**Assignment 1**

1. Armstrong Number

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

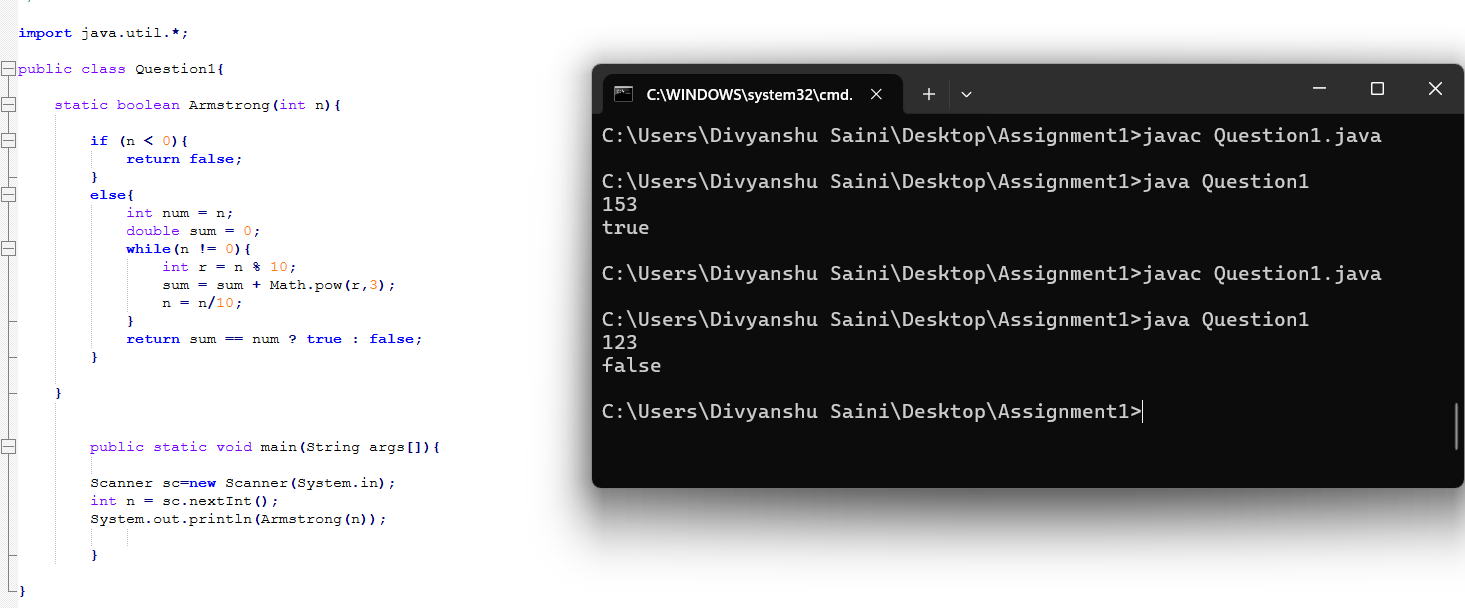
Test Cases:

Input: 153

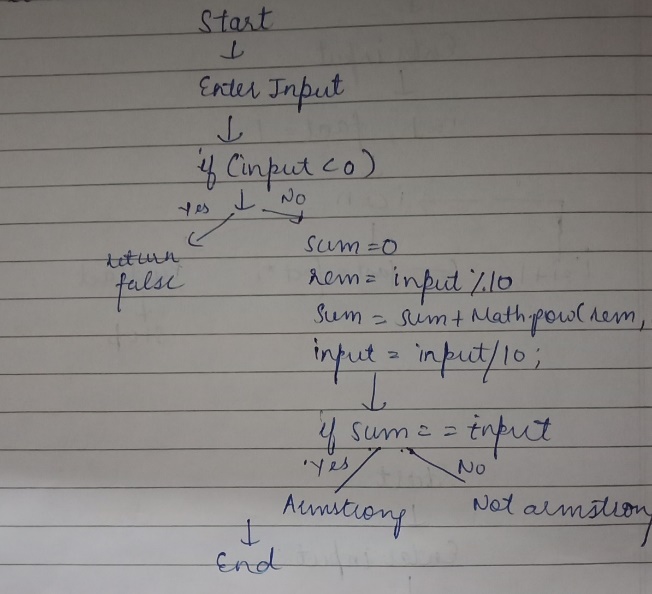
Output: true

Input: 123

Output: false

****

**Flowchart-**

****

**Explanation –** Firstly, I created a method called as Armstrong of Boolean return type. Then inside it I’ve checked the condition that whether the input is less than 0. If it is greater than 0 then false is returned. If it is not then it will move to next condition. There is a while loop which continues till the input is not equal to 0 and inside it I’ve calculated the remainder and sum. If the sum is equal to the input that is entered by the user then it will return true otherwise it will return false. In the main method with the help of Scanner class I’ve taken the input from the user and invoked the Armstrong method.

**Time Complexity –** O(log n)

**Space Complexity-** O(1)

2. Prime Number

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

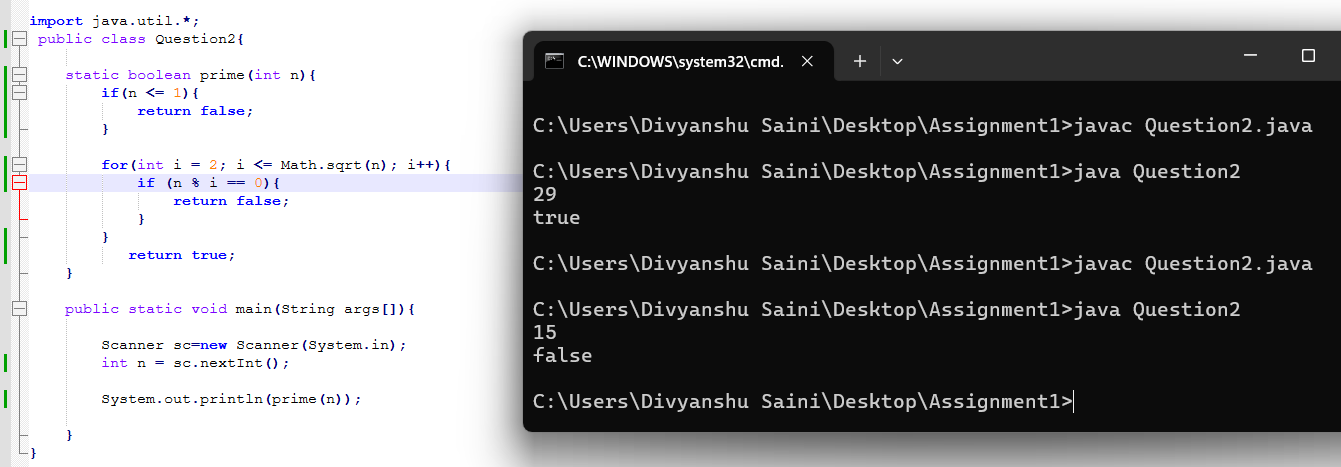
Test Cases:

Input: 29

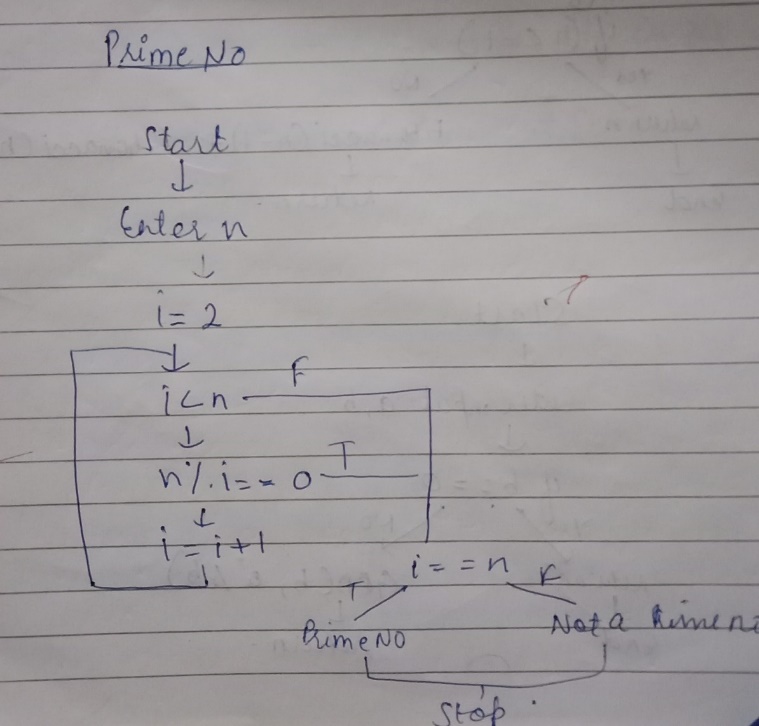
Output: true

Input: 15

Output: false



**Flowchart-**



**Explanation –** Firstly, I have created a method called as Prime of Boolean return type. Then inside it I’ve checked the condition that whether the input is less than 1. If it is less than 1 then false is returned. If it is not then it will move to next condition. Then I run a for loop which checks if input is divisible by any integer i from 2 to square root of n. If input is divisible then the false is returned otherwise it will return true. In the main method with the help of Scanner class I’ve taken the input from the user and invoked the Prime method.

**Time Complexity –**  O (√n)

**Space Complexity-** O(1)

3. Factorial

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

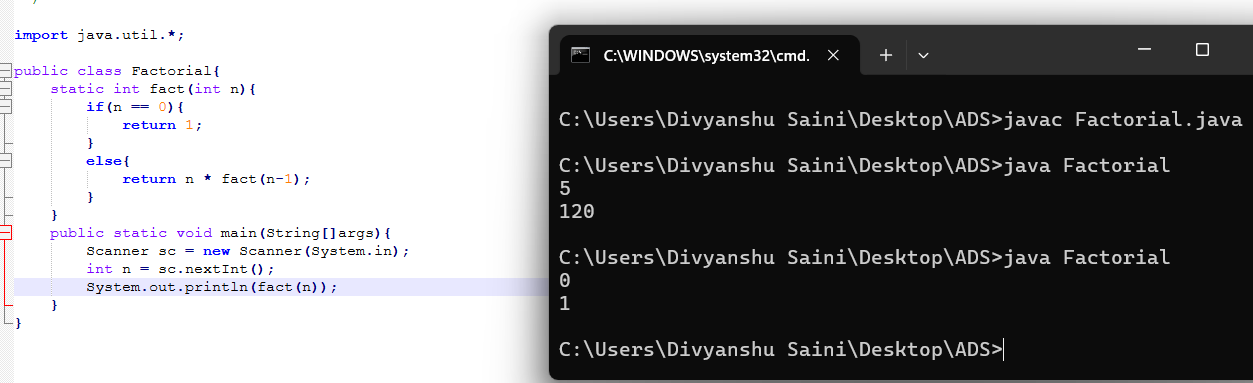
Test Cases:

Input: 5

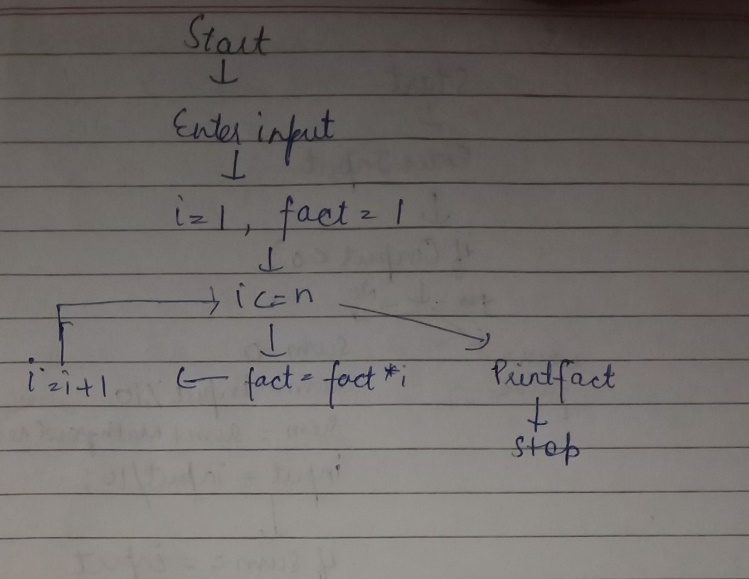
Output: 120

Input: 0

Output: 1

****

**Flowchart-**

****

**Explanation -** Firstly, I have created a method called as fact of int return type. Then inside it I’ve checked the condition that whether the input is equal to 0. If it is equal to 0 then it will return 1. If input is not 0, the method calculates n \* fact(n - 1). This is a recursive call where the method calls itself with a value of n - 1. This process continues until n reaches 0, at which point the base case returns 1. In the main method with the help of Scanner class I’ve taken the input from the user and invoked the fact method.

