

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

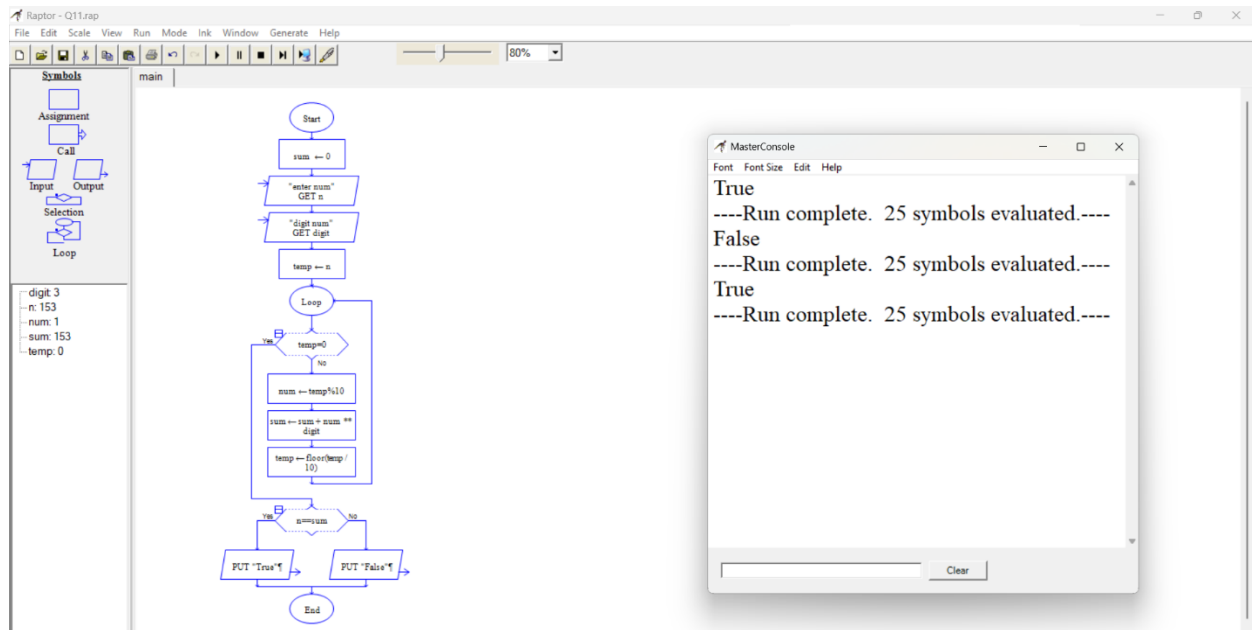
Input: 123

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");
        }
    }
}
```



```
1 import java.util.*;
2
3 class Armstrong1 {
4     static int show(int n){
5         if(n == 0)
6             return 0;
7         int rem = n % 10;
8         return (rem*rem*rem) + show(n/10);
9     }
10
11     public static void main(String[] args){
12         // System.out.println(show(123));
13         //System.out.println(show(153,0,0));
14         Scanner sc = new Scanner(System.in);
15         System.out.print("Enter number : ");
16         int n = sc.nextInt();
17         if(show(n) == n){
18             System.out.println("True");
19         }else {
20             System.out.println("False");
21         }
22     }
23 }
24
```

The terminal window shows the execution of the program:

```
D:\Module 3\Day_1\Assignment1>javac Armstrong1.java
D:\Module 3\Day_1\Assignment1>java Armstrong1
Enter number : 153
True
D:\Module 3\Day_1\Assignment1>java Armstrong1
Enter number : 123
False
D:\Module 3\Day_1\Assignment1>
```

Explanation:-

- 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 $O(\log_{10}(n))$.

2. Prime Number

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

Test Cases:

Input: 29

Output: true

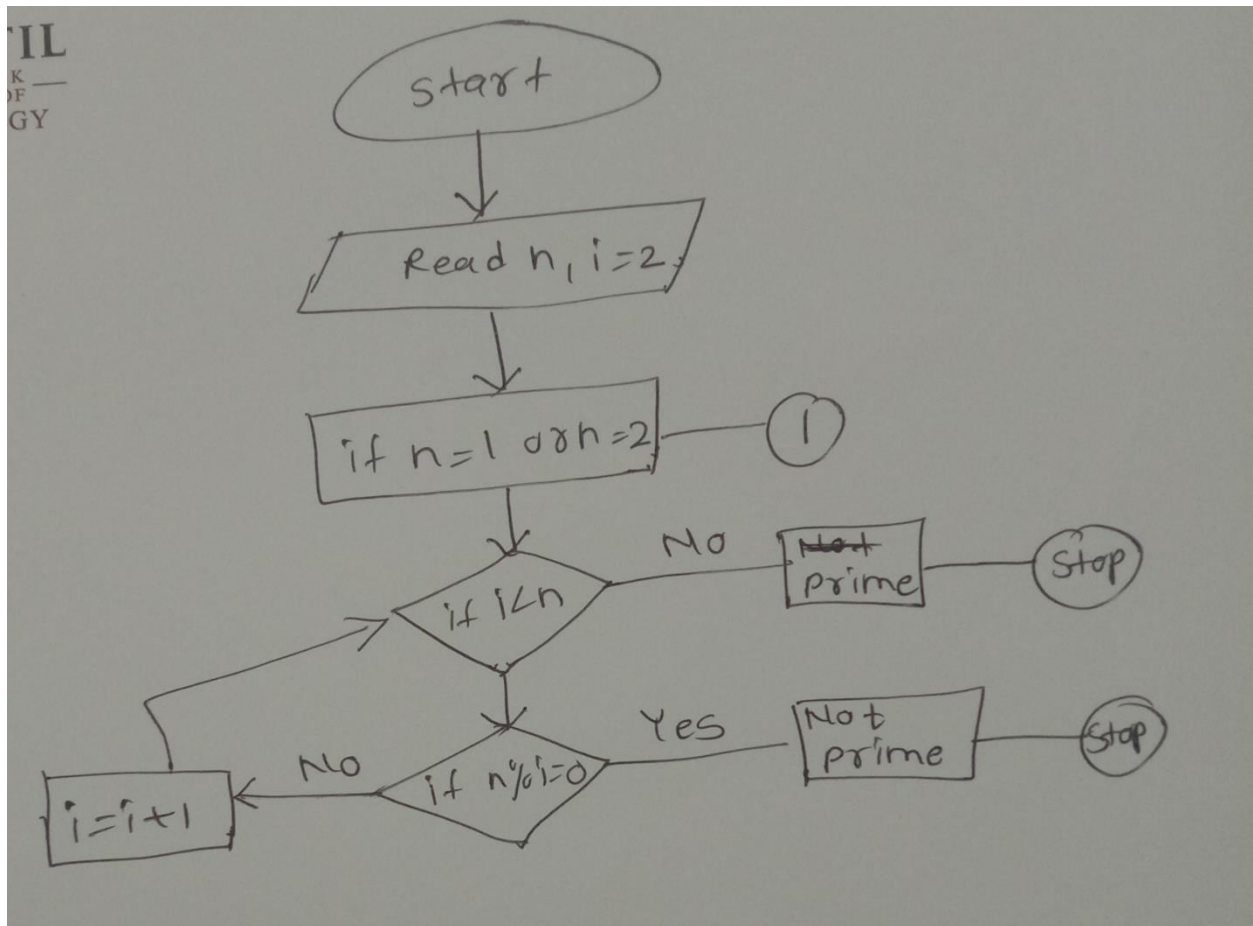
Input: 15

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"));
    }
}
```



```

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"));
    }
}
  
```

```

D:\Module 3\Day_1\Assignment1>javac Prime.java
D:\Module 3\Day_1\Assignment1>java Prime
Enter a number : 5
True
D:\Module 3\Day_1\Assignment1>java Prime
Enter a number : 4
False
D:\Module 3\Day_1\Assignment1>
  
```

Explanation:

- 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

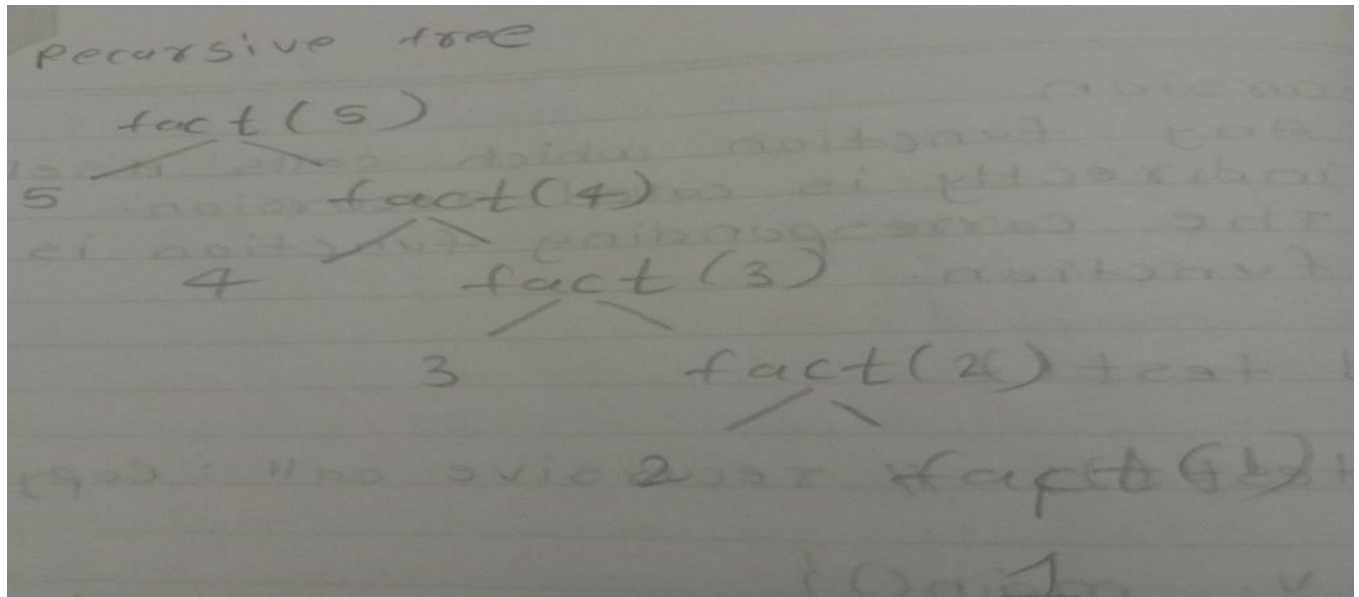
Output: 120

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));
    }
}
```



```
1 class Recursion3{
2     static int show(int n){
3         if(n<=1)
4             return n;
5         else
6             return n*show(n-1);
7     }
8
9     public static void main(String[] args){
10         System.out.println(show(5));
11     }
12 }
```

```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.22631.4169]
(c) Microsoft Corporation. All rights reserved.

D:\Module 3\Day_1>javac Recursion3.java
D:\Module 3\Day_1>java Recursion3
120
D:\Module 3\Day_1>
```

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 $5 * \text{show}(4)$ $4 * \text{show}(3)$ $3 * \text{show}(2)$ $2 * \text{show}(1)$ 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) + " ");
        }
    }
}
```

