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.

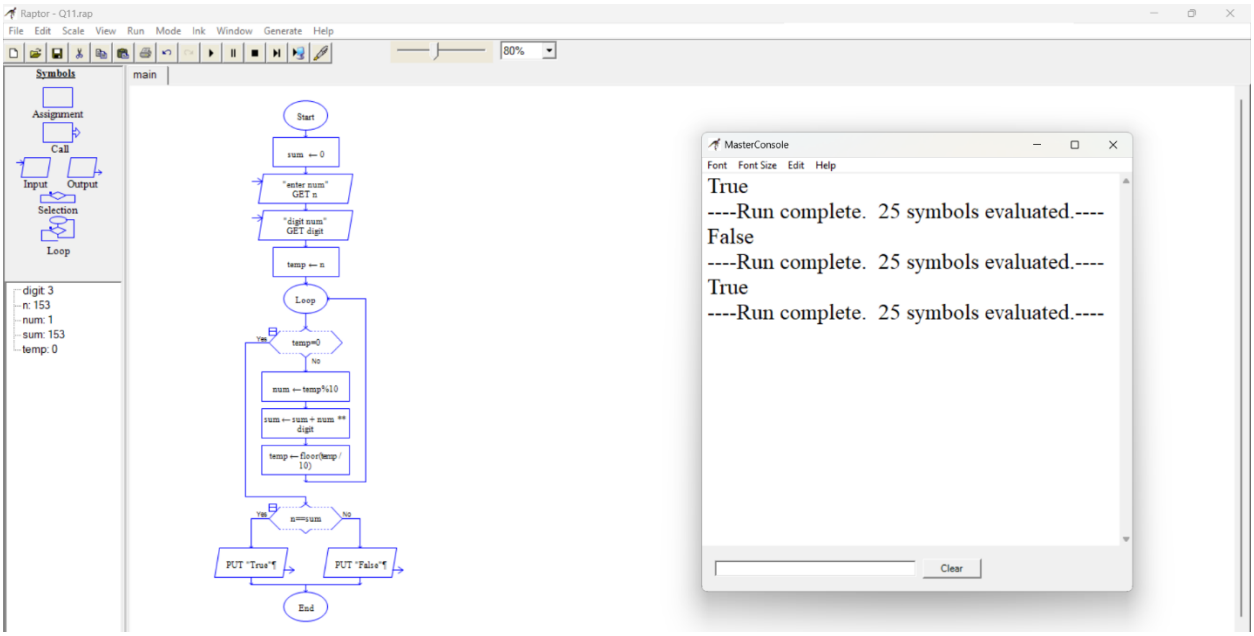
-Program
-Flow chart
-Explanation
-Output
-Time and Space complexity

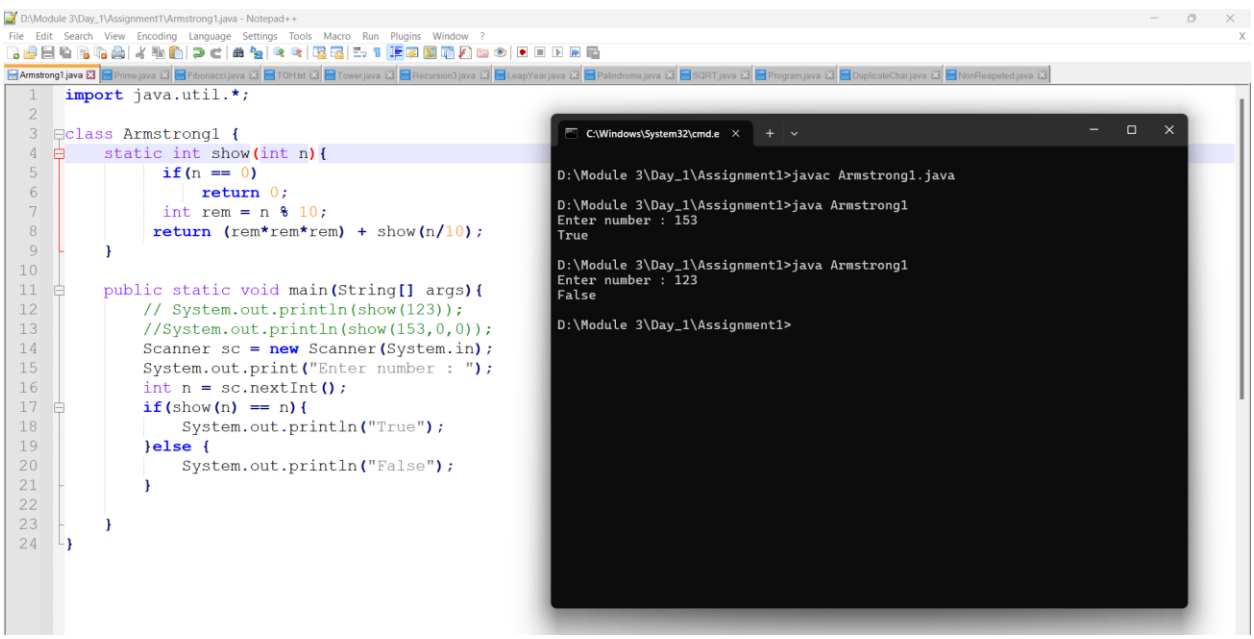
1. Armstrong Number
Problem: Write a Java program to check if a given number is an Armstrong number.

Test Cases:

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

```
import java.util.*;

class Armstrong1 {
	static int show(int n){
		if(n == 0)
			return 0;
		int rem = n % 10;
		return (rem*rem*rem) + show(n/10);
	}

	public static void main(String[] args){
		Scanner sc = new Scanner(System.in);
		System.out.print("Enter number : ");
		int n = sc.nextInt();
		if(show(n) == n){
			System.out.println("True");
		}else {
			System.out.println("False");
		}

	}
}
```





Explaination:-

1) When we enter a particular no. then we have to check that wheather that no is Armstrong or not just by adding each digits cube.
2) To find it I have use if condition in that it will check if the no is zero then it will terminate the program else it will continue.

3) I have taken remainder of that no so that we can get a one particular digit from that no itself to calculate it.
4) And calculate the sum of cube .