**Time Complexity –**  O (n)

**Space Complexity-** O(n)

4. Fibonacci Series

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

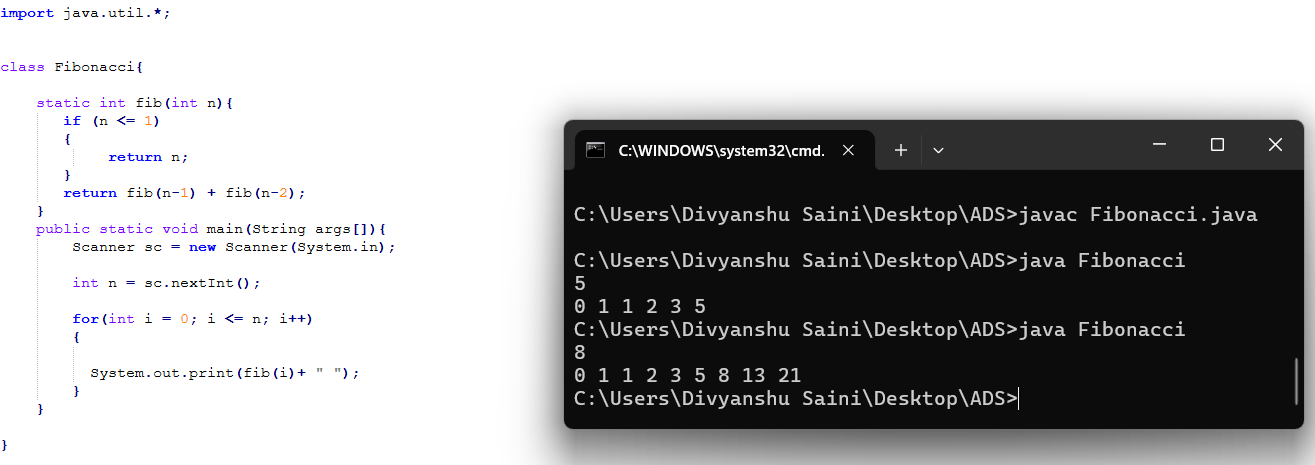
Test Cases:

Input: n = 5

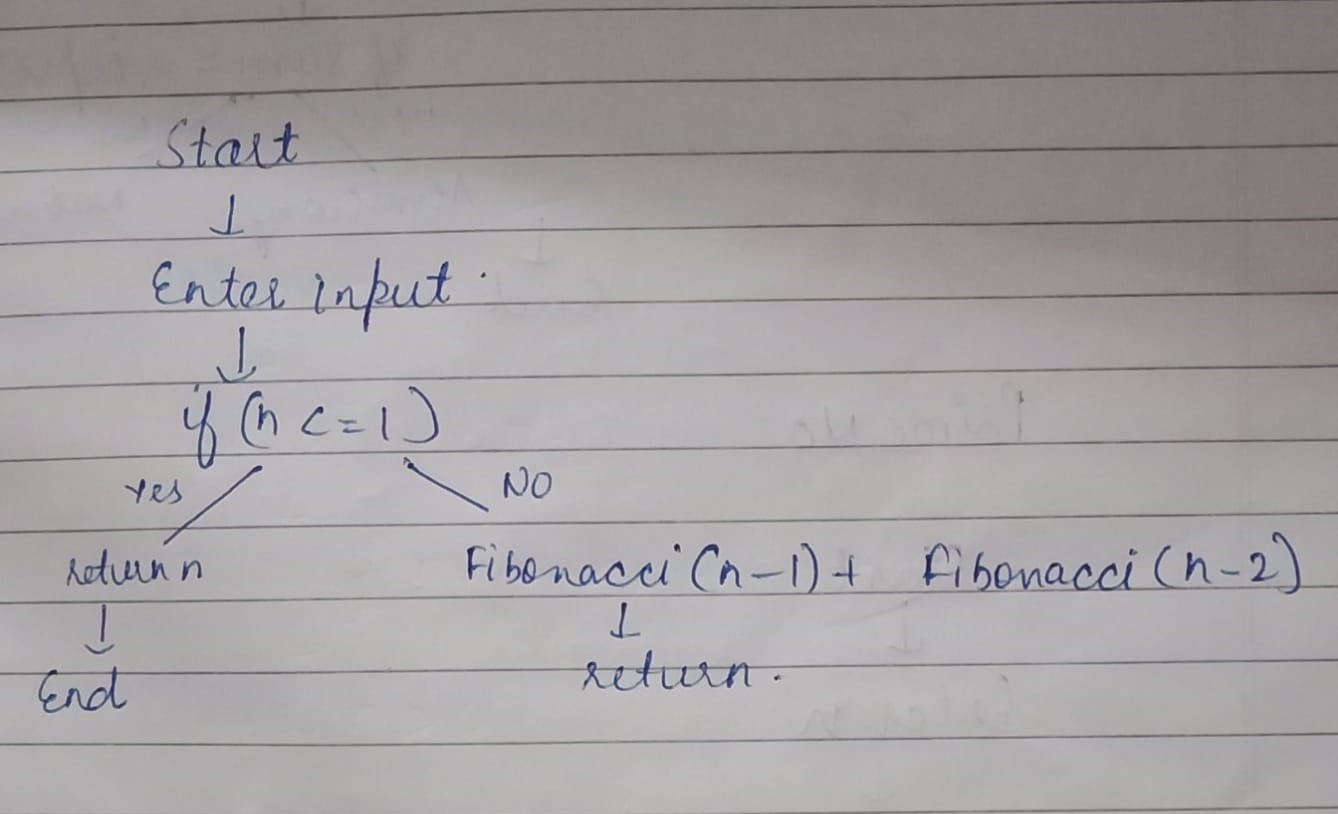
Output: [0, 1, 1, 2, 3]

Input: n = 8

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

****

**Flowchart-**

****

**Explanation -** Firstly, I have created a method called as fib of int return type. Then inside it I’ve checked the condition that whether the input is less than 1. If it is less than 1 then input is returned. If it is not less than 1 ,the method will calculate fib(n-1) + fib(n-2). This is a recursive call where the method calls itself. In the main method with the help of Scanner class I’ve taken the input from the user. There is a for loop which iterates from 0 to n. For each iteration, fib(i) is called to calculate the ith Fibonacci number.

