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.

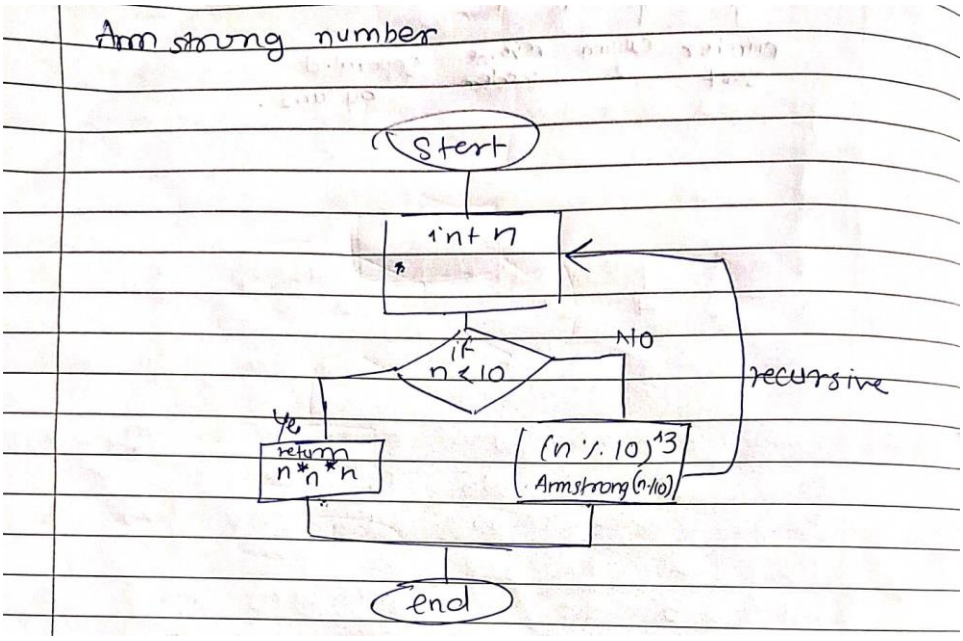
Test Cases:

Input: 153
Output: true
Input: 123
Output: false

Program:

```
//Armstrong number
//number in which the addition of digit raised to no.ofdigits is equal.4
import java.util.Scanner;
class ArmstrongNumber{

	static int armstrong(int n){
		if(n<10) return n*n*n; //if single digit number //base condition
		return(n%10)*(n%10)*(n%10)+armstrong(n/10); //recursive call
	}
	public static void main(String[] args){
		Scanner sc = new Scanner(System.in);
		int num = sc.nextInt();
		int a = armstrong(num); //method call
		if(num == a) //checking
			System.out.println(num+" is an armstrong number");
		else
			System.out.println(num+" not an armstrong number");
	}
}
```

Flowchart:



Explanation:

1. We create a num variable of int type and then assign a user input in it using a scanner;
2. Make a recursive method i.e armstrong and pass num as an argument in it.
3. Then inside that method check if the num is single digit,
    - if it is a single digit number then return n^3 (as we are only considering 3 digit number)
    - else use num%10 to get the digit and then find the cube of that digit and call armstrong method again for the next digit.
4. Call the static method directly inside main and check if the returned value from the method and num is same or not if same it's an Armstrong number.

Output:
153
153 is an armstrong number

Time and space complexity:
Time:

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

Test Cases:

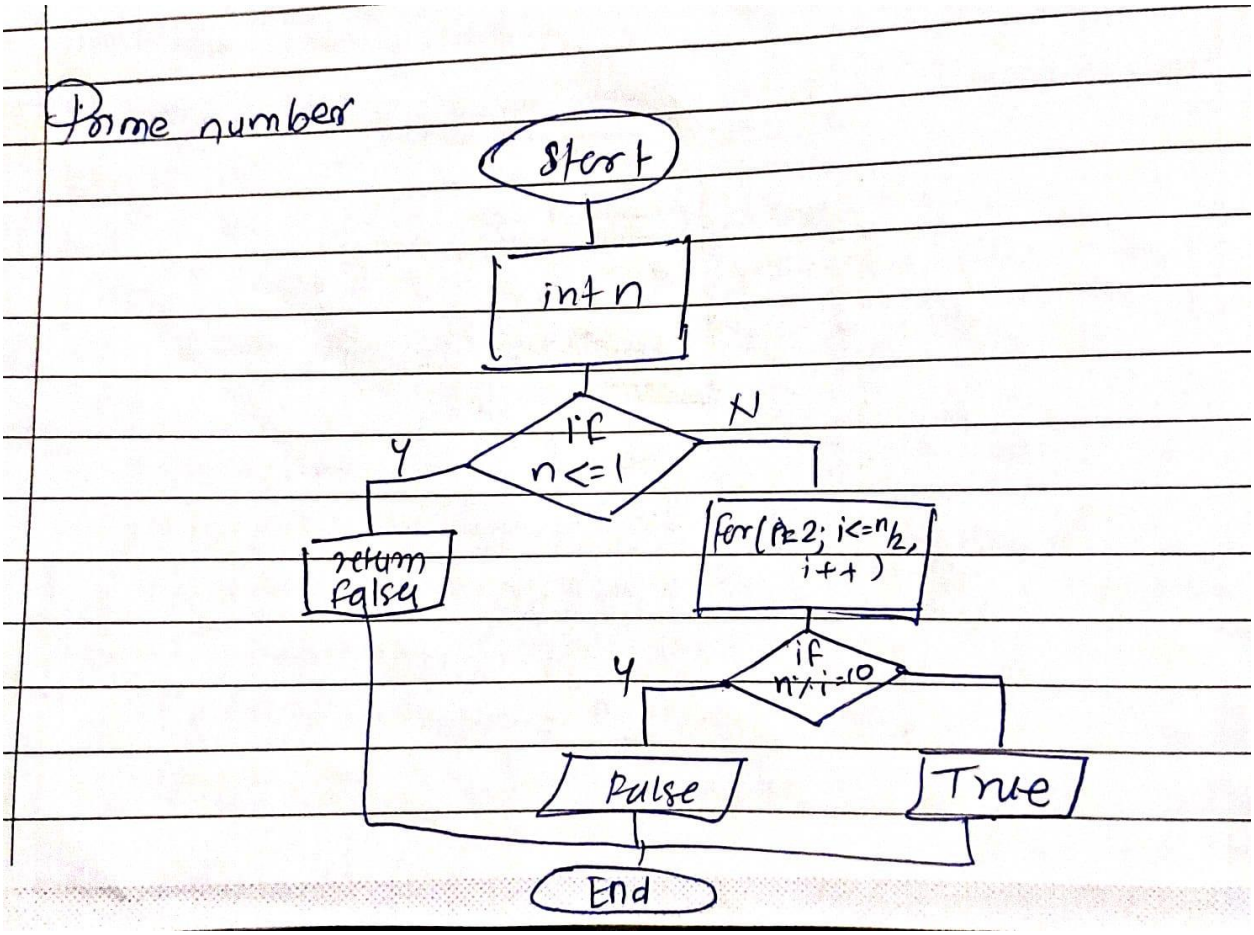
Input: 29
Output: true
Input: 15
Output: false

Program:

```
//PRIME NUMBER
import java.util.Scanner;
class PrimeNumber{
```

```
    static boolean prime(int n){
        if(n<=1) return false; //if num is smaller than 1 return false
        for(int i=2;i<=n/2;i++){ //check starting from 2
            if(n%i == 0)  //if divisible than false
                return false;
        }
        return true;//else return true
    }

    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number: ");
        int num = sc.nextInt();
        boolean a = prime(num);
        if(a) System.out.println(true);
        else System.out.println(false);

    }
}
```

Flowchart:



Explanation:

1. A prime number is a number which is only divisible by 1 or by itself

2. we are making a method called prime to check if a number is prime or not and return values accordingly
3. we are first checking if number is 1 if it is then it should return false.
4. now we defined as for loop for checking if it is divisible by any numbers between 2 and n/2 as division is possible for n/2 of that number.
5. Calling the method and printing the output.