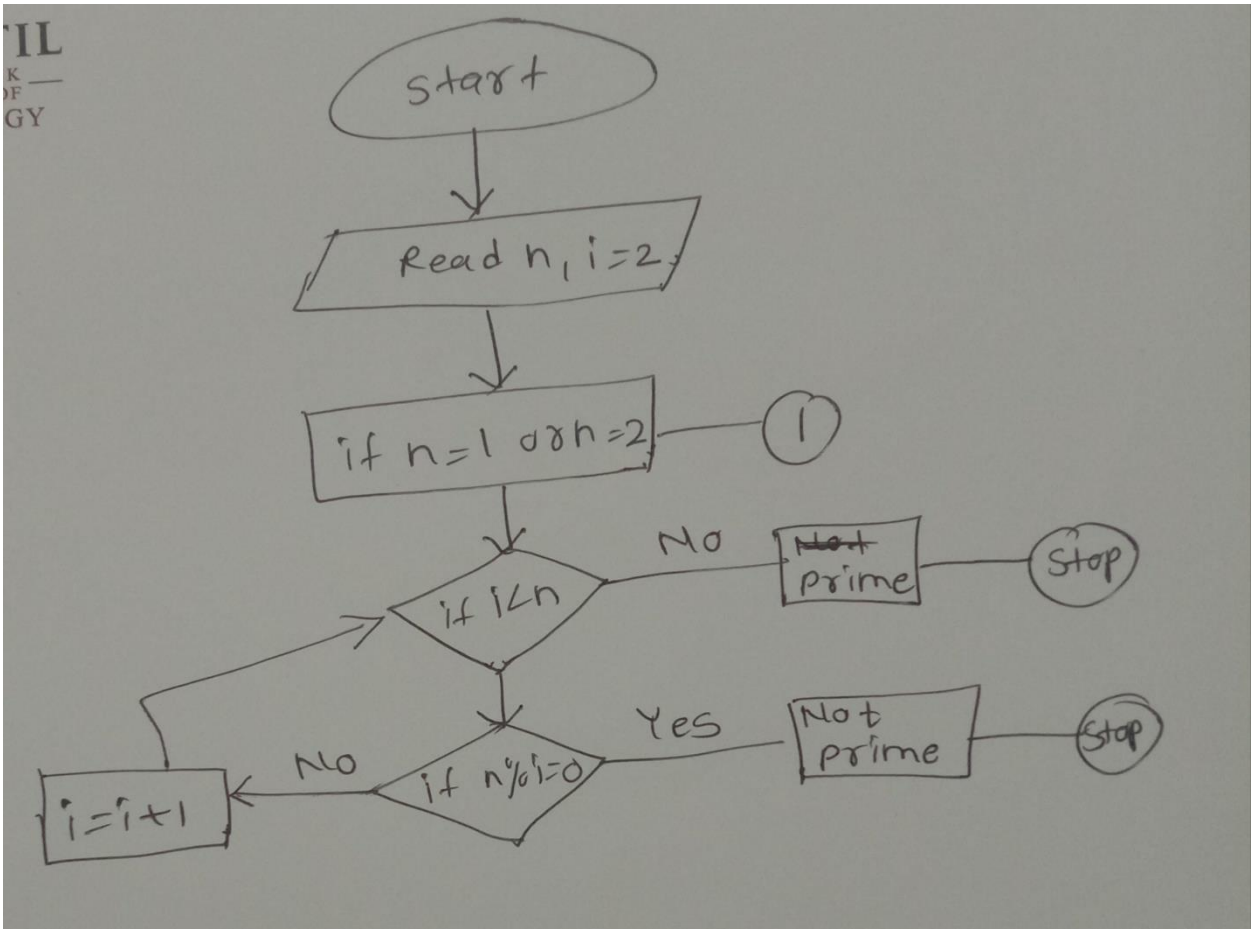
Time and space complexity is O(log10(n)).

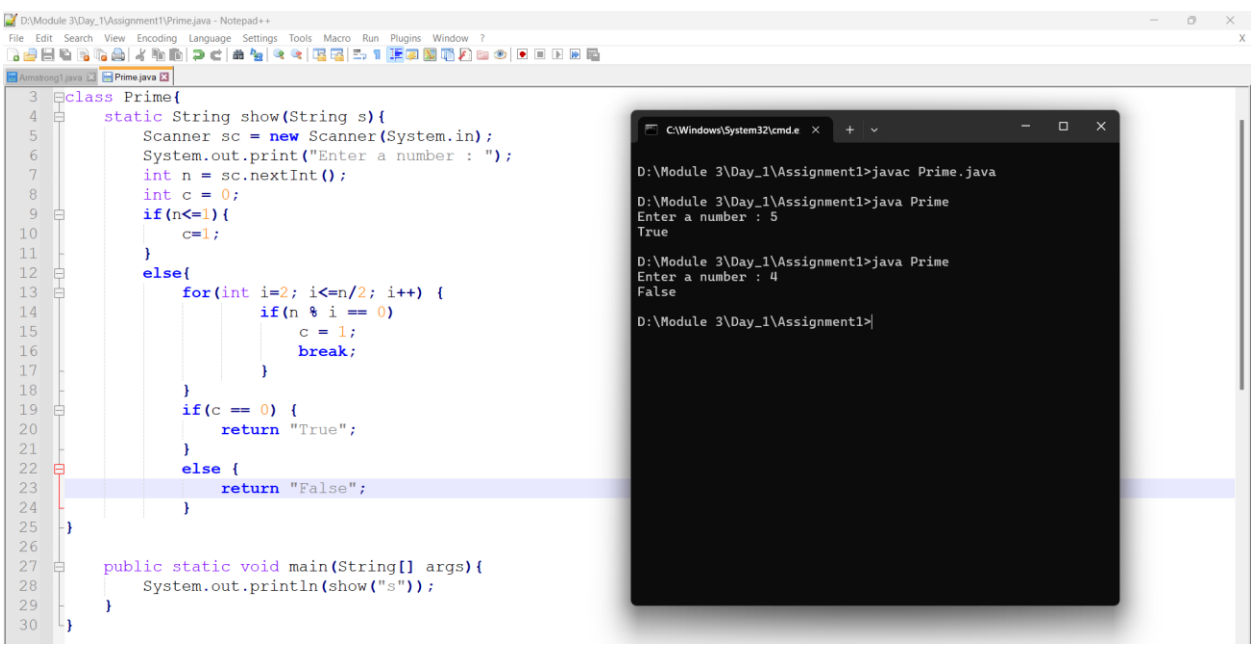
2. Prime Number
Problem: Write a Java program to check if a given number is prime.