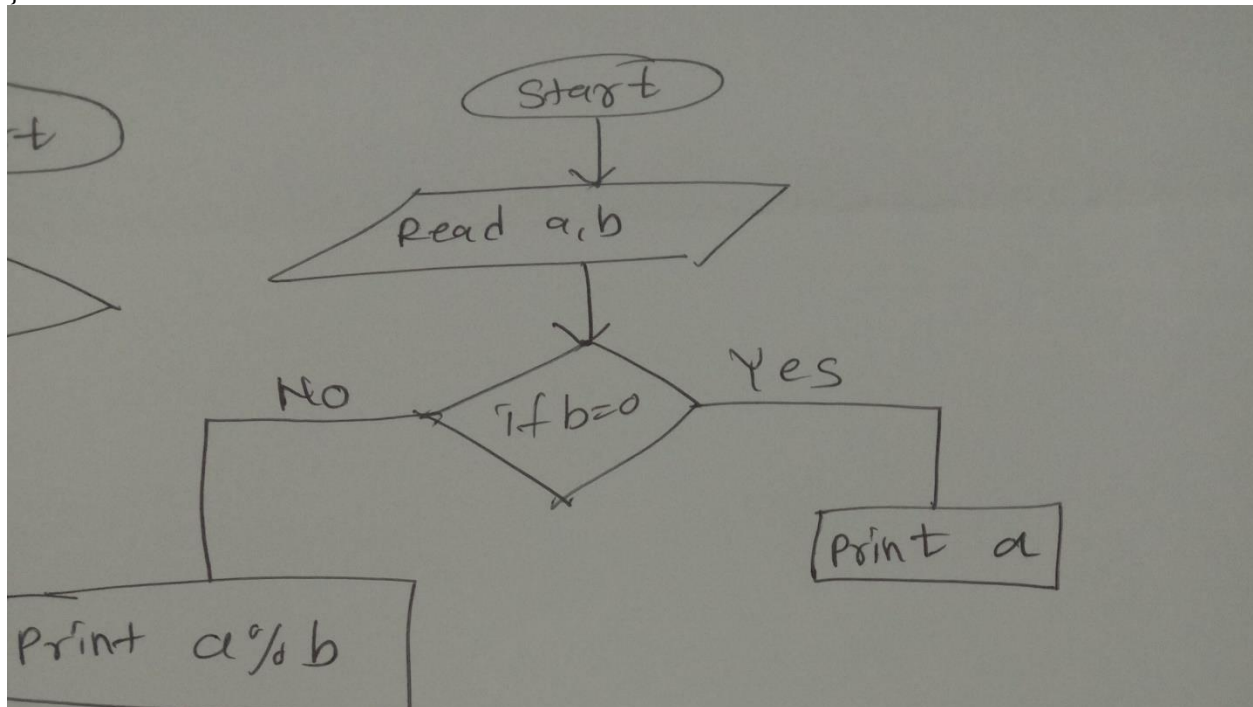

Output: 1

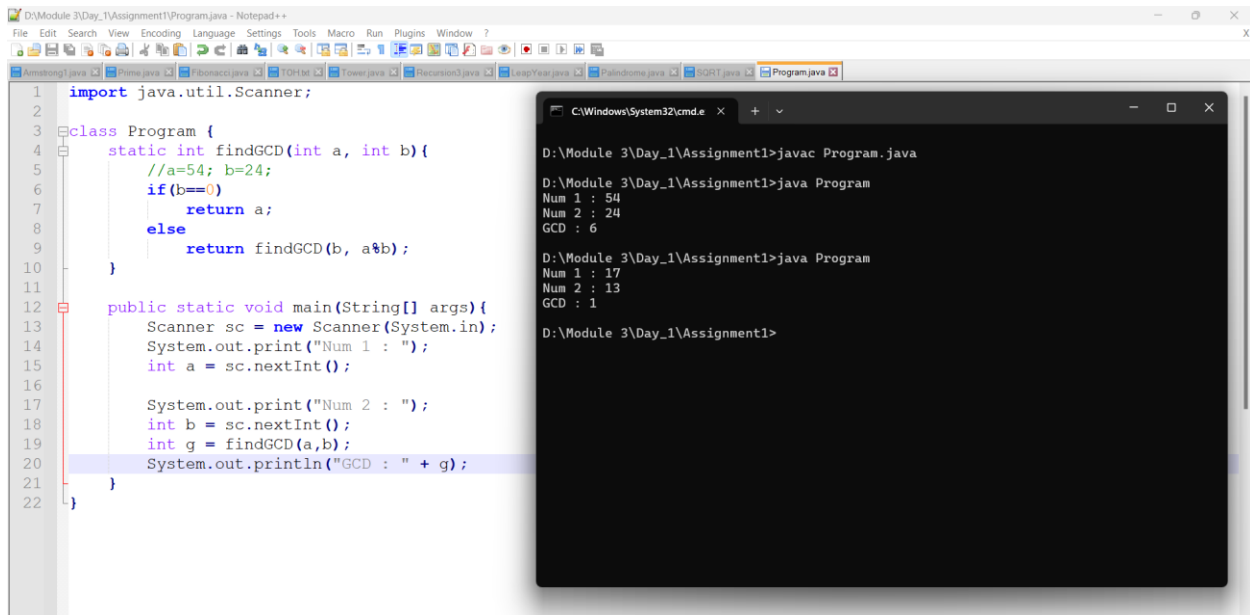
```
import java.util.Scanner;

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);
    }
}
```





Explanation:-

- 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;
```

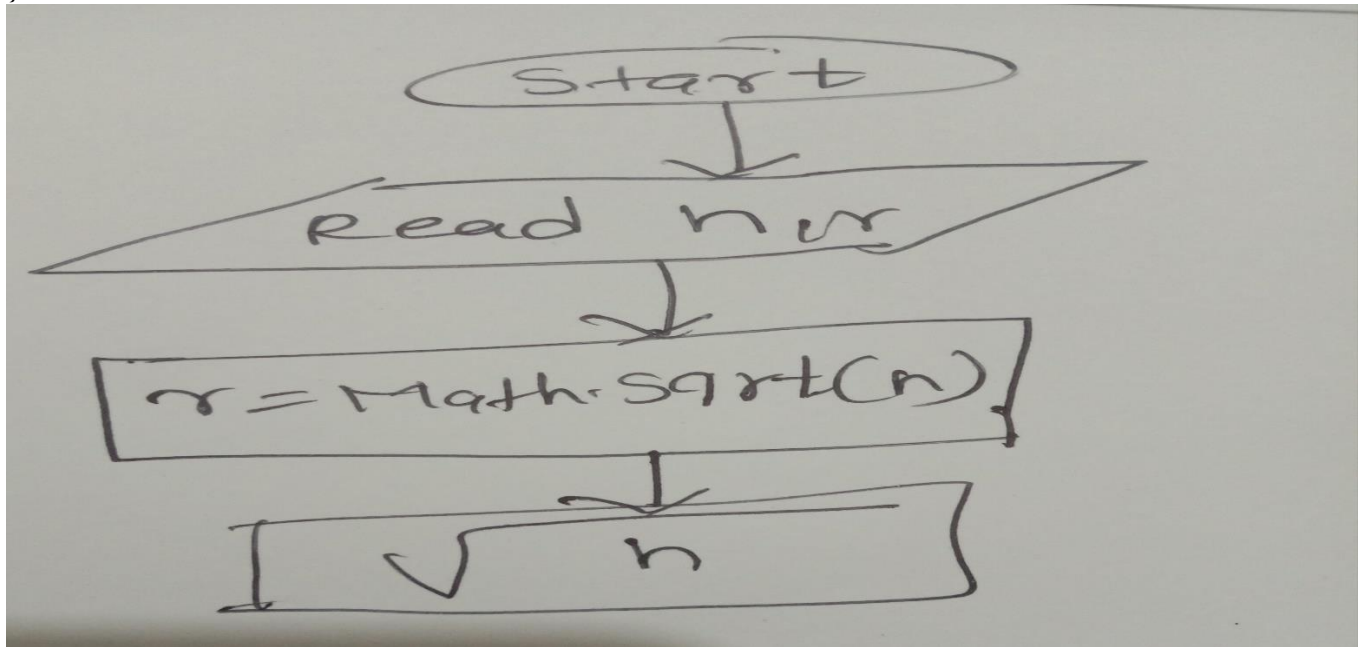
```