**Time Complexity –**  O (2^n)

**Space Complexity-**  O(n)

5. Find GCD

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

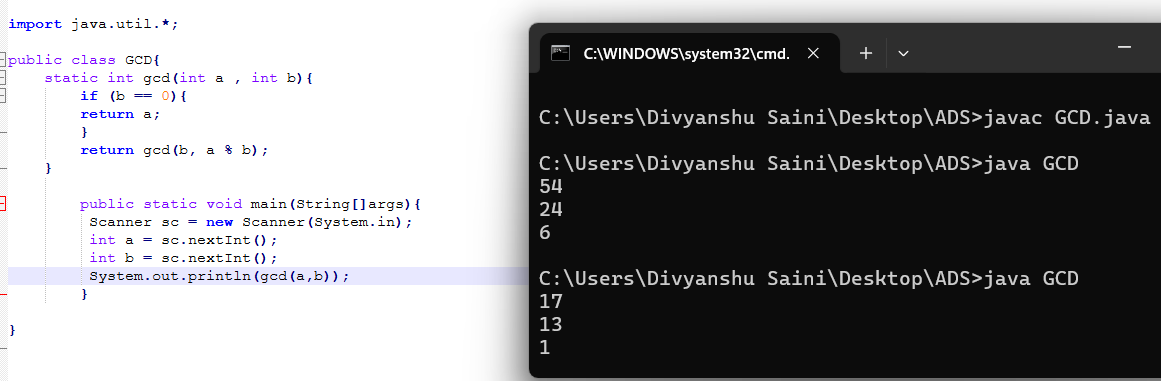
Test Cases:

Input: a = 54, b = 24

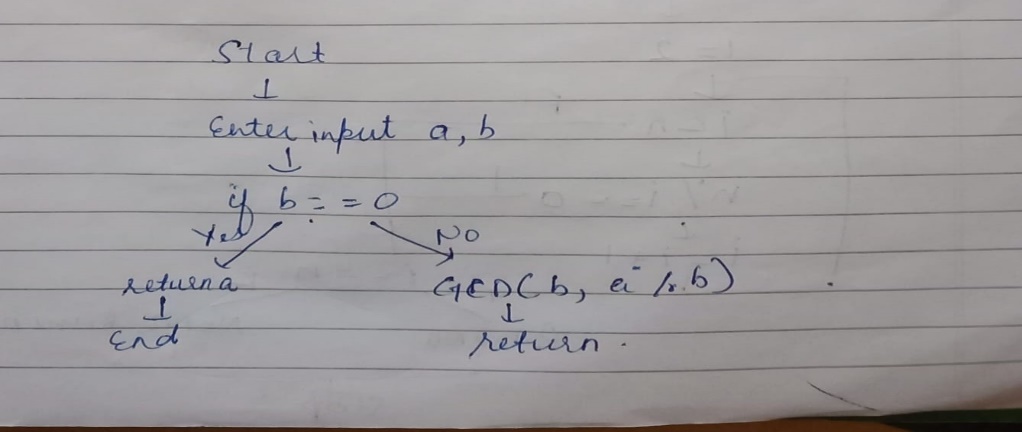
Output: 6

Input: a = 17, b = 13

Output: 1

****

**Flowchart-**

****

**Explanation -** Firstly, I have created a method called as gcd of int return type. Then inside it I’ve checked the condition that whether one of the input i.e. b is equal to 0. If it is equal to 0 then only one input i.e. a is returned. If it is not equal to 0 ,the method will calculate gcd(b, a%b). This is a recursive call where the method calls itself. In the main method with the help of Scanner class I’ve taken the input from the user and invoked the gcd method.

**Time Complexity –**  O (log min(a , b))

**Space Complexity-** O (log min(a , b))

6. Find Square Root

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

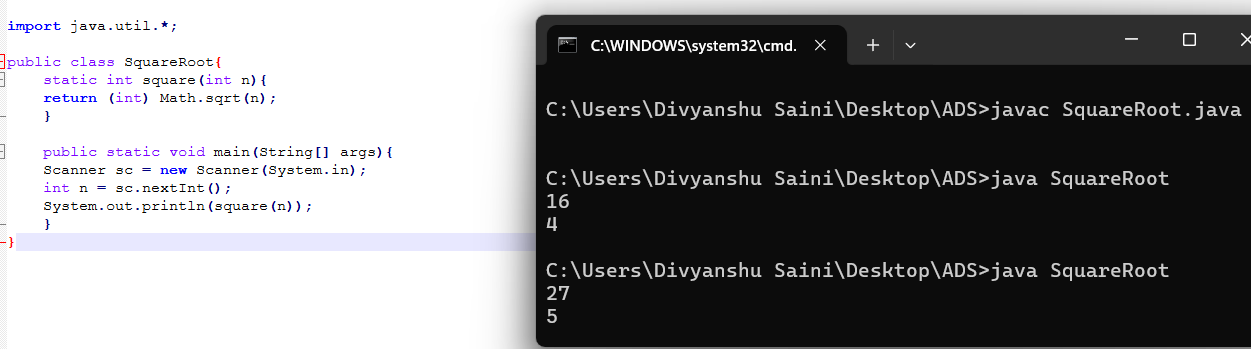
Test Cases:

Input: x = 16

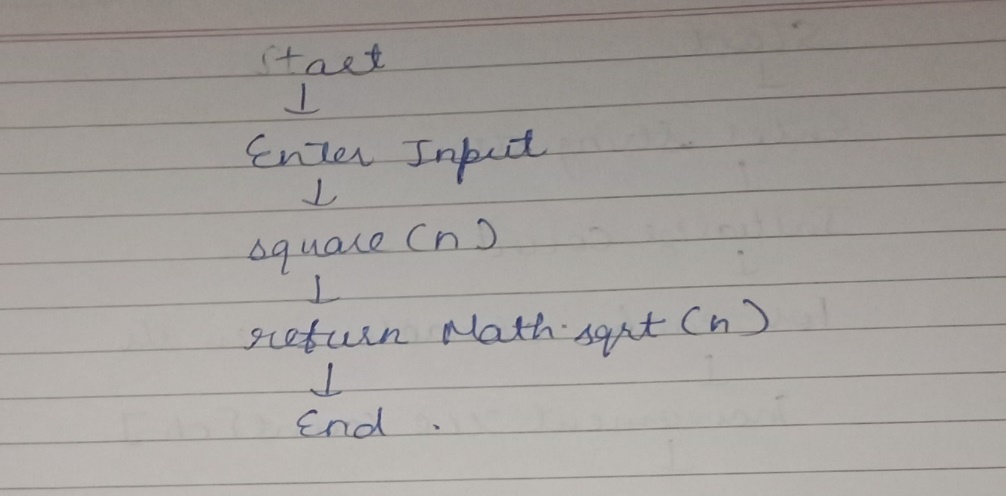
Output: 4

Input: x = 27

Output = 5



**Flowchart-**

****

**Explanation -** Firstly, I have created a method called as square of int return type. Inside this method I’ve used Math.sqrt(n) which calculates the square root of n and returns it as a double. The (int) before Math.sqrt(n) is a type cast, which converts the result from a double to int. In the main method with the help of Scanner class I’ve taken the input from the user and invoked the square method.

**Time Complexity –**  O (1)

**Space Complexity-** O (1)

7. Find Repeated Characters in a String

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

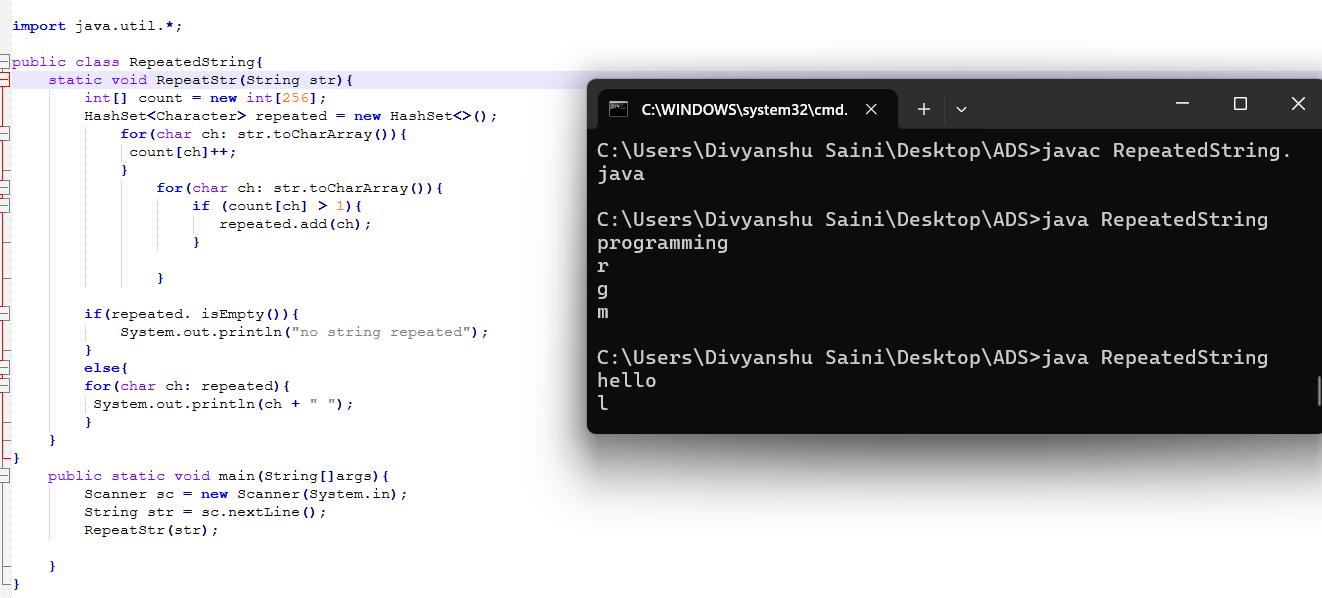
Test Cases:

Input: "programming"

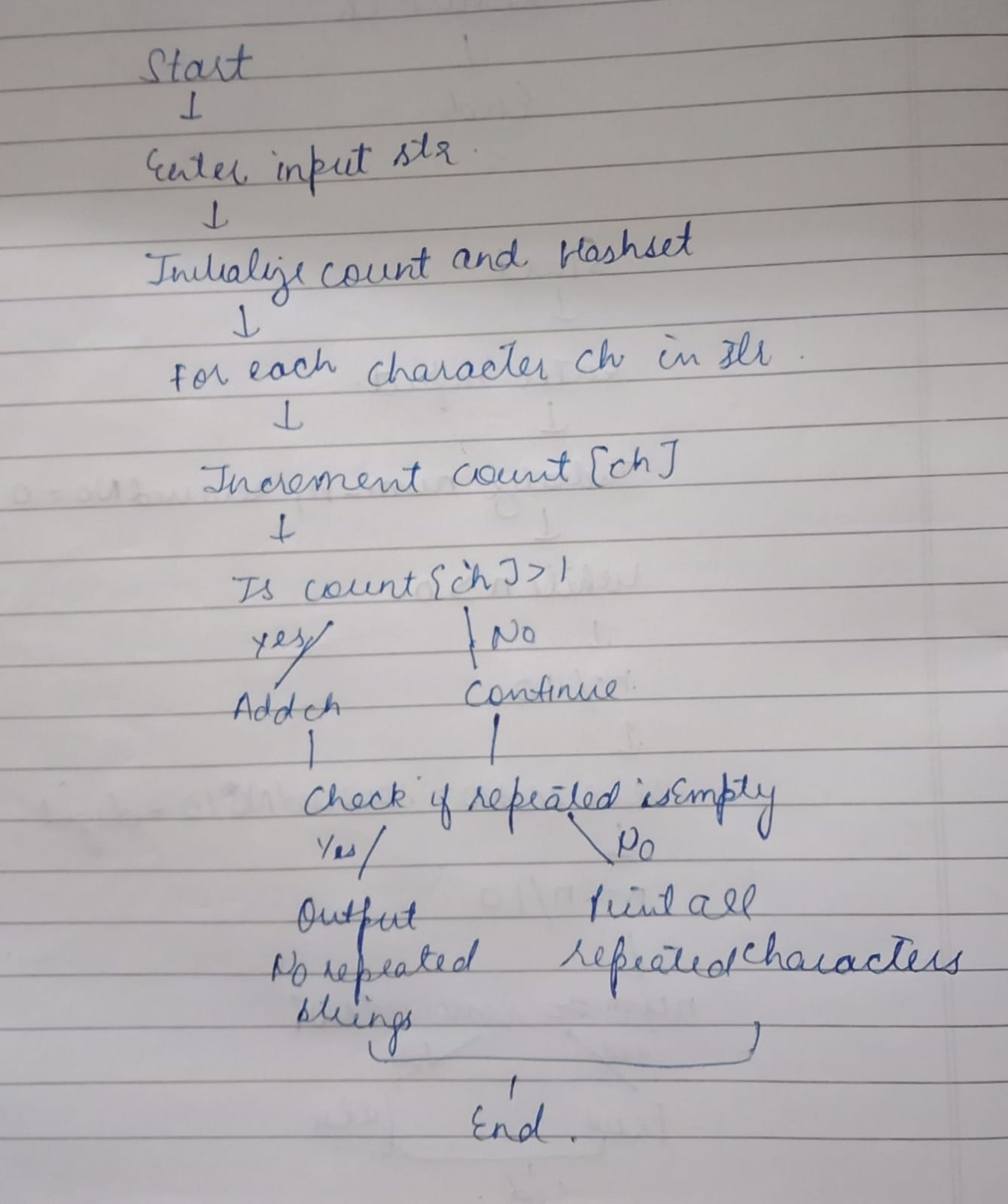
Output: ['r', 'g', 'm']