Test Cases:

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

```
import java.util.Scanner;

class Prime{
	static String show(String s){
		Scanner sc = new Scanner(System.in);
		System.out.print("Enter a number : ");
		int n = sc.nextInt();
		int c = 0;
		if(n<=1){
			c=1;
		}
		else{
			for(int i=2; i<=n/2; i++) {
				if(n % i == 0)
					c = 1;
					break;
				}
			}
			if(c == 0) {
				return "True";
			}
			else {
				return "False";
			}
}

	public static void main(String[] args){
		System.out.println(show("s"));
	}
}
```





Explaination:

1) I have use if condition to check wheather that no. is less than or equal to 1, if it is then make a count to 1;

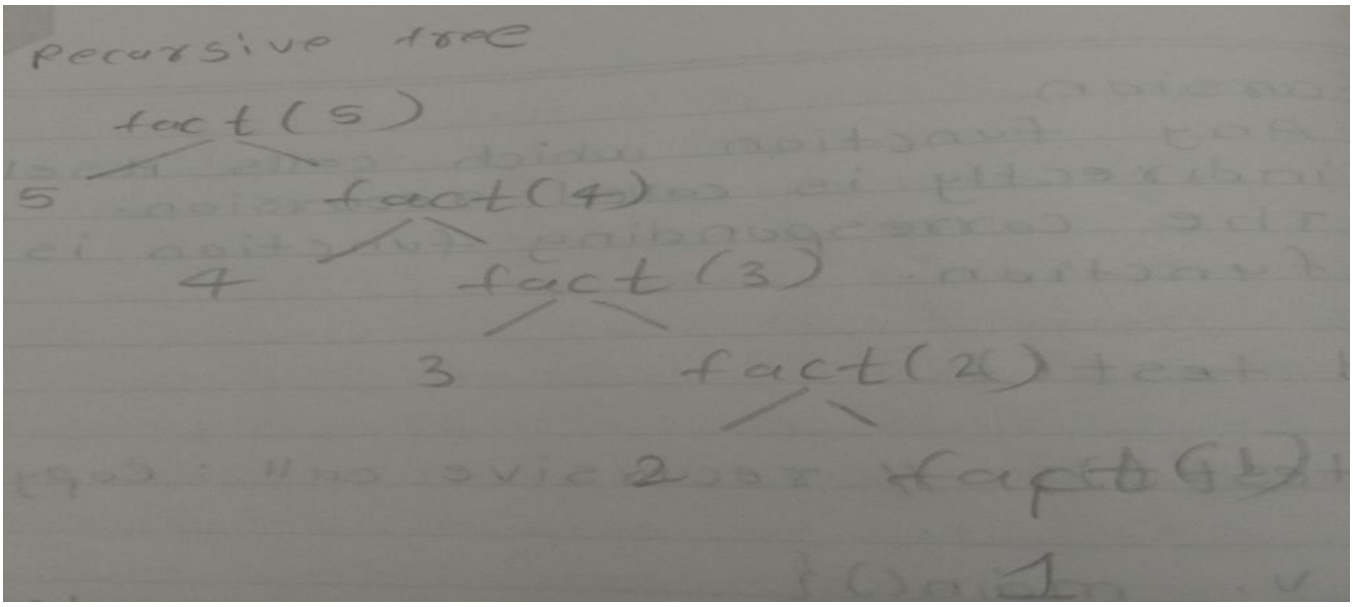
2) Then I use for loop in else condition to check wheather that no is completely divided bt that no.s half or not if it is then it is not a prime no if it is not then only it is a prime no.
3) Then user if condition to check if that count is 0 means not divisible by any no. except 1 and itself then that no. is a prime no.
4) Time complexity is O(n). Space complexity is O(1)

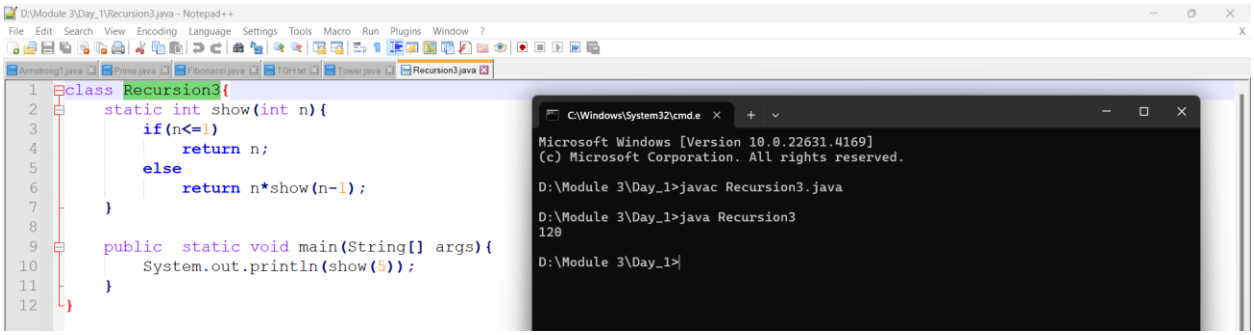
3. Factorial
Problem: Write a Java program to compute the factorial of a given number.

Test Cases:

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

```
class Recursion3{
	static int show(int n){
		if(n<=1)
			return n;
		else
			return n*show(n-1);
	}

	public  static void main(String[] args){
		System.out.println(show(5));
	}}
```





Explanation:-

1) I have use recursion method in it.
2) First I check wheather that no is less that or equal to 1 if it is then it will simply print that no.
3) Else it will multiply that no. with no. which are less than that till 1.
4) Means 5*show(4) 4*show(3) 3*show(2) 2*show(1) 1 like this it will iterate.
5) In last it will multiply 5*4*3*2*1 = 120
6) Time complexity = O(n) space complexity = O(n)

