args) {

int num1 = 56;

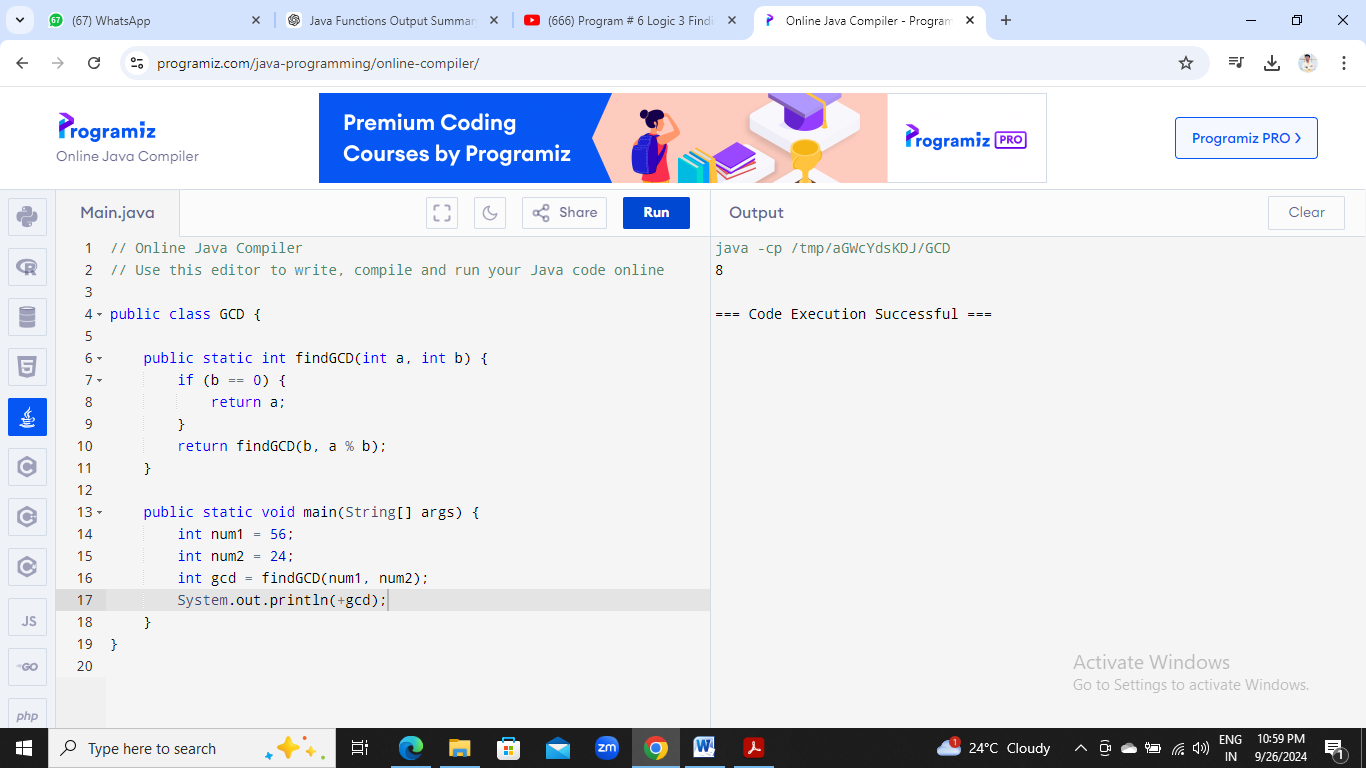
int num2 = 24;

int gcd = findGCD(num1, num2);

System.out.println(+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

class SqrtOfNum{

public static void main(String[] args){

int num=16;

double sqrt;

System.out.println(squareroot(num));

}

public static double squareroot(int num){

double temp;

double sqrt=num/2;

do{

temp=sqrt;

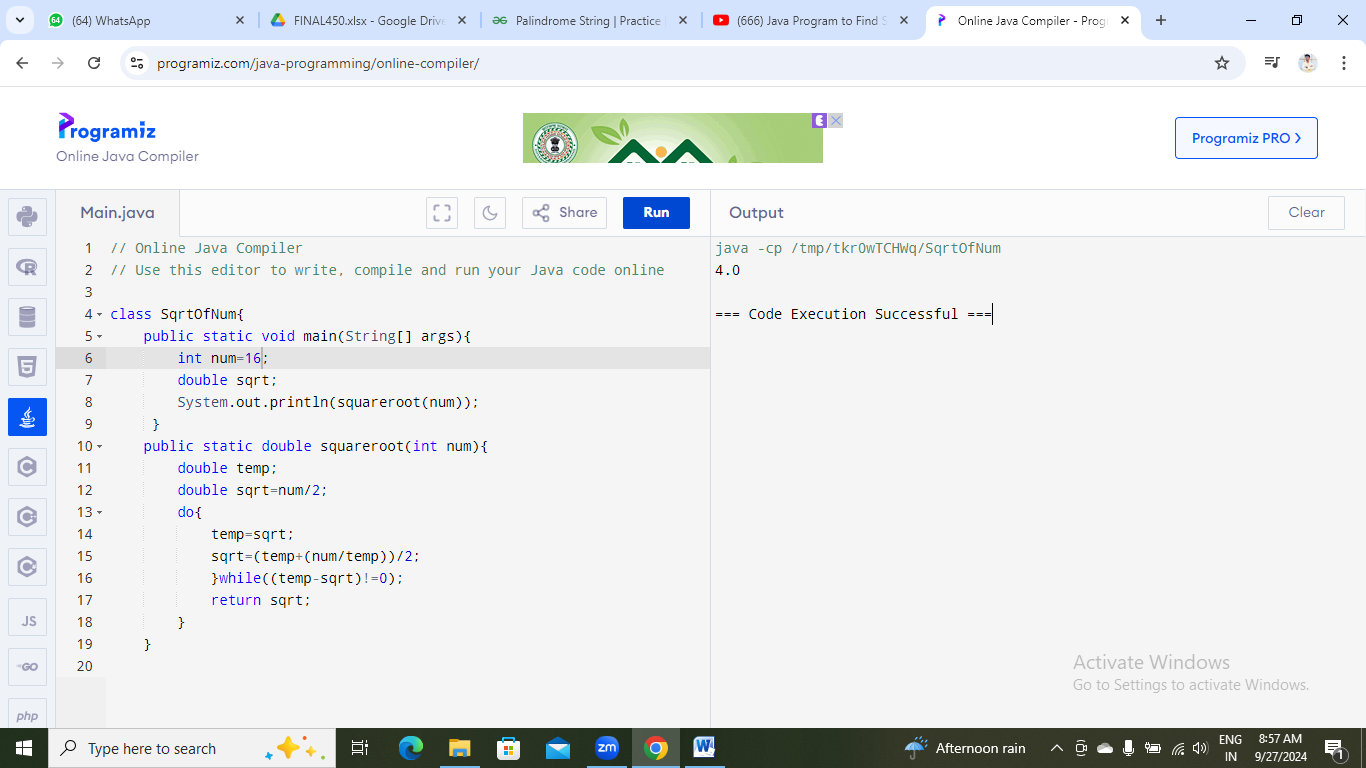
sqrt=(temp+(num/temp))/2;