Input: "hello"

Output: ['l']

****

**Flowchart-**

****

**Explanation -** Firstly, I have created a method called as RepeatStr of void return type. Then I initialized an array count of size 256 to store the frequency of each character in the string. The size 256 is used because it covers all possible extended ASCII characters. Created a HashSet named repeated to store characters that occur more than once in the string. In the first for loop the string str is converted into a character array and iterates over each character ch. count[ch]++ increments the count for each character in the count array. In the second for loop it checks if the character appears more than once and adds the repeated character to the HashSet. Then if (repeated.isEmpty()) condition checks if there are no repeated characters. If true, it prints "No repeated Strings." And if repeated characters exist, the program iterates over the repeated HashSet and prints each repeated character. In the main method with the help of Scanner class I’ve taken the input from the user and invoked the RepeatStr method.

**Time Complexity –**  O (n)

**Space Complexity-** O (n)

8. First Non-Repeated Character

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

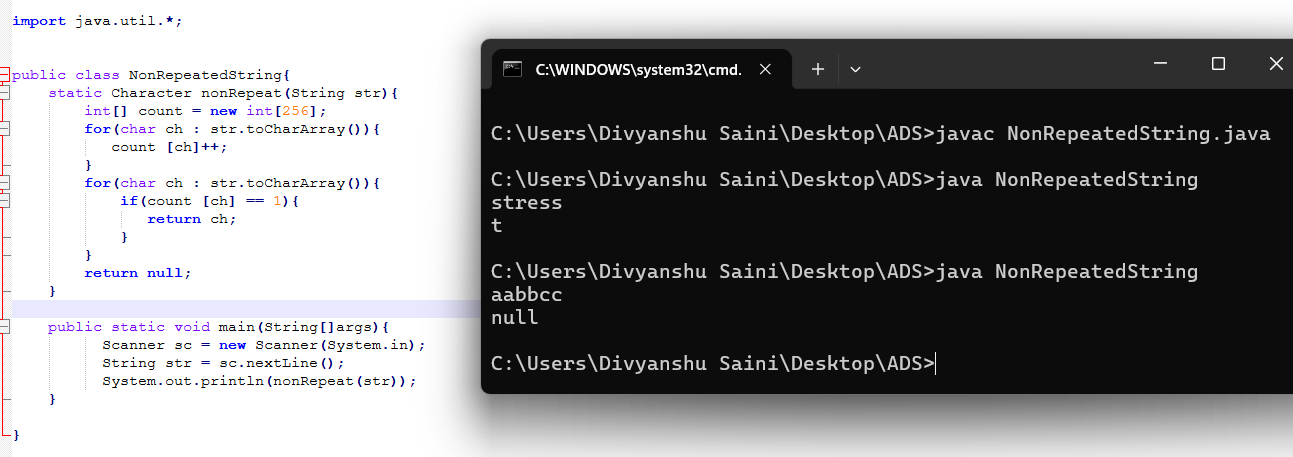
Test Cases:

Input: "stress"

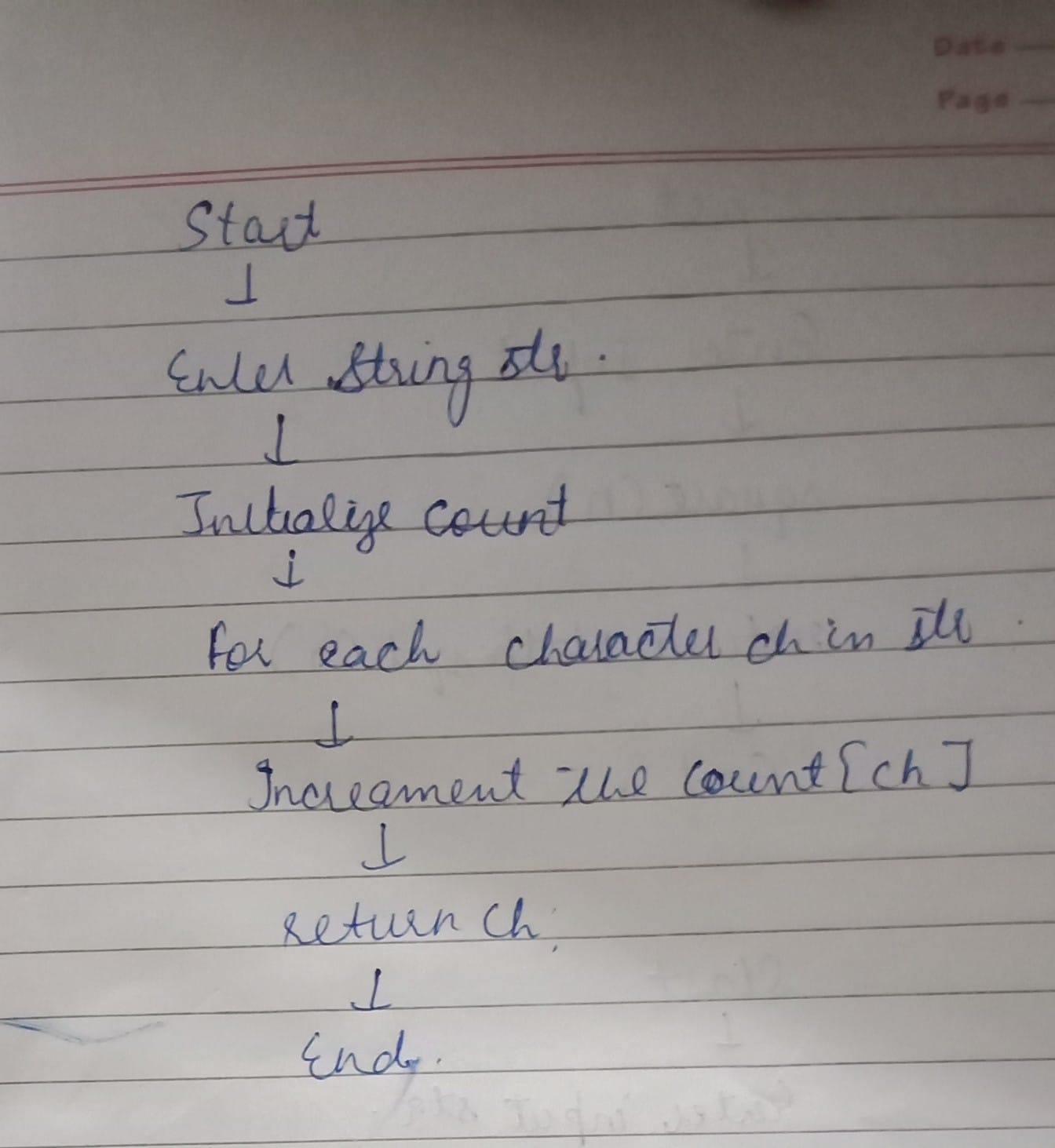
Output: 't'

