**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

**Program :**

**package org\_example\_1;**

**import java.util.Scanner;**

**public class Program {**

**public static boolean isArmstrong(int num) {**

**int res = 0, d = 0;**

**int orgNum = num;**

**while (num != 0) {**

**num /= 10;**

**d++;**

**}**

**num = orgNum;**

**while (num != 0) {**

**int rem = num % 10;**

**res += Math.*pow*(rem, d);**

**num /= 10;**

**}**

**return res == orgNum;**

**}**

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

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

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

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

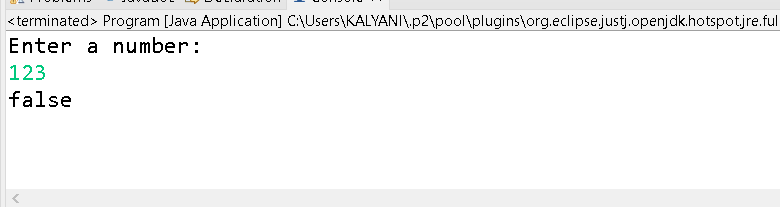
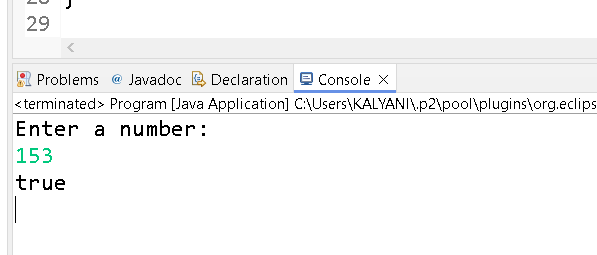
**System.*out*.println(*isArmstrong*(num));**

**sc.close();**

**}**

**}**

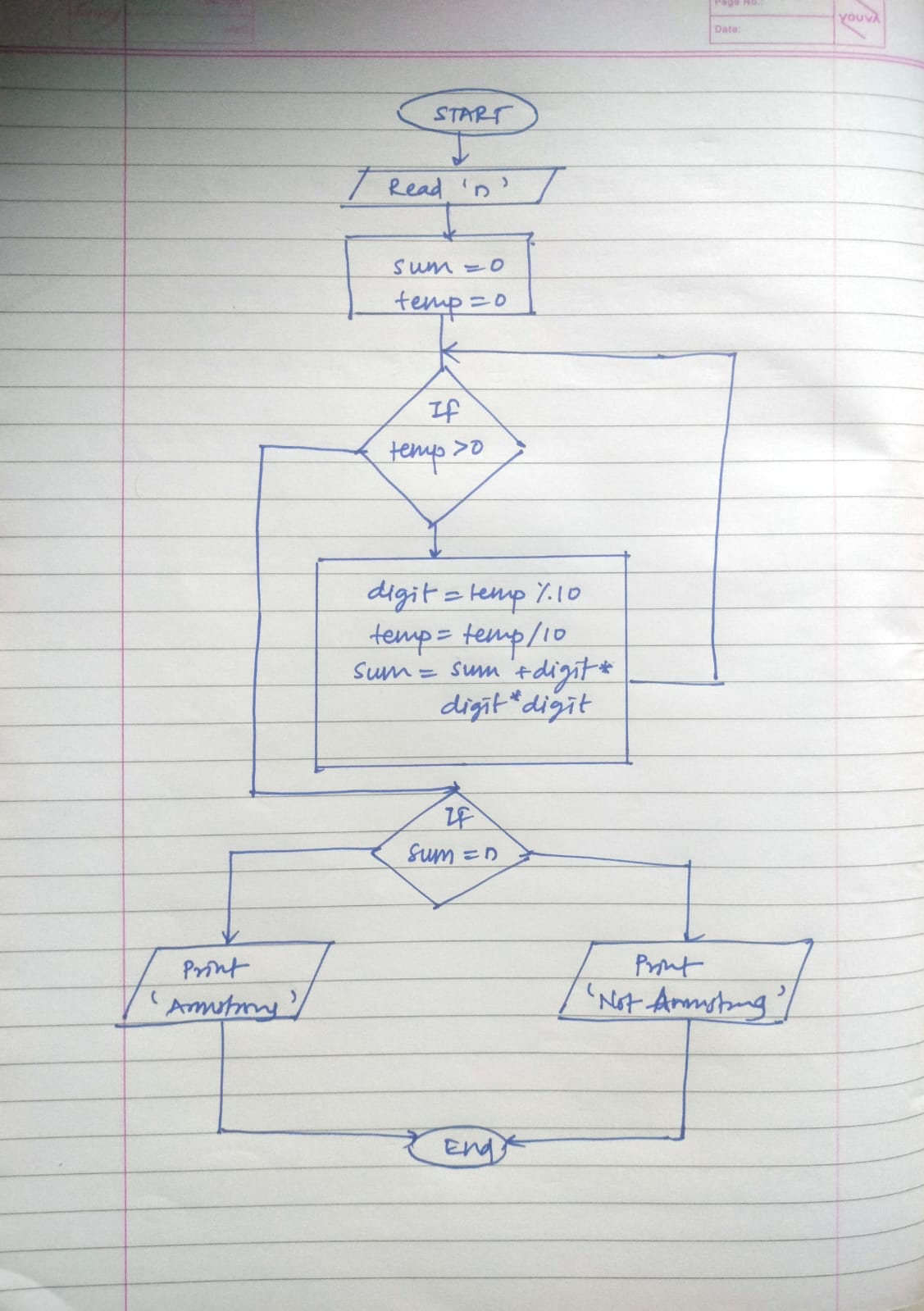
**Output:**

****

Time and Space complexity

TC=O(n^2)                                                                                                                                          SC= O(1)