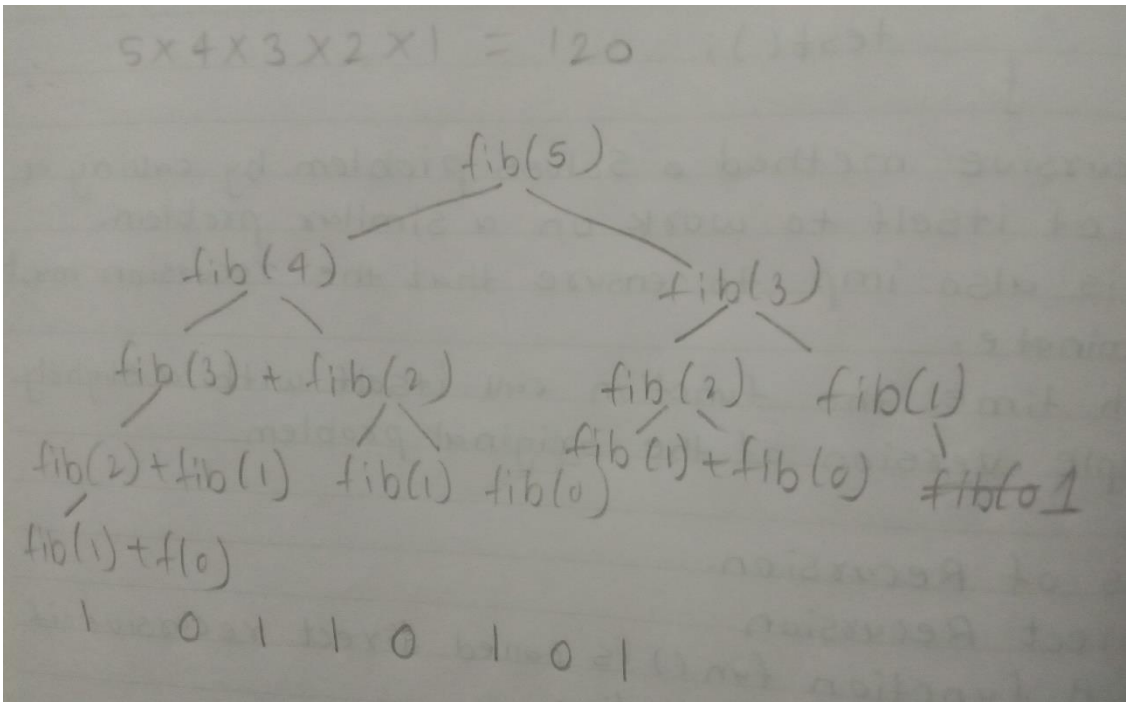
4. Fibonacci Series

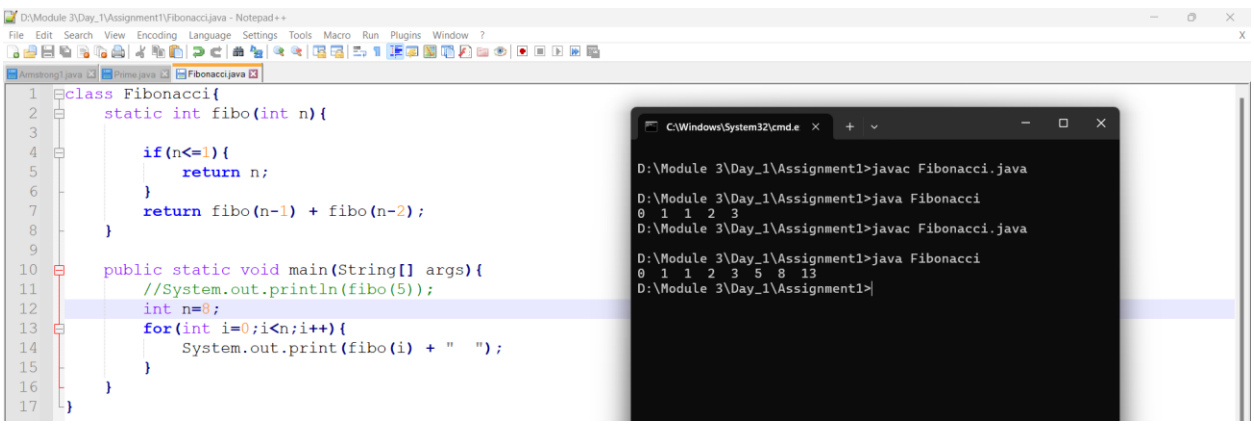
Problem: Write a Java program to print the first n numbers in the Fibonacci series.

Test Cases:

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

```
class Fibonacci{
        static int fibo(int n){

                if(n<=1){
                        return n;
                }
                return fibo(n-1) + fibo(n-2);
        }

        public static void main(String[] args){
                //System.out.println(fibo(5));
                int n=8;
                for(int i=0;i<n;i++){
                        System.out.print(fibo(i) + "  ");
                }
        }
}
```





Explaination:-

1) First I check wheather that no. less that 1 or equal to it if it is then simply print n it is our base condition
2) Then add fibo(n-1) to fibo(n-2) because every time we want a fibonacchi series we have to add previous two no to get the next no.
3) To get previous 1 no I have taken fibo(n-1) and for previous 2nd no I have taken fibo(n-2).
4) Time complexity = O(2^n) space complexity = O(n)