Output:
Enter the number:
29
True
Enter the number:
15
false

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

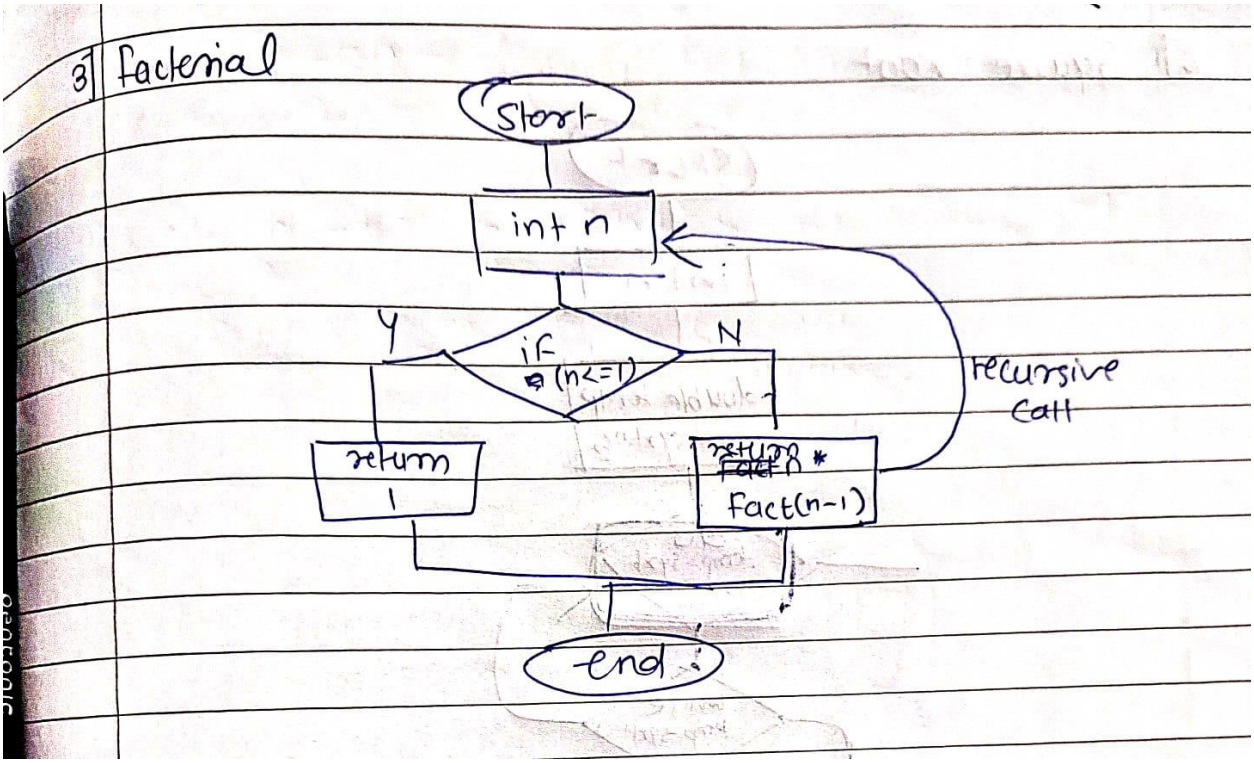
Test Cases:

Input: 5
Output: 120
Input: 0
Output: 1

Program:

```
//Factorial number using recursion
import java.util.Scanner;
class FactorialNumber{

	static int fact(int n){

		if(n<=1)return 1;
		else return n*fact(n-1);
	}

	public static void main(String[] args){
		Scanner sc = new Scanner(System.in);
		System.out.println("Enter the number");
		int num =  sc.nextInt();
		System.out.println(fact(num));
	}

}
```