**Flow chart:**



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

**Program :**

**package** org\_example\_2;

**import** java.util.Scanner;

**public** **class** PrimeNumber {

**public** **static** **boolean** isPrime(**int** num) {

**if** (num <= 1) {

**return** **false**;

}

**for** (**int** i = 2; i <= Math.*sqrt*(num); i++) {

**if** (num % i == 0) {

**return** **false**;

}

}

**return** **true**;

}

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

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

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

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

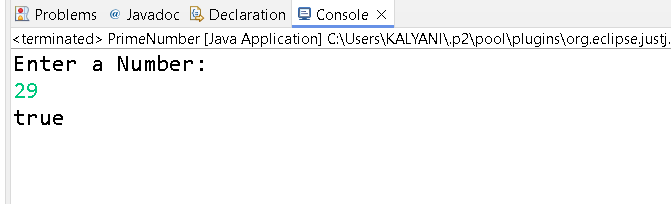
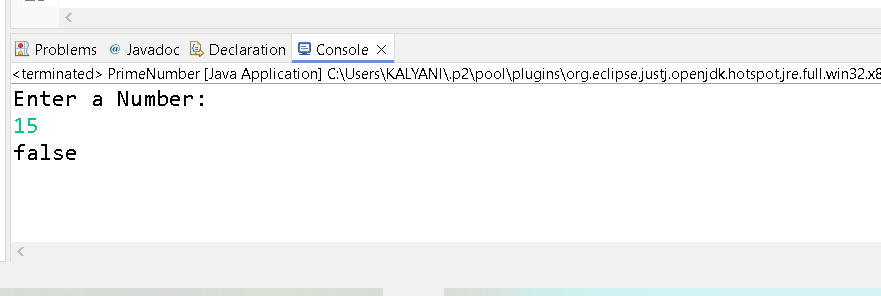
System.***out***.println(*isPrime*(num));

sc.close();

}

}

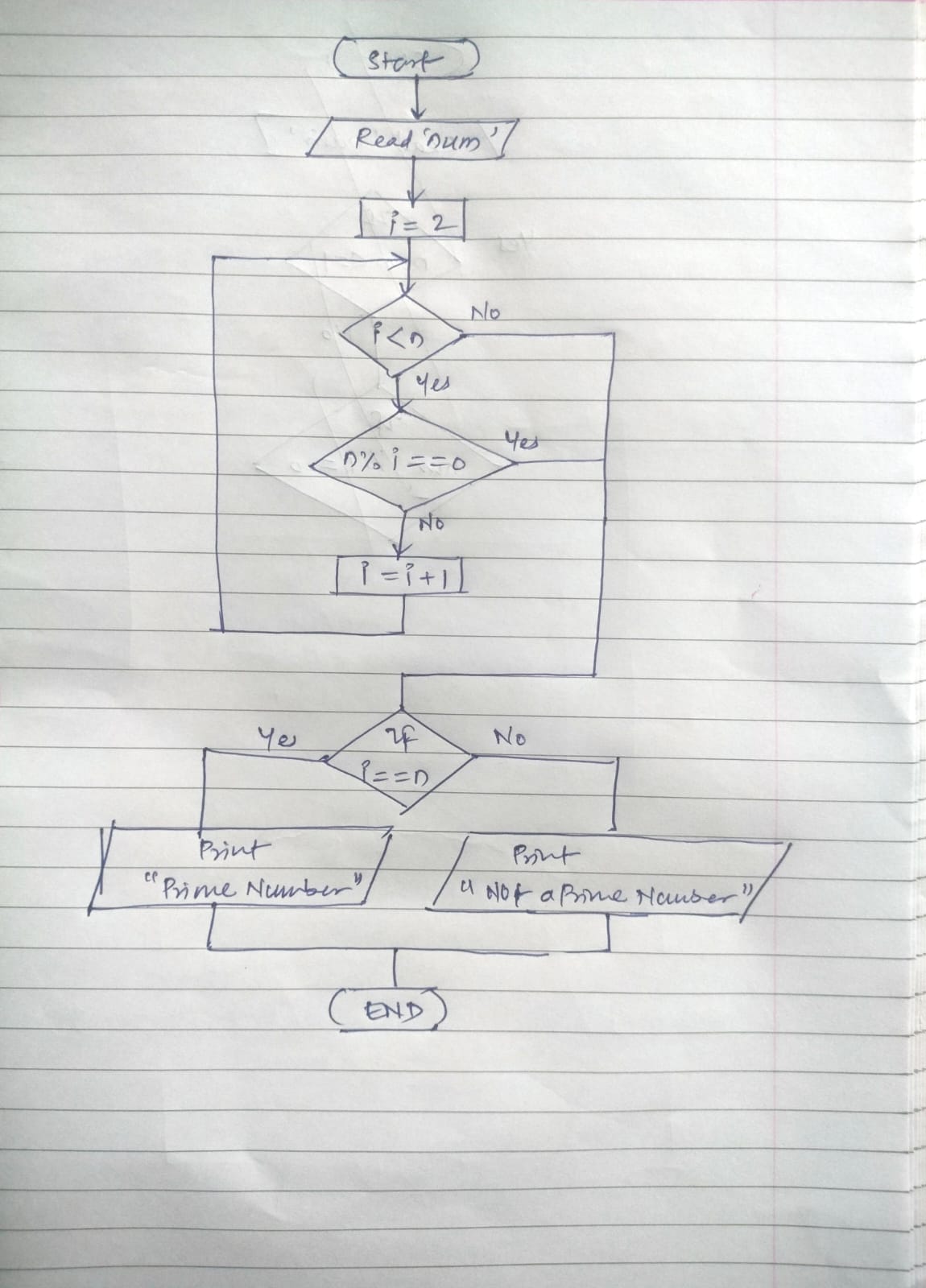
**Output:**



Time and Space complexity

T C = O(Square root of n)

S C = O(1)



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

**Program :**

**package** org\_example\_3;

**import** java.util.Scanner;

**public** **class** Factorial {

**public** **static** **int** factorial(**int** num) {

**int** res = 1;

**for** (**int** i = 1; i <= num; i++) {

res \*= i;

}

**return** res;

}

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

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

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

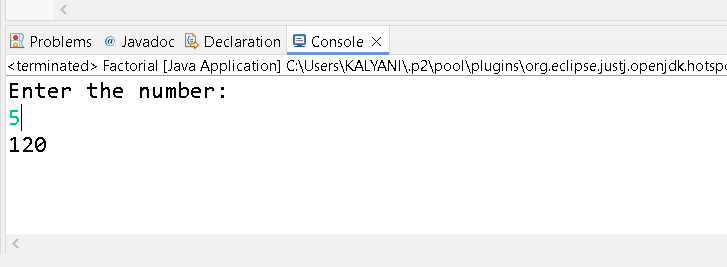
**int** num = sc.nextInt();

sc.close();