Input: "aabbcc"

Output: null

****

**Flowchart-**

****

**Explanation- -** Firstly, I have created a method called as nonRepeat of Character return type. Then I initialized an array count of size 256 to store the frequency of each character in the string. The size 256 is used because it covers all possible extended ASCII characters. . In the first for loop the string str is converted into a character array and iterates over each character ch. count[ch]++ increments the count for each character in the count array. In the second for loop it checks if any character appeared only once and than that ch is returned. And if there are no characters that appeared only once, it will return null. In the main method with the help of Scanner class I’ve taken the input from the user and invoked the nonRepeat method.

**Time Complexity –**  O (n)

**Space Complexity-** O (n)

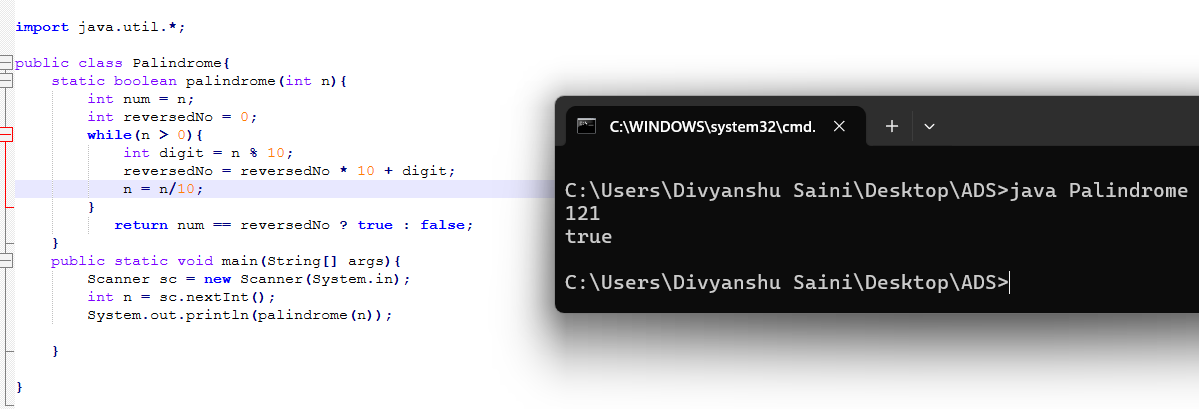
9. Integer Palindrome

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

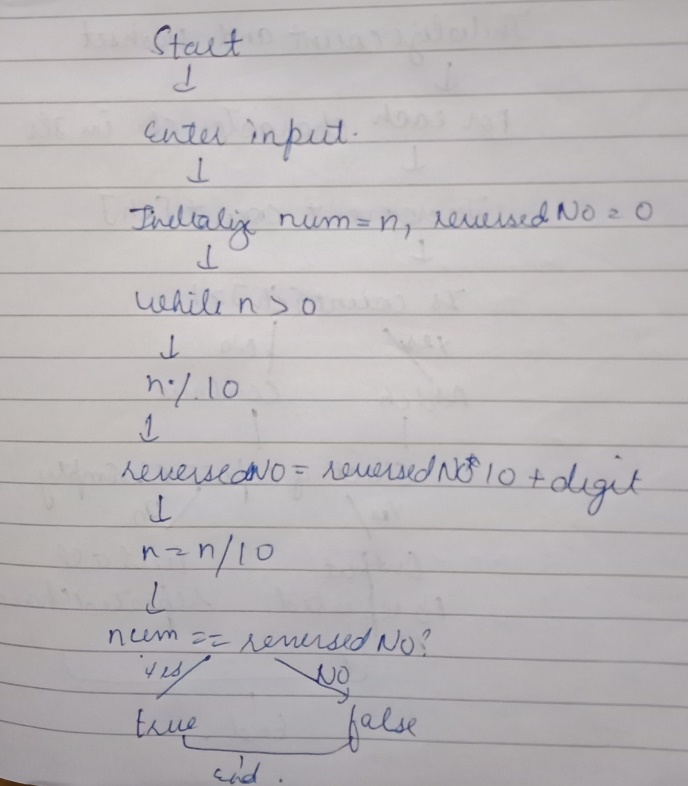
Test Cases:

Input: 121

Output: true

****

**Flowchart-**

****

**Explanation -** Firstly, I have created a method called as pallindrome of boolean return type. Then I stored the original value of n in the variable num. Initialized reversedNo to store the reversed version of the original number. Then inside the while loop which continues as long as n is greater than 0 the last digit of n is extracted using the mod operation. The reversedNo is updated by shifting its digits left and adding the extracted digit. The last digit is removed from n by performing integer division by 10. Then the num is compared with reversedNo. If they are equal, true is returned; otherwise false is returned. In the main method with the help of Scanner class I’ve taken the input from the user and invoked the pallindrome method.