Flowchart:



Explanation:

1. Factorial of number is the multiplication of number by its previous number so factorial of 5 will be 5*4*3*2*1 i.e 120.
2. In this program we have defined a method called fact.
3. In that method we are using recursive method i.e a method that calls itself.
4. We will check if the number is greater than equal to 1 if true return 1or else do n*fact(n-1) this give recursive call with n-1 so the n goes on decreasing until it becomes 1 which is the termination condition for the recursive method.
5. Then we directly call the fact method in main method as it is a static method.

Output:
Enter the number
5
120
Enter the number
0
1

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

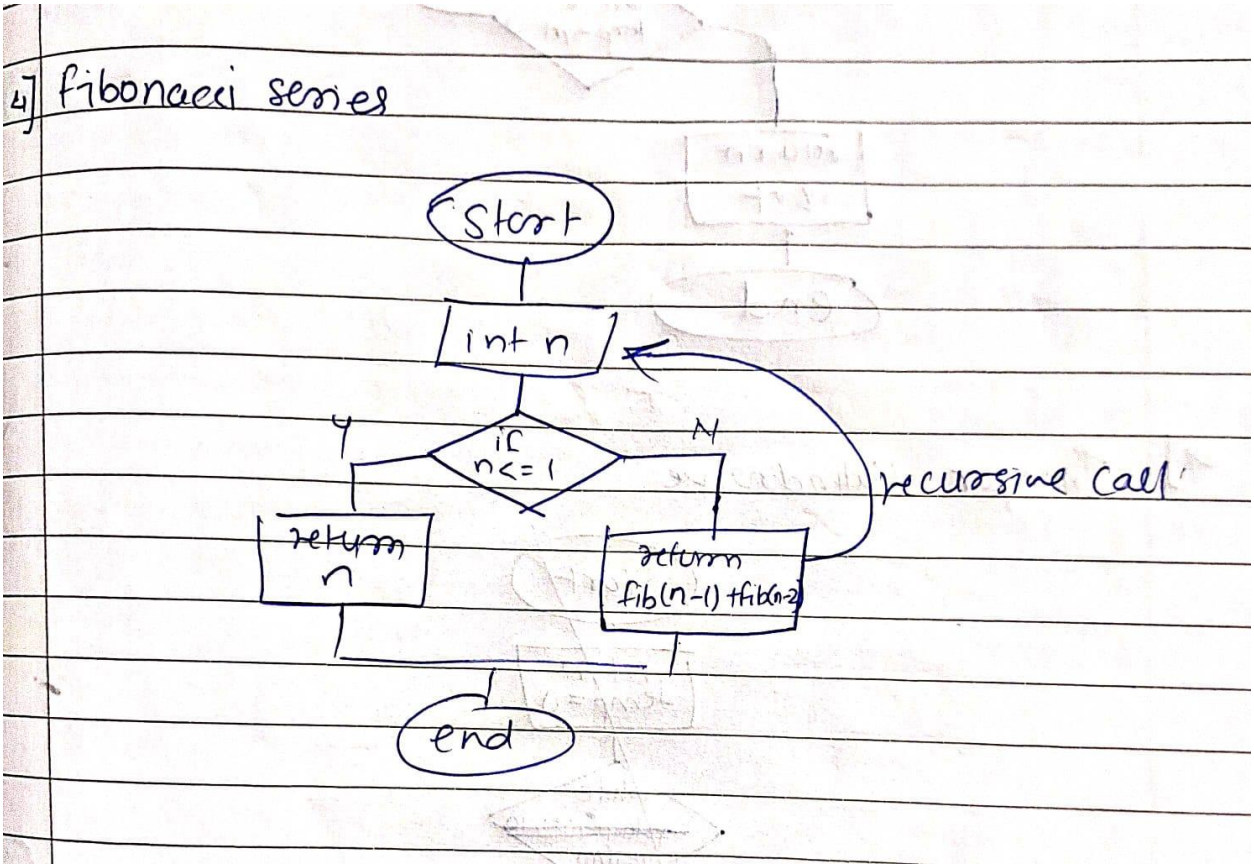
Test Cases:

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

Program:

```
//fibonacci series
import java.util.Scanner;
class FibonacciSeries{

	static int fib(int n){
		if(n<=1){  //checking for +ve and if number is 1
		return n;
	}
	return fib(n-1)+fib(n-2);
	//as we know in fibonacci we add previous number + previous of previous number
	}

	public static void main(String[] args){
		Scanner sc = new Scanner(System.in);
		System.out.println("enter the number: ");
		int n = sc.nextInt(); //input
		for(int i=0;i<n;i++){ //loop for printing values
		System.out.print(fib(i)+" ");
	}
	}
}
```

Flowchart:



Explanation:

1. Fibonacci series is a series for number which are the addition of the previous two number starting from 0 and 1 so the series for 5 is 0 1 1 2 3
2. In this program we have defined a static method called fib
3. We are doing this using recursion method so

Output:
enter the number:
5
0 1 1 2 3
enter the number:
8
0 1 1 2 3 5 8 13

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

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

Test Cases:

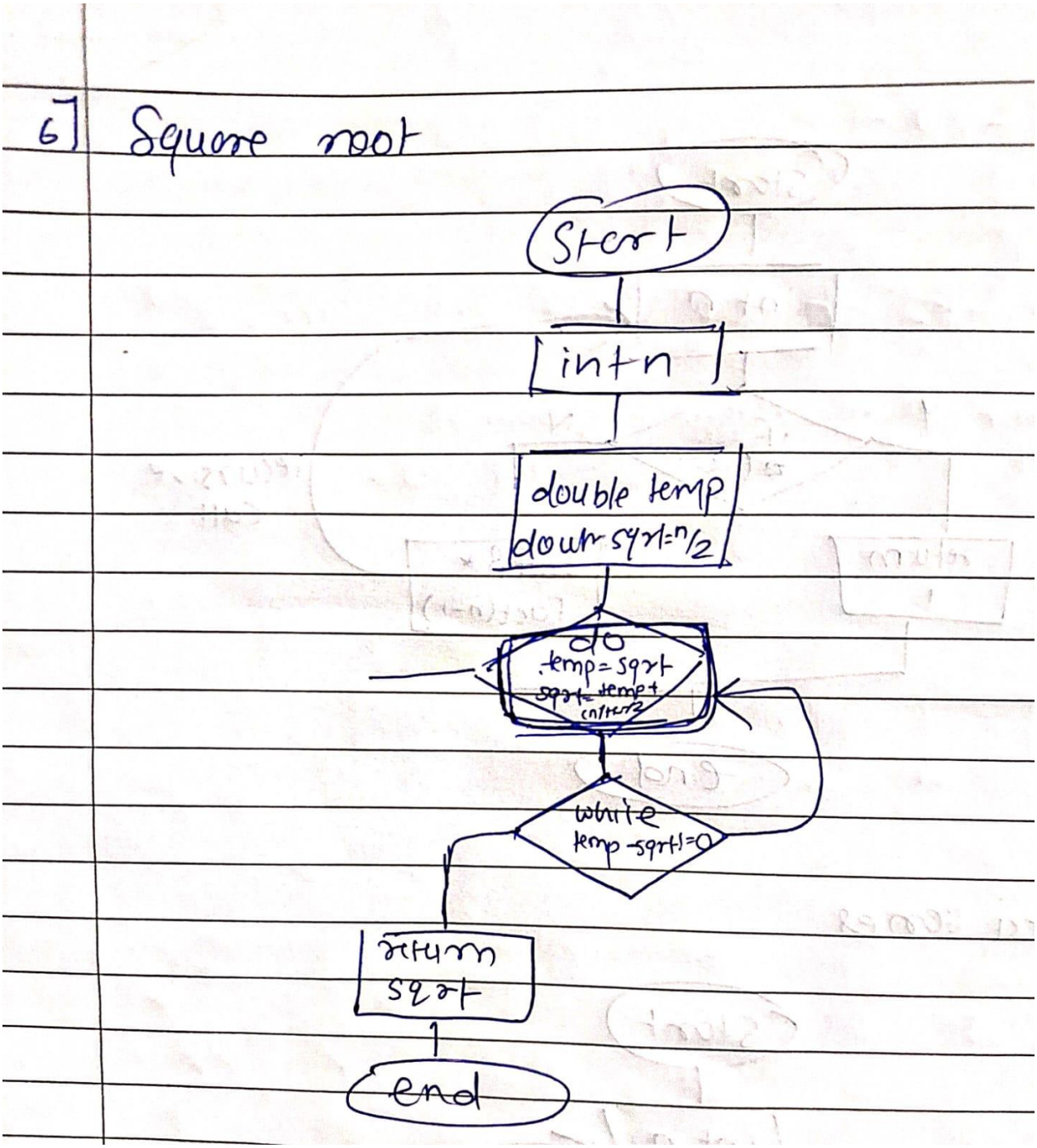
Input: x = 16
Output: 4
Input: x = 27
Output: 5