System.***out***.println(*factorial*(num));

}

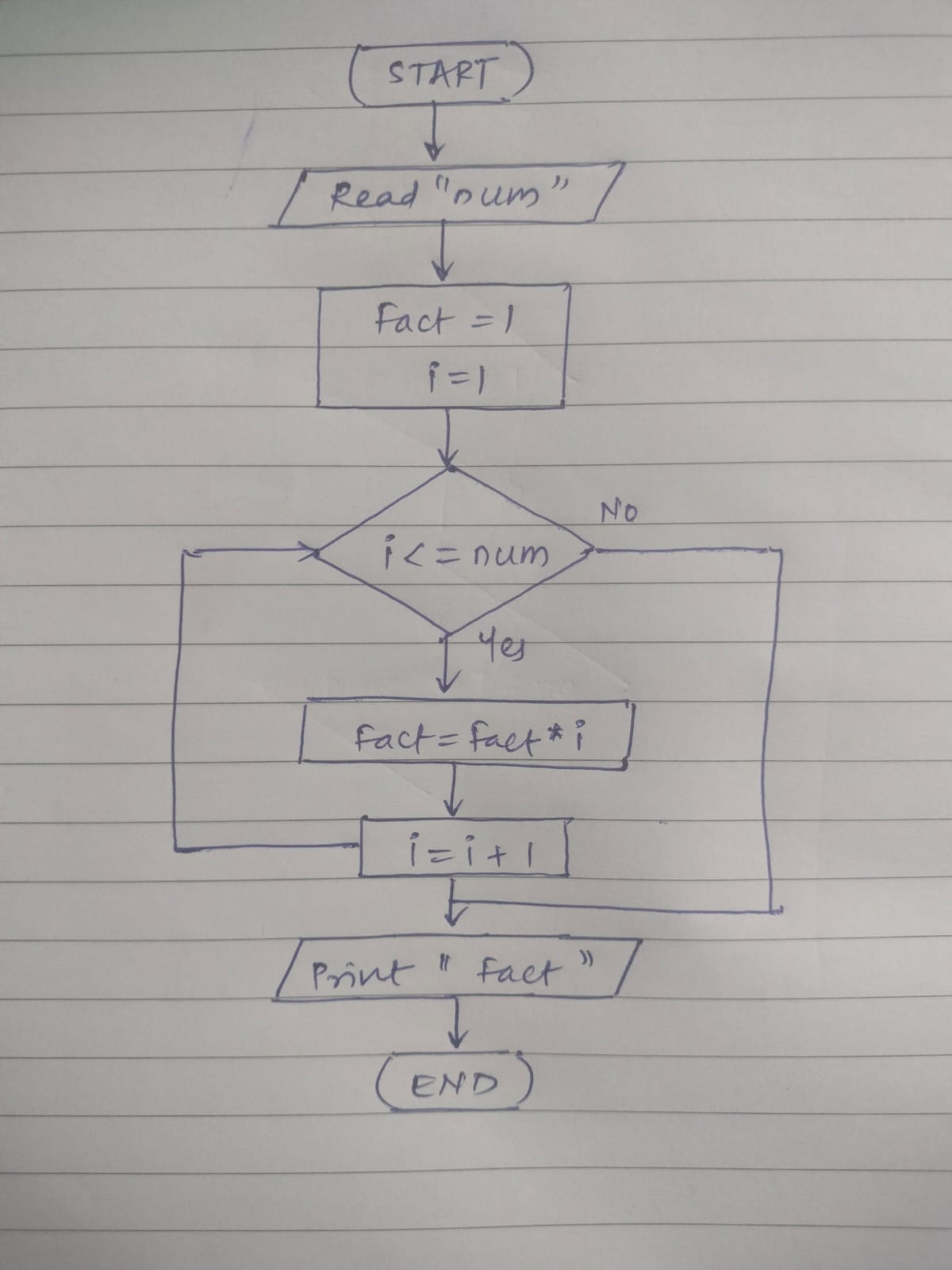
}

**Output:**

Time and Space complexity

T C = O( n )

S C = O( 1 )



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]

**Program :**

**package** org\_example\_4;

**import** java.util.Scanner;

**public** **class** FibonacciSeries {

**public** **static** **void** printRecord(**int** num) {

**int** first = 0, second = 1;

**if** (num > 0) {

System.***out***.print("[" + first);

}

**if** (num > 1) {

System.***out***.print(", " + second);

}

**for** (**int** i = 3; i <= num; i++) {

**int** next = first + second;

System.***out***.print(", " + next);

first = second;

second = next;

}

System.***out***.println("]");

}

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

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

System.***out***.println("Enter number upto which you want seris: ");

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

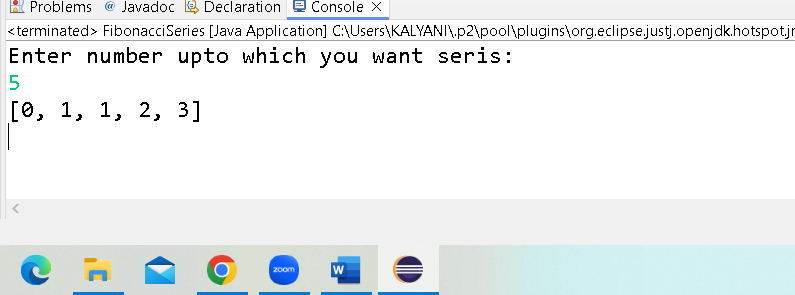
*printRecord*(num);

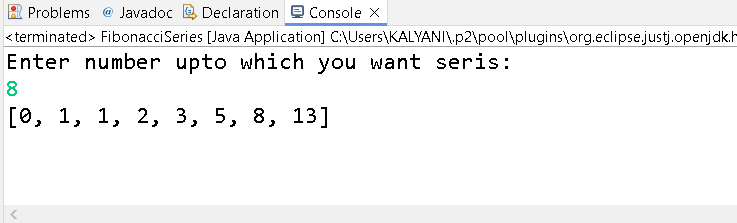
sc.close();