5. Find GCD

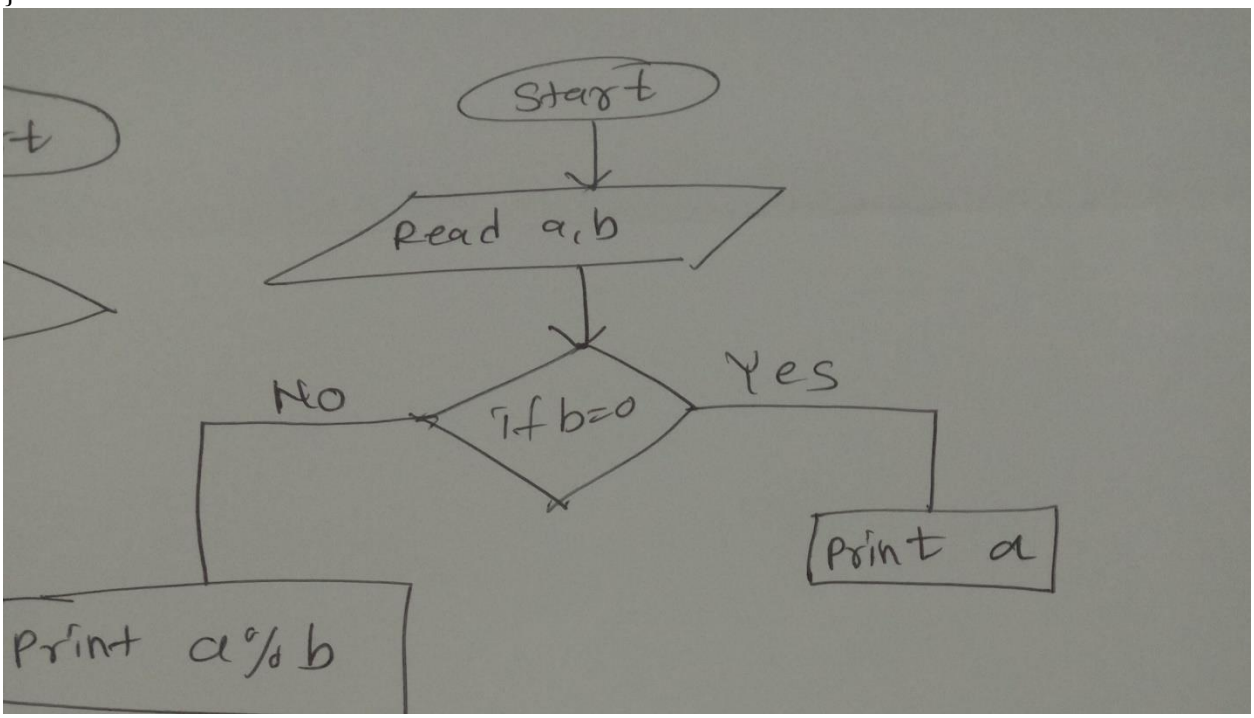
Problem: Write a Java program to find the Greatest Common Divisor (GCD) of two numbers.

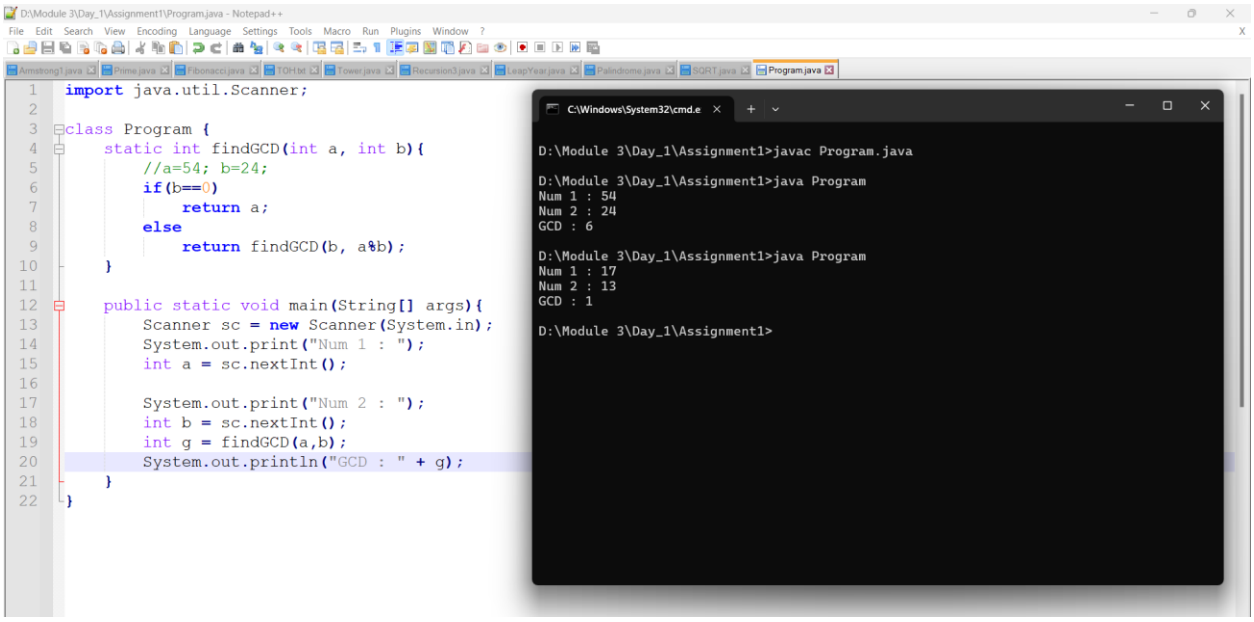
Test Cases:

Input: a = 54, b = 24
Output: 6
Input: a = 17, b = 13

Output: 1

```
import java.util.Scanner;

class Program {
	static int findGCD(int a, int b){
		if(b==0)
			return a;
		else
			return findGCD(b, a%b);
	}

	public static void main(String[] args){
		Scanner sc = new Scanner(System.in);
		System.out.print("Num 1 : ");
		int a = sc.nextInt();

		System.out.print("Num 2 : ");
		int b = sc.nextInt();
		int g = findGCD(a,b);
		System.out.println("GCD : " + g);
	}
}
```





Explaination:-

1) Take two no a,b
2) First check wheather that b is equal to zero or not
3) If it is then simply print a as gcd
4) If not then find remainder of two no by a%b
5) Get gcd