}while((temp-sqrt)!=0);

return sqrt;

}

}



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']

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

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;

class PalindromeNumber{

public static void main(String[] args){

Scanner s=new Scanner(System.in);

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

int no=s.nextInt();

int temp=no;

int rev=0, rem;

while(temp != 0){

rem=temp%10;

rev= rev\*10+rem;

temp=temp/10;

}

if(no==rev)

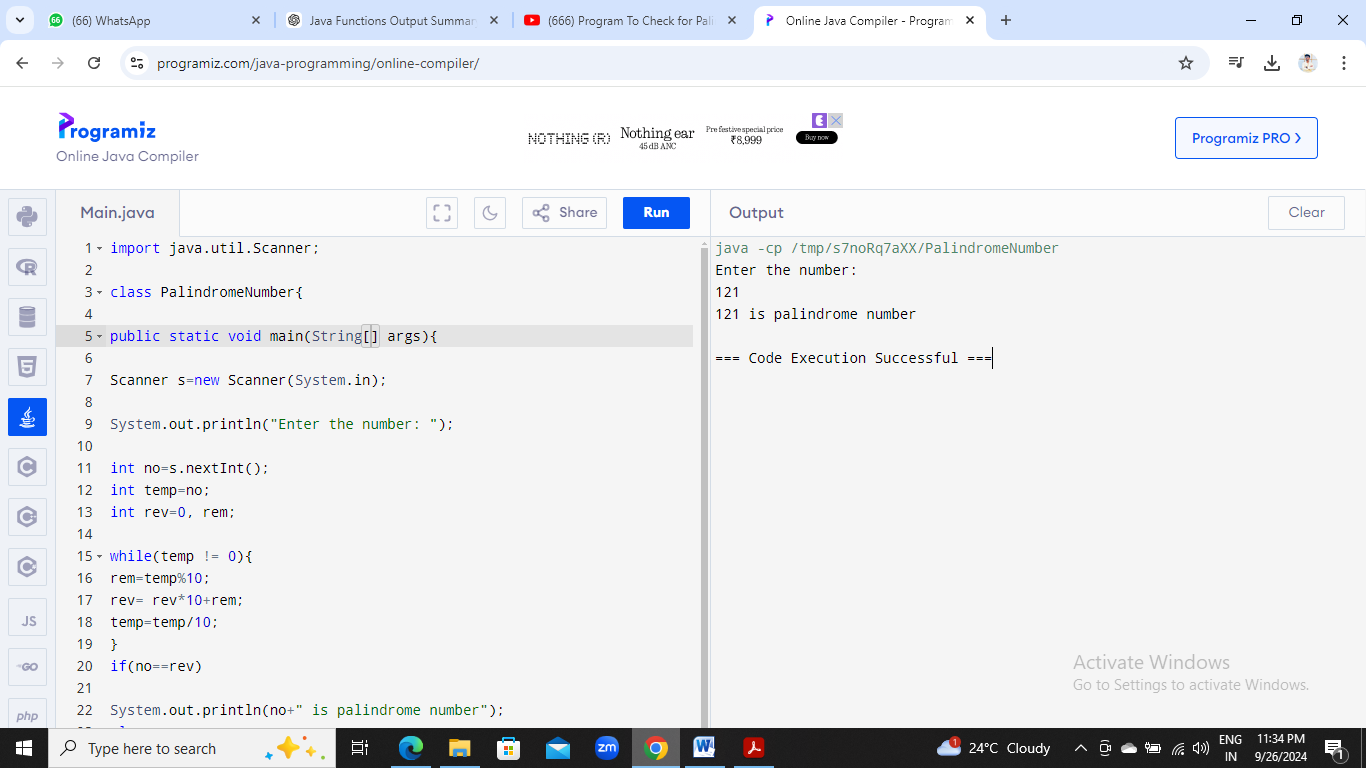
System.out.println(no+" is palindrome number");

else

System.out.println(no+" is not a palindrome number");

}

}



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

class LeapYear1{

public static void main(String args[]){

int year=2020;

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

{

System.out.println("true");

}

else{

System.out.println("false");

}

}

}