}

}

**Output:**

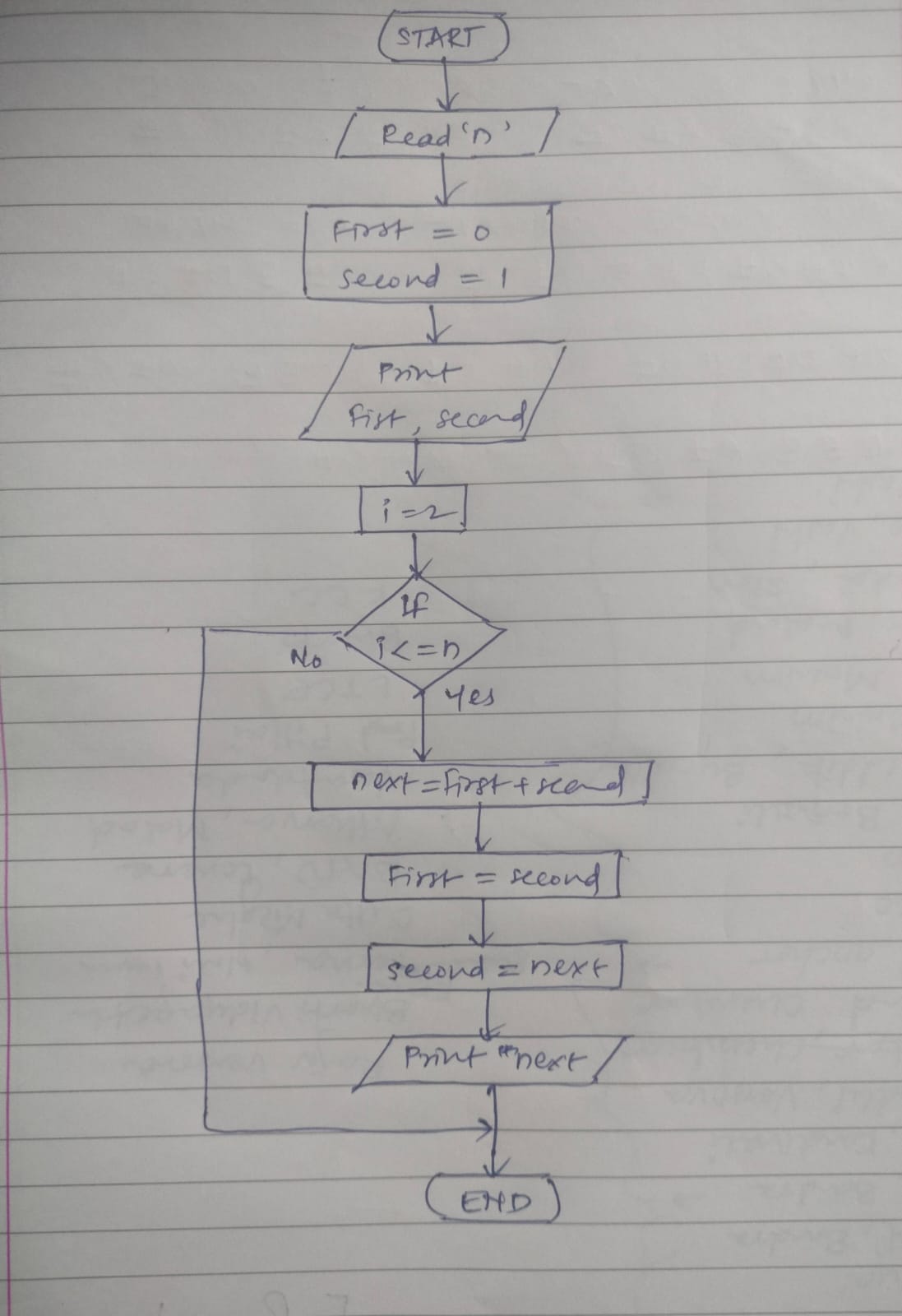




Time and Space complexity

T C = O( n )

S C = O( 1 )



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

**Program :**

**package** org\_example\_5;

**import** java.util.Scanner;

**public** **class** Program {

**public** **static** **int** gcd(**int** num1, **int** num2) {

**while** (num2 != 0) {

**int** temp = num2;

num2 = num1 % num2;

num1 = temp;

}

**return** num1;

}

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

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

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

**int** num1 = sc.nextInt();

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

**int** num2 = sc.nextInt();

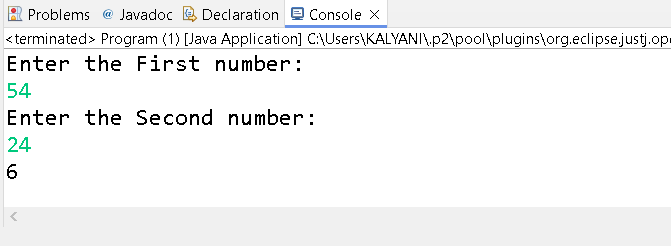
System.***out***.println(*gcd*(num1,num2));

sc.close();