6. Find Square Root

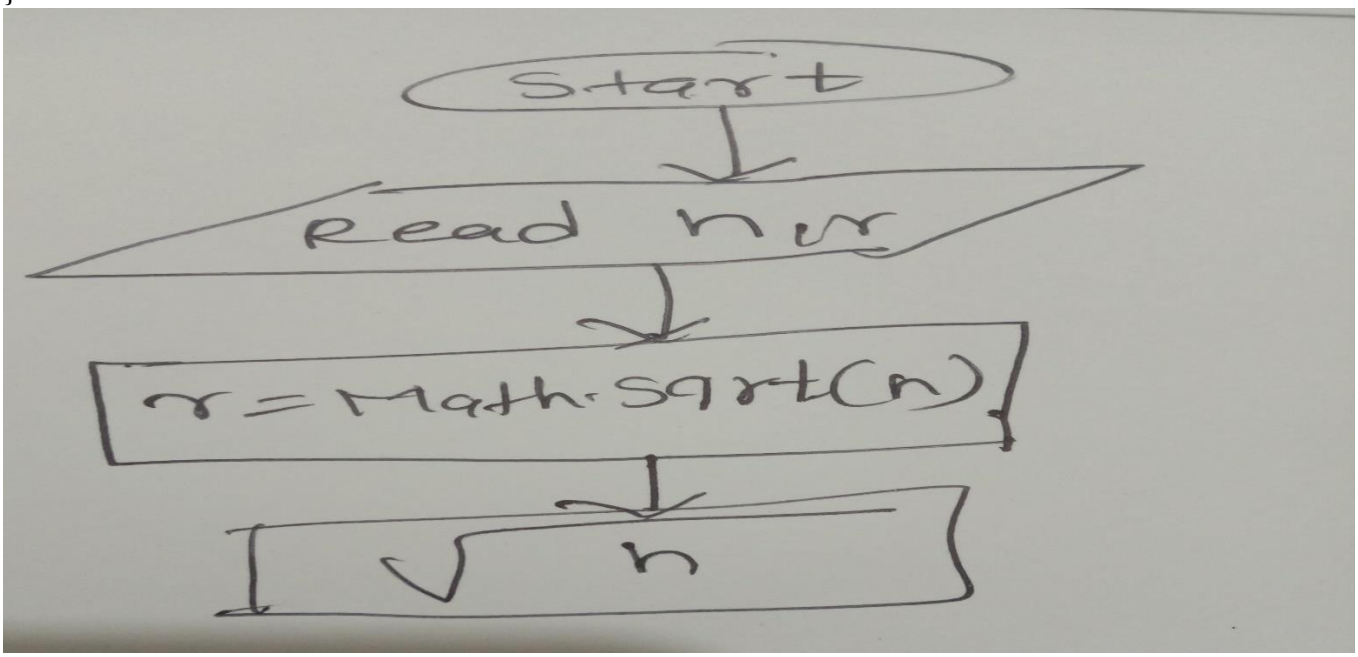
Problem: Write a Java program to find the square root of a given number (using integer approximation).

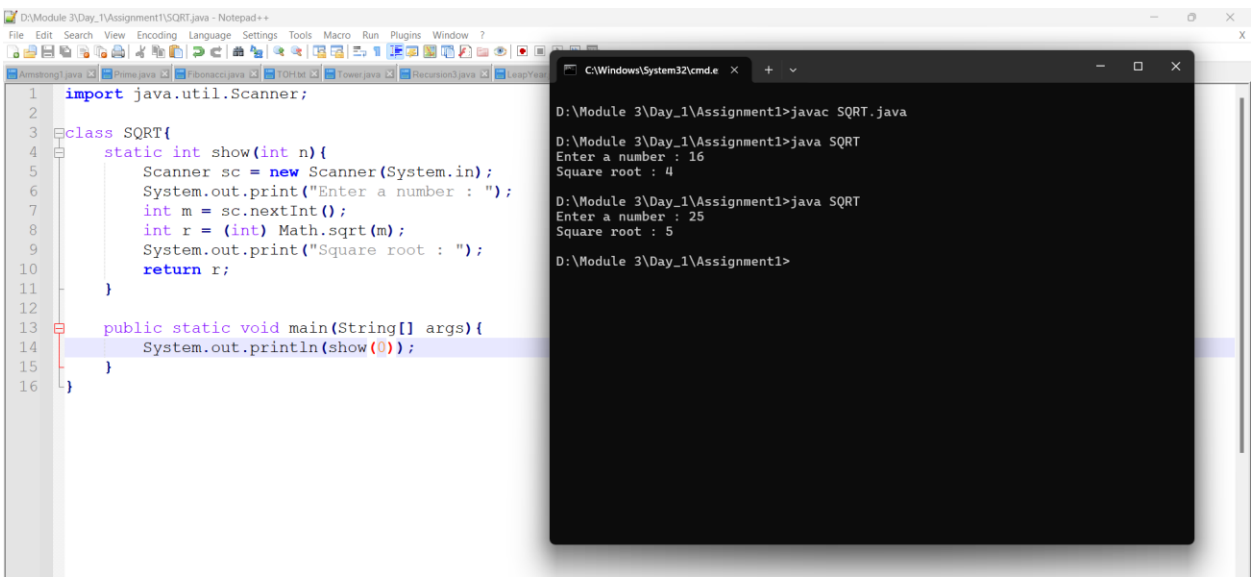
Test Cases:

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

```
import java.util.Scanner;

class SQRT{
	static int show(int n){
		Scanner sc = new Scanner(System.in);
		System.out.print("Enter a number : ");
		int m = sc.nextInt();
		int r = (int) Math.sqrt(m);
		System.out.print("Square root : ");
		return r;
	}

	public static void main(String[] args){
		System.out.println(show(0));
```

```
        }
}
```





Explaination:-

1) First take two a input
2) Then use Math.sqrt() operation on it to find square root Math.sqrt() accept only double values that why I have use int as explicit type casting to get square root as integer no.
3) In this way I get square root of that particular no
4) Time complexity = O(1) and space compexity = O(1).

7. Find Repeated Characters in a String

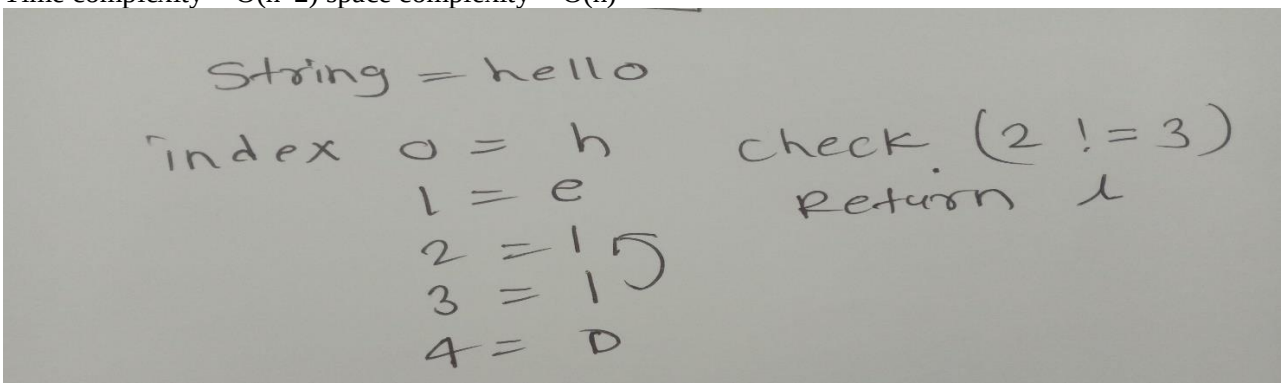
Problem: Write a Java program to find all repeated characters in a string.