**Time Complexity –**  O (logn)

**Space Complexity-** O (1)

10. Leap Year

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

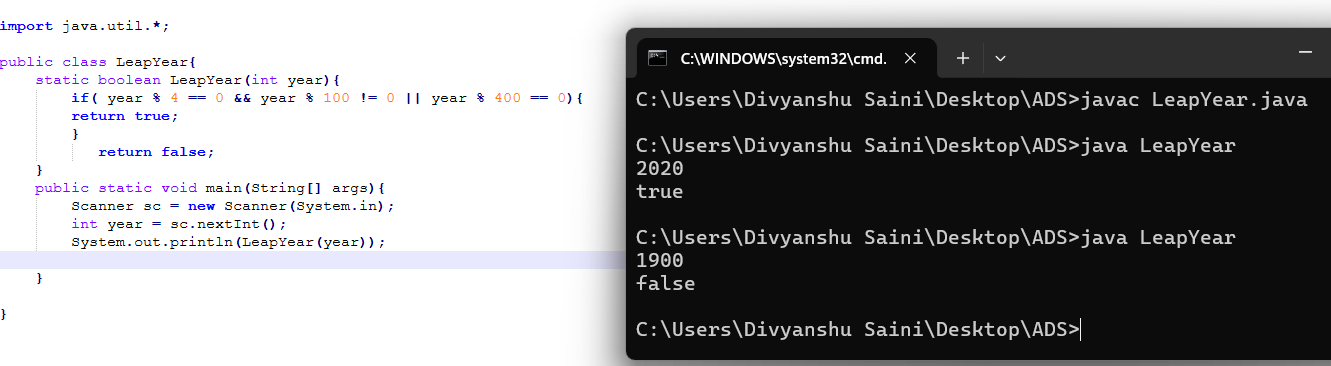
Test Cases:

Input: 2020

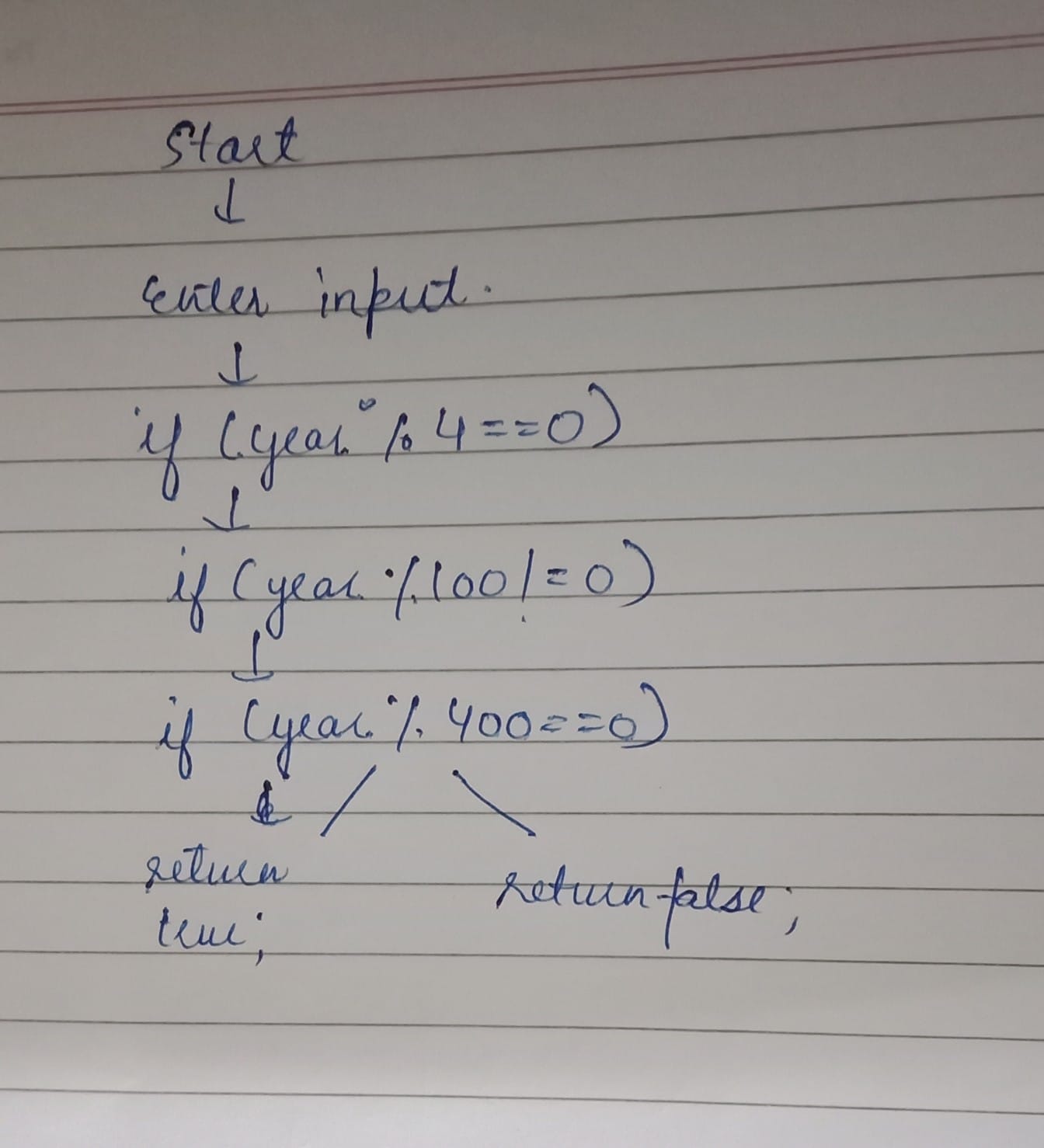
Output: true

Input: 1900

Output: false

****

**Flowchart-**

****

**Explanation -** Firstly, I have created a method called as leapYear of int return type. The if condition checks if the year is divisible by 4, ensures it’s not divisible by 100 and at last checks if the year is divisible by 400. If any of these conditions are satisfied, true is returned, indicating it is a leap year. Otherwise, false is returned. In the main method with the help of Scanner class I’ve taken the input from the user and invoked the leapYear method.

**Time Complexity –**  O (1)

**Space Complexity-** O (1)