}

}

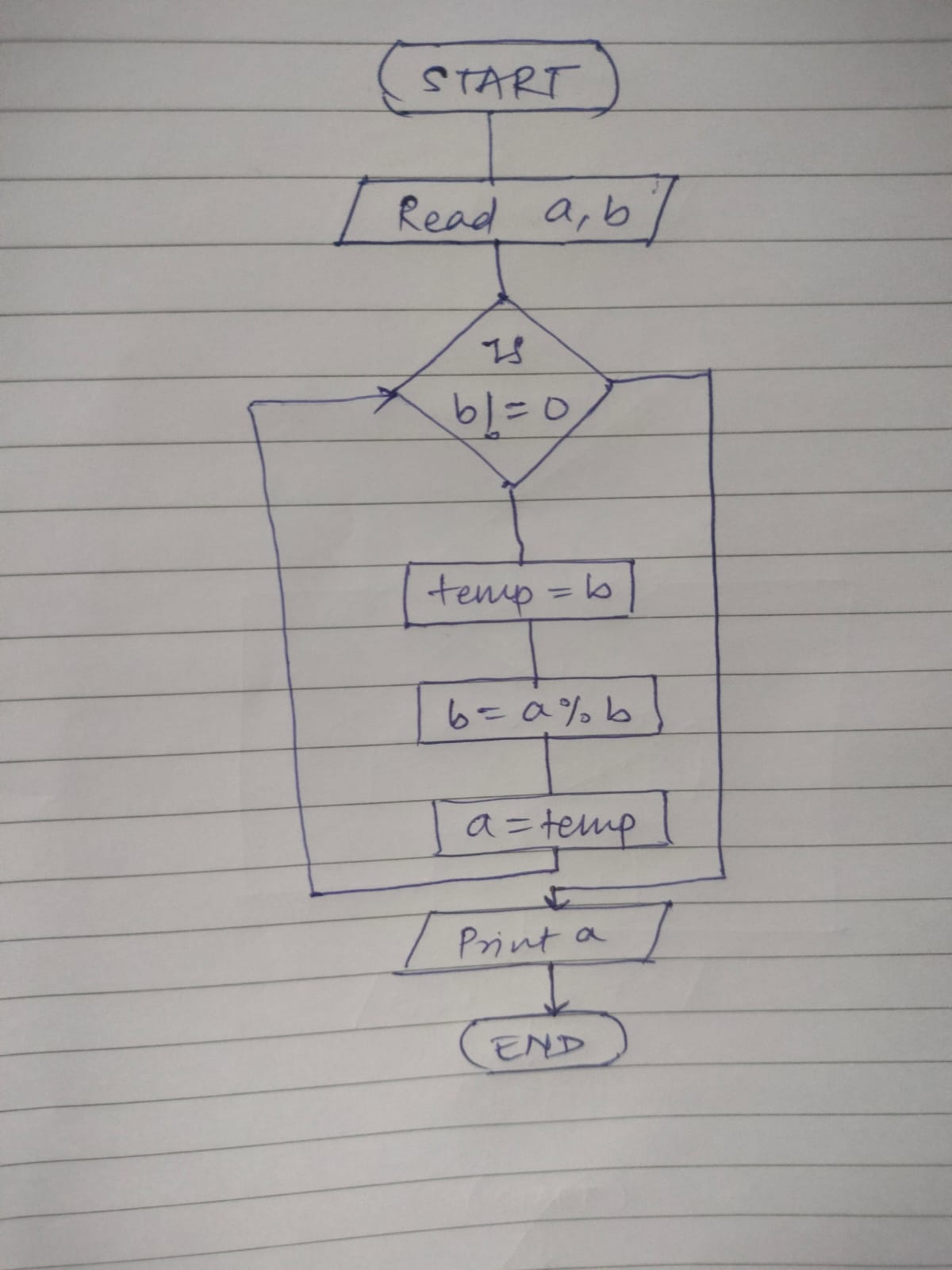
**Output:**

****

Time and Space complexity

T C = O( log(min(num1,num2)) )

S C = O( 1 )



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

**Program :**

**package** org\_example\_6;

**import** java.util.Scanner;

**public** **class** SquareRoot {

**public** **static** **int** findSquareRoot(**int** x) {

**int** result = 0;

**for** (**int** i = 1; i \* i <= x; i++) {

result = i;

}

**return** result;

}

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

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

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

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

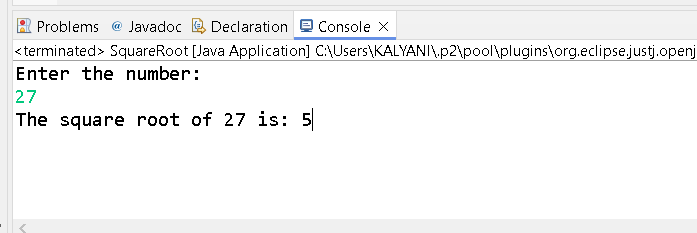
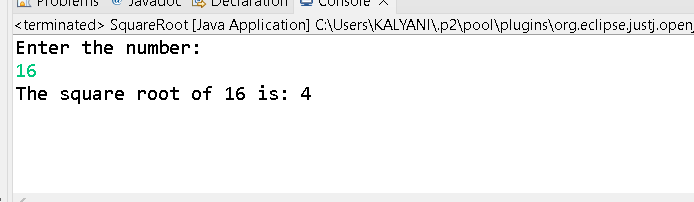
System.***out***.println("The square root of " + x + " is: " + *findSquareRoot*(x));

sc.close();