Program:

```
//Square root
import java.util.Scanner;
class SquareRoot{

	static double sq(int n){
		double temp;
		double sqrt = n/2;
		do
		{
			temp = sqrt;
			sqrt = (temp+(n/temp))/2;
		}
		while((temp-sqrt)!=0);
		return sqrt;
	}

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

```
        Scanner sc = new Scanner(System.in);
        System.out.print("x = ");
        int n = sc.nextInt();
        System.out.println((int)sq(n));
    }
}
```

Flowchart:



Explanation:

1.we are defining a method call sqrt have single parameter n having a temp variable as double and sqrt is half of n.
2. then we are using a do while loop to find the squareroot of n by using formula
$sqrt_{n+1}=(sqrt_n+(num/sqrt_n))/2.0$
3. calling method sqrt in main method for execution.

Output:
x = 16
4
x = 27
5

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:

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:

```
public class FirstNonRepeatedCharacter {

    public static Character findFirstNonRepeatedChar(String str) {
        int[] charCount = new int[300];

        // Count frequency of each character in the string
        for (int i = 0; i < str.length(); i++) {
            charCount[str.charAt(i)]++;
        }

        // Find the first character with frequency 1
        for (int i = 0; i < str.length(); i++) {
            if (charCount[str.charAt(i)] == 1) {
                return str.charAt(i);
            }
        }
```

```
        return null; // If no non-repeated character is found
    }

    public static void main(String[] args) {
        // Test cases
        String input1 = "stress";
        String input2 = "aabbcc";

        System.out.println("Input:" + input1);
        System.out.println("Output: "+ findFirstNonRepeatedChar(input1));