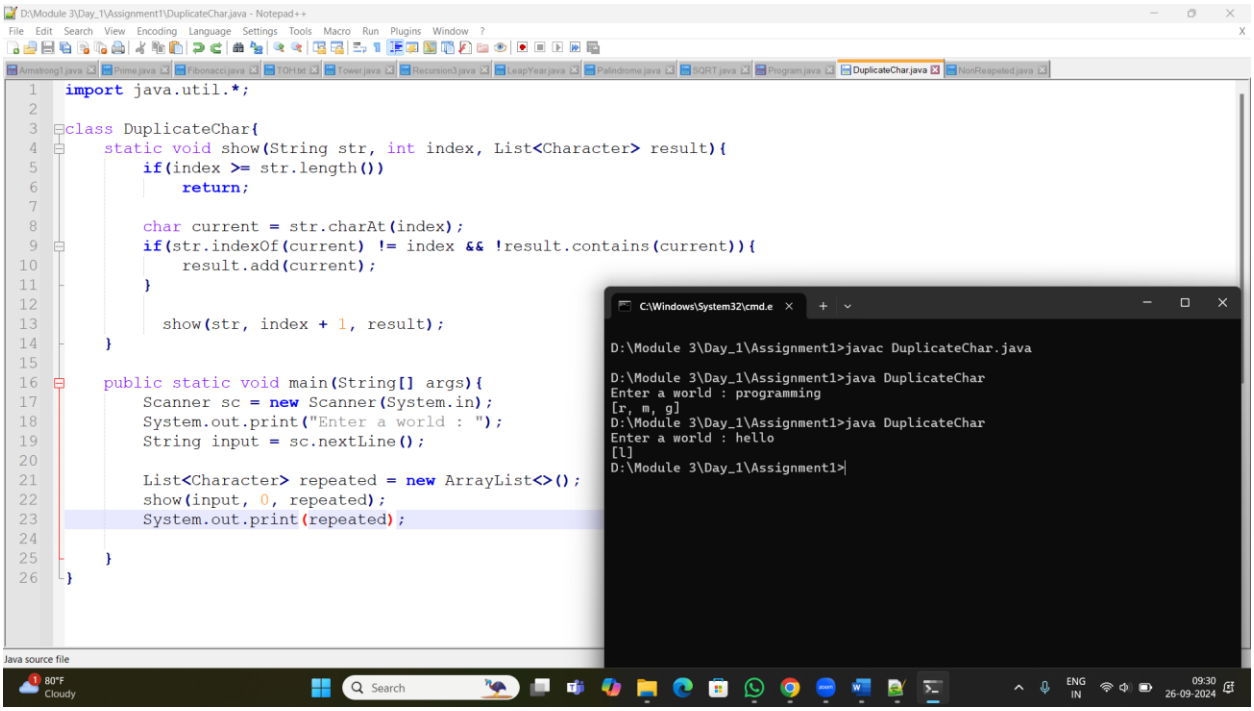
Test Cases:

Input: "programming"
Output: ['r', 'g', 'm']
Input: "hello"
Output: ['l']

```
import java.util.*;

class DuplicateChar{
	static void show(String str, int index, List<Character> result){
		if(index >= str.length())
			return;

		char current = str.charAt(index);
		if(str.indexOf(current) != index && !result.contains(current)){
			result.add(current);
		}

		  show(str, index + 1, result);
	}

	public static void main(String[] args){
		Scanner sc = new Scanner(System.in);
		System.out.print("Enter a world : ");
		String input = sc.nextLine();

		List<Character> repeated = new ArrayList<>();
		show(input, 0, repeated);
		System.out.print(repeated);

	}
}
```

Explaination:-

1) Check whether that index is greater than string length or not if it is , will terminate the program
2) Taken char to iterate through ech charater of that string so I use str.charAt(index) so that it will go to each and every character to check that character is duplicate or not.
3) Then check If that character index is not equal to that particular character and is that character not contain in that particular list, it will add to that list
4) Then we get duplicate character from that string.
5) Time complexity = O(n^2) space complexity = O(n)
6) 



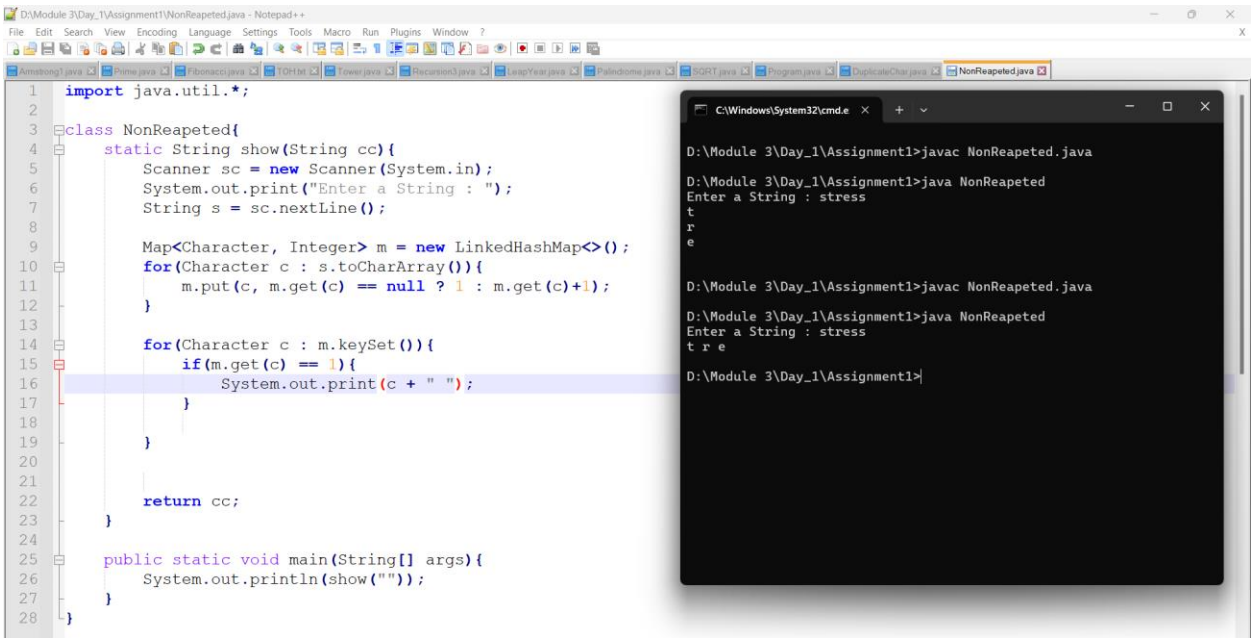
8. First Non-Repeated Character
Problem: Write a Java program to find the first non-repeated character in a string.