}

}

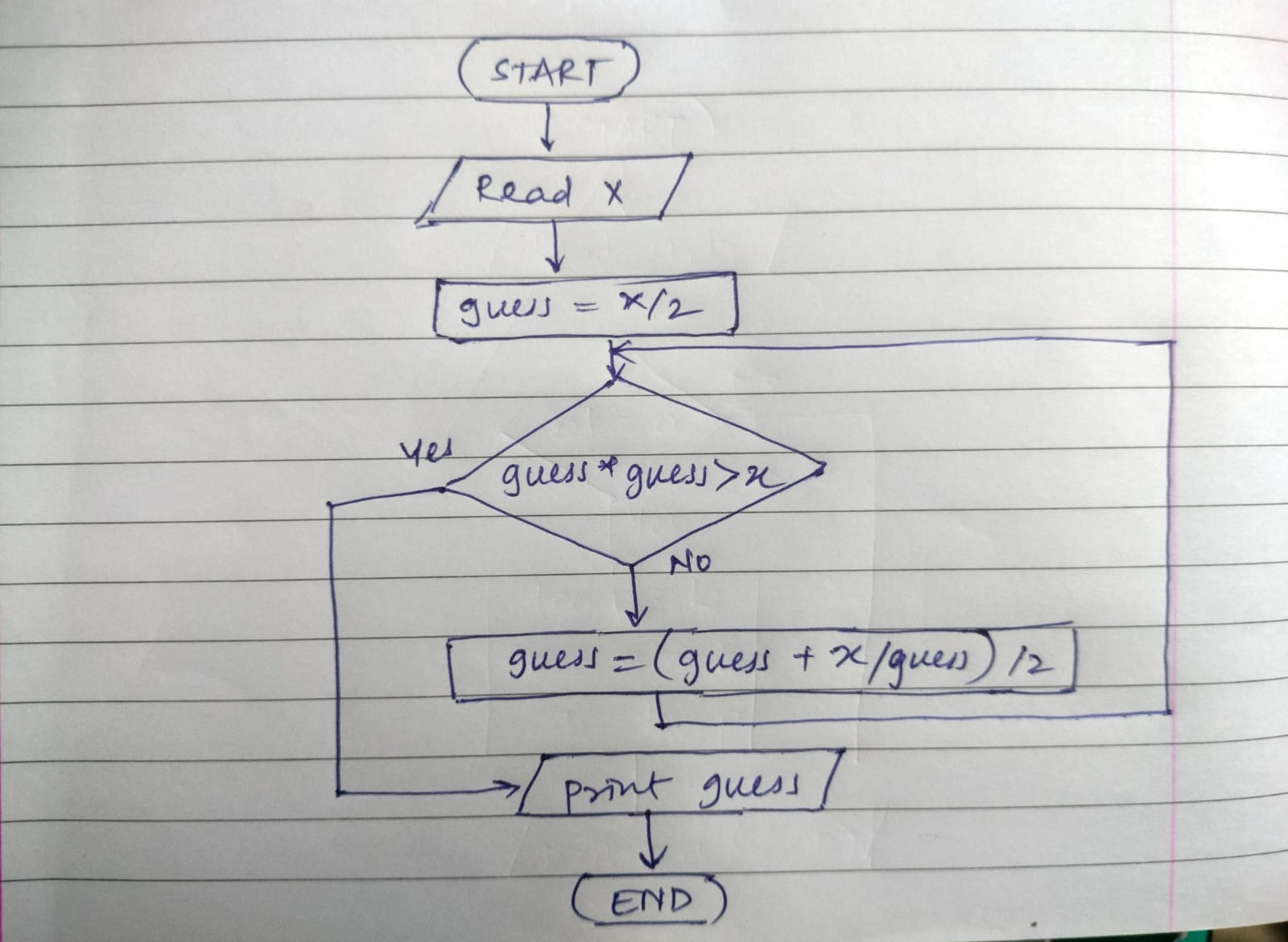
**Output:**



Time and Space complexity

T C = O(Square root of n )

S C = O( 1 )



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']

**Program :**

**package** org\_example\_7;

**import** java.util.Scanner;

**public** **class** Program {

**public** **static** **void** findRepeatedCharacters(String str) {

**boolean**[] a = **new** **boolean**[256];

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

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

**char** c = str.charAt(i);

**if** (!a[c]) {

**for** (**int** j = i + 1; j < str.length(); j++) {

**if** (str.charAt(j) == c) {

System.***out***.print(c + " ");

**break**;

}

}

a[c] = **true**;

}

}

System.***out***.println("]");

}

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

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

System.***out***.println("Enter a String: ");

String str = sc.nextLine();

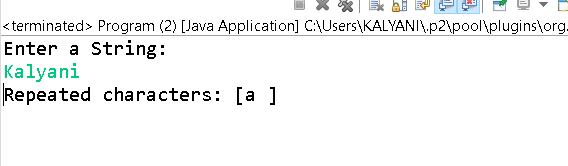
*findRepeatedCharacters*(str);

sc.close();

}

}

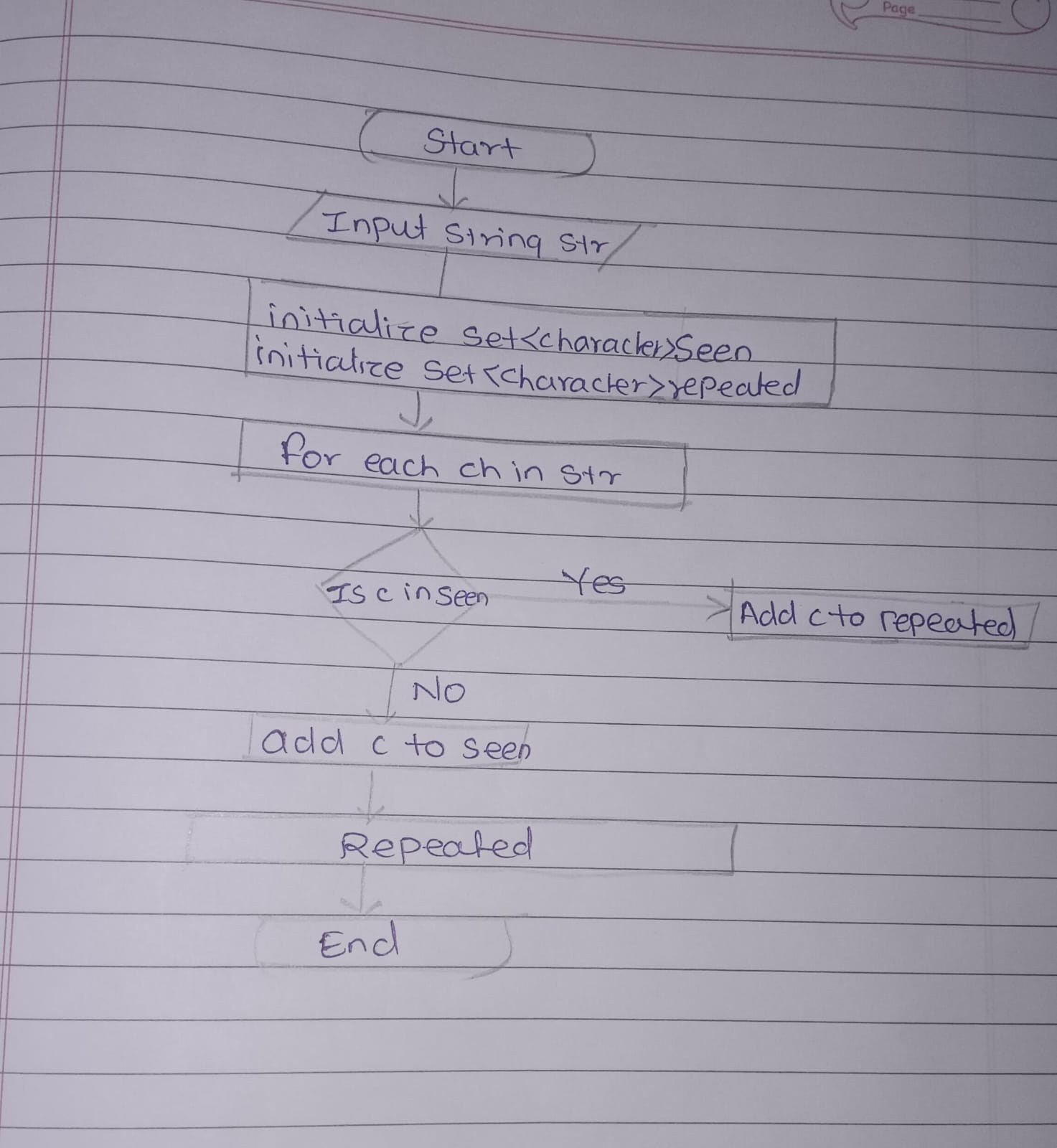
**Output:**



Time and Space complexity

T C = O( n ^2)