        System.out.println("Input: " + input2);
        System.out.println("Output: " + findFirstNonRepeatedChar(input2));
    }
}
```

Output:
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:

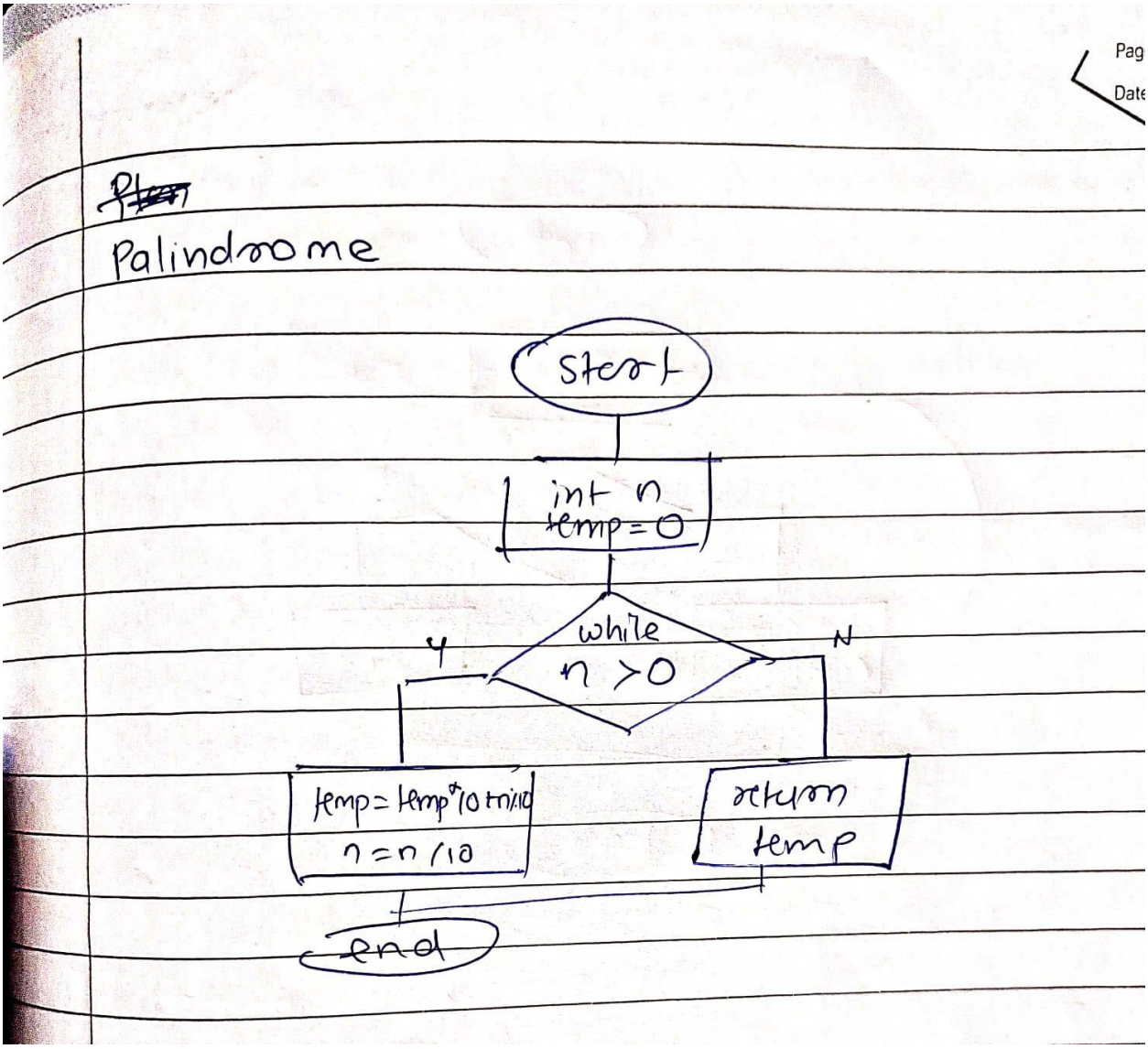
Input: 121
Output: true
Input: -121
Output: false

Program:

```
//Integer palindrome
import java.util.Scanner;
class Palindrome{

        static int pal(int n){
                int temp = 0;
                while(n>0){
                        temp = temp * 10 + n % 10;
                        n = n/10;
                }
                return temp;
        }

        public static void main(String[] args){
                Scanner sc = new Scanner(System.in);
                System.out.print("x = ");
                int n = sc.nextInt();
```