Test Cases:

Input: "stress"
Output: 't'
Input: "aabbcc"
Output: null

```
import java.util.*;

class NonReapeted{
	static String show(String cc){
		Scanner sc = new Scanner(System.in);
		System.out.print("Enter a String : ");
		String s = sc.nextLine();

		Map<Character, Integer> m = new LinkedHashMap<>();
		for(Character c : s.toCharArray()){
			m.put(c, m.get(c) == null ? 1 : m.get(c)+1);
		}

		for(Character c : m.keySet()){
			if(m.get(c) == 1){
				System.out.print(c + " ");
```

```
                        }

                }


                return cc;
        }

        public static void main(String[] args){
                System.out.println(show(""));
        }
}
```



Explaination:-

1) Use map to check uniqe character
2) Use for each loop to iterate over all characters from string
3) In that m.put() means to add in that use a condition which check wheather that character is null or not means is that character appears befor or not if it is then I increase that characters key by 1 if not then that key will be 1 only
4) Then check for those characters who are having key 1
5) And print those to get unique characters.
6) Time complexity = O(n^2) 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
Input: -121

Output: false

```
import java.util.Scanner;

class Palindrome{
	static String show(String s){
		Scanner sc = new Scanner(System.in);
		System.out.print("Enter a number : ");
		int n = sc.nextInt();
		int temp = n;
		int p = 0;
		while(n>0){
			int digit = n % 10;
			p = (p * 10) + digit ;
			n /= 10;
		}

		if(p == temp)
			return "True";
		else
			return "False";
	}

	public static void main(String[] args){
		System.out.println(show("s"));
	}
}
```

Explaination:-

1) Did some mathematical steps to get the reverse no which I have written below
2) Then check wheather that entered no is equal to reversed no or not if it is print true else print false;
3) Time complexity = O(n) space complexity = O(1)

digit = 121 % 10

= 1

p = 0 x 10 + 1

= 1

n = 121 / 10

= 12

digit = 12 % 10

= 2

p = 1 x 10 + 2

= 12

n = 12 / 10

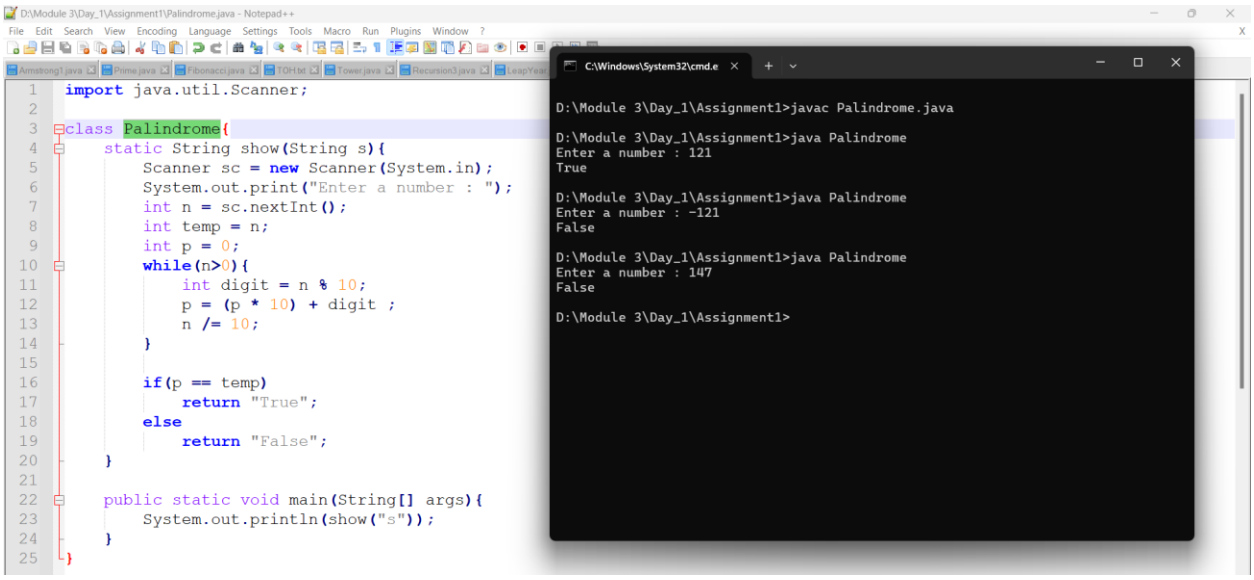
= 1

digit = 1 % 10

= 1

p = 12 x 10 + 1

= 121



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.Scanner;

class LeapYear{
	static String show(String s){
		Scanner sc = new Scanner(System.in);
		System.out.print("Enter a year : ");
		int n = sc.nextInt();
		if(n%4 == 0 && (n%100 != 0 || n%400 == 0))
			return "True";
		else
			return "False";
	}

	public static void main(String[] args){
		System.out.println(show("s"));
	}
}
```

Explaination:-

1) Use if condition in that if that year is completely divisible by 4 and divisible by 400 but not by 100.
2) Then only it will be a leap year else it will not be a leap year.
3) Time complexity = O(1) space complexity = O(1)

D:\Module 3\Day_1\Assignment1\LeapYear.java - Notepad++

```java
import java.util.Scanner;

class LeapYear{
    static String show(String s){
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a year : ");
        int n = sc.nextInt();
        if(n%4 == 0 && (n%100 != 0 || n%400 == 0))
            return "True";
        else
            return "False";
    }

    public static void main(String[] args){
        System.out.println(show("s"));
    }
}
```

C:\Windows\System32\cmd.e

```
D:\Module 3\Day_1\Assignment1>javac LeapYear.java

D:\Module 3\Day_1\Assignment1>java LeapYear
Enter a year : 2020
True

D:\Module 3\Day_1\Assignment1>java LeapYear
Enter a year : 1900
False

D:\Module 3\Day_1\Assignment1>
```