S C = O( 1 )



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

**Program :**

**package** org\_example\_8;

**import** java.util.Scanner;

**public** **class** Program8 {

**public** **static** Character firstNonRepeatedCharacter(String str) {

**int**[] charCount = **new** **int**[256];

**for** (**char** c : str.toCharArray()) {

charCount[c]++;

}

**for** (**char** c : str.toCharArray()) {

**if** (charCount[c] == 1) {

**return** c;

}

}

**return** **null**;

}

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

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

System.***out***.println("Enter a string: ");

String str = sc.nextLine();

Character result = *firstNonRepeatedCharacter*(str);

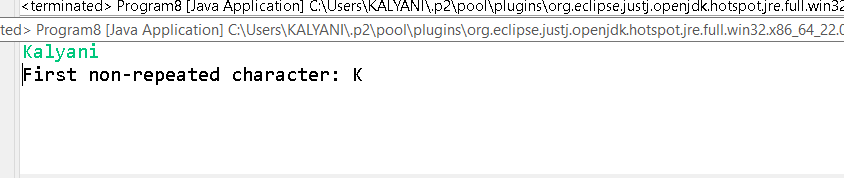
System.***out***.println("First non-repeated character: " + result);

sc.close();

}

}

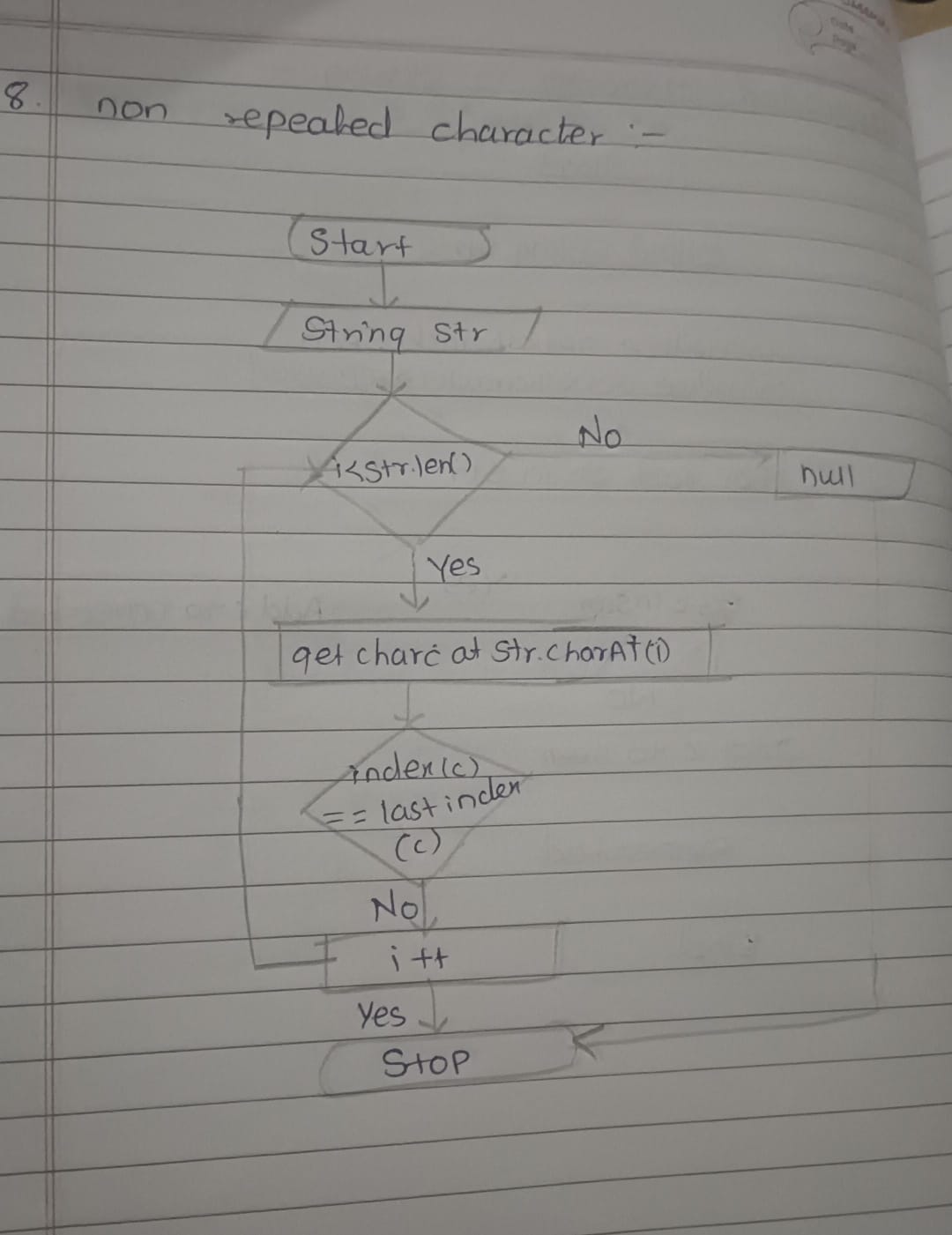
**Output:**



Time and Space complexity

T C = O( n )

S C = O( 1 )



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

**Program :**

**package** org\_example\_9;

**import** java.util.Scanner;

**public** **class** Palindrome {

**public** **static** **boolean** isPalindrome(**int** num) {

**if** (num < 0) {

**return** **false**;

}

**int** original = num;

**int** reversed = 0;

**while** (num > 0) {

**int** digit = num % 10;

reversed = reversed \* 10 + digit;

num /= 10;

}

**return** original == reversed;

}

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

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

System.***out***.println("Enter the Number: ");