```
                int temp = pal(n);
                if(temp == n){
                        System.out.println(true);
                }else
                        System.out.println(false);
        }
}
```

flowchart:



Explanation:

1. In this program we have created a separate static method called pal to reverse the number series
2. We have taken a temp variable to store the reverse value
3. We have used while loop to reverse the integer with taking each digit.

Output:

Complexity: time:O(n)

Space:

10. Leap Year

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

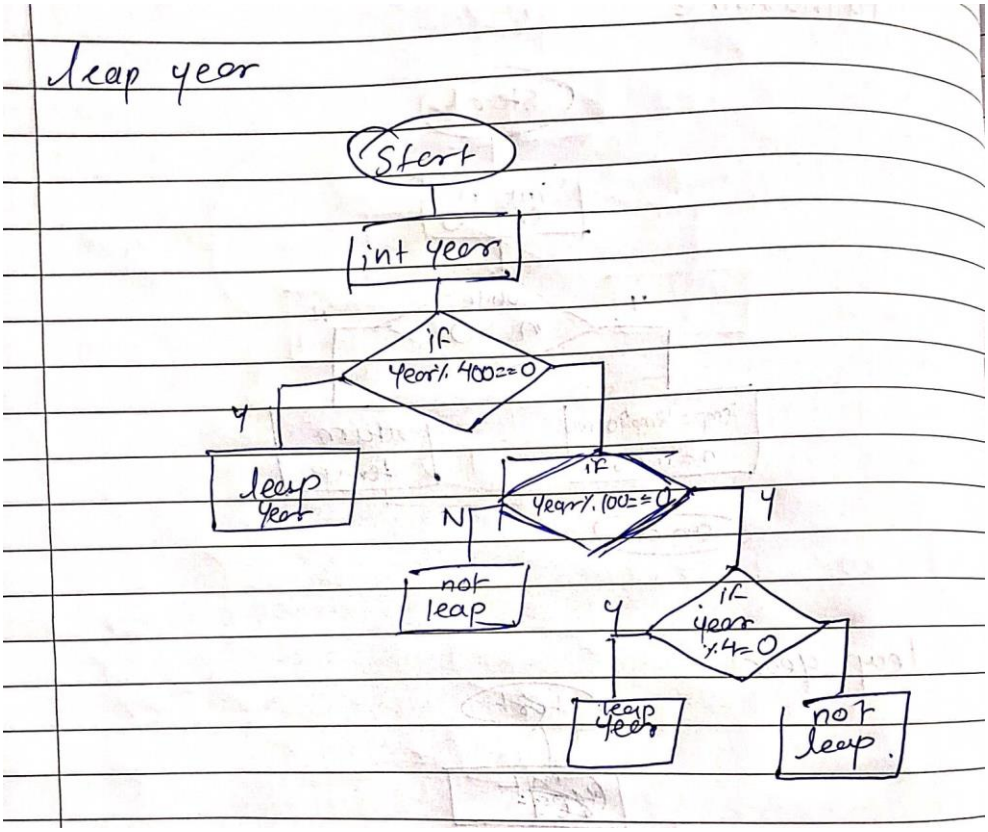
Test Cases:

Input: 2020
Output: true
Input: 1900
Output: false

Program:

```
//leap year
import java.util.Scanner;
class LeapYear{

        public static void main(String[] args){
                Scanner sc = new Scanner(System.in);
                System.out.print("year = ");
                int year = sc.nextInt();

                if (year % 400 == 0) {
      System.out.println(year + " is a Leap Year.");
    }

    // Checking the second condition
    else if (year % 100 == 0) {
      System.out.println(year + " is not a Leap Year.");
    }

    // checking the third condition
    else if (year % 4 == 0) {
      System.out.println(year + " is a Leap Year.");
    }
    else {
      System.out.println(year + " is not a Leap Year.");
    }
         }

}
```

Flowchart:



Output:
year = 2020
2020 is a Leap Year.

year = 1900
1900 is not a Leap Year.