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

**boolean** result = *isPalindrome*(num);

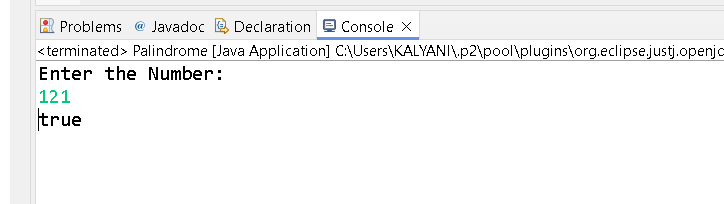
System.***out***.println(result);

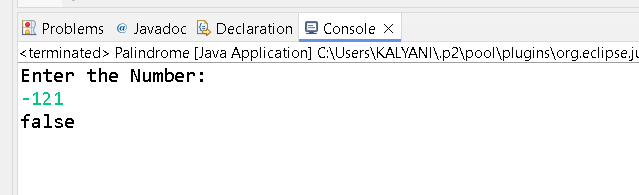
sc.close();

}

}

**Output:**

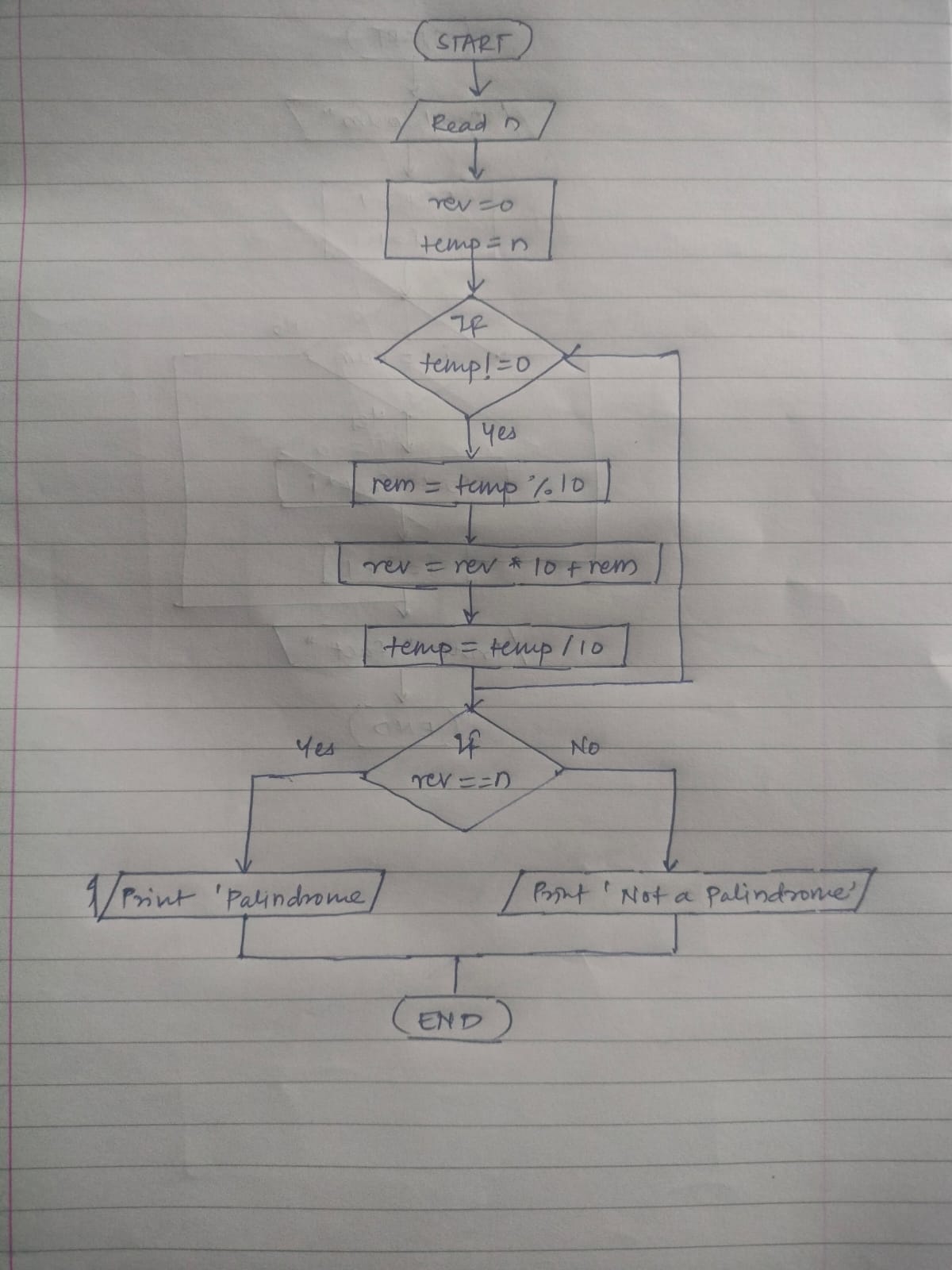
****



Time and Space complexity

T C = O( k ) where k is the number of digits

S C = O( 1 )



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

**Program :**

**package** org\_example\_10;

**import** java.util.Scanner;

**public** **class** LeapYear {

**public** **static** **boolean** isLeapYear(**int** year) {

**return** (year % 4 == 0 && year % 100 != 0) || (year % 400 == 0);

}

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

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

System.***out***.println("Enter the year: ");

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

**boolean** isLeap = *isLeapYear*(year);

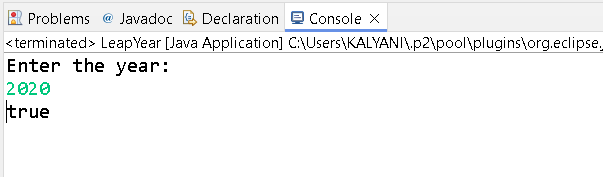
System.***out***.println(isLeap);

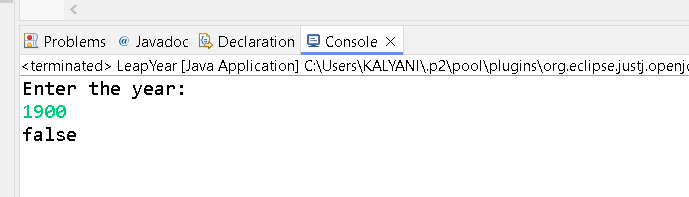
sc.close();

}

}

**Output:**





Time and Space complexity

T C = O( 1 )

S C = O( 1 )