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));
    }
}

```

```
}  
}
```



The screenshot shows a Java IDE on the left and a Windows command prompt on the right. The IDE contains the following code:

```
1 import java.util.Scanner;  
2  
3 class Sqrt{  
4     static int show(int n){  
5         Scanner sc = new Scanner(System.in);  
6         System.out.print("Enter a number : ");  
7         int m = sc.nextInt();  
8         int r = (int) Math.sqrt(m);  
9         System.out.print("Square root : ");  
10        return r;  
11    }  
12  
13    public static void main(String[] args){  
14        System.out.println(show(0));  
15    }  
16 }
```

The command prompt shows the following output:

```
D:\Module 3\Day_1\Assignment1>javac Sqrt.java  
D:\Module 3\Day_1\Assignment1>java Sqrt  
Enter a number : 16  
Square root : 4  
D:\Module 3\Day_1\Assignment1>java Sqrt  
Enter a number : 25  
Square root : 5  
D:\Module 3\Day_1\Assignment1>
```

Explanation:-

- 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 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']

```
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);
    }

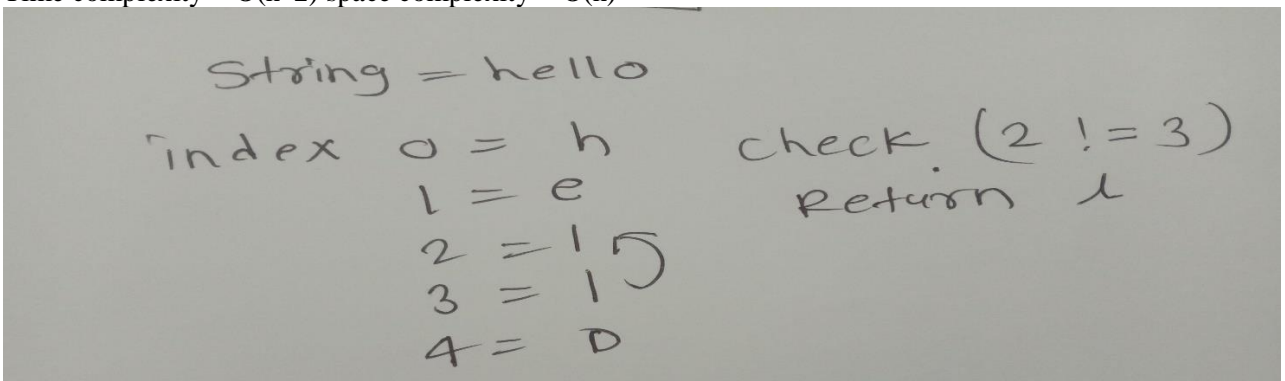
    public static void main(String[] args){
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a world : ");
        String input = sc.nextLine();

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

Explanation:-

- 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 each character 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)



The handwritten note shows the execution for the string "hello". It lists the index and character at each step: index 0 = h, index 1 = e, index 2 = l, index 3 = l, and index 4 = o. A check is performed at index 2 (2 != 3) and the result is 'l'.

```
String = hello
index 0 = h
      1 = e
      2 = l
      3 = l
      4 = o
check (2 != 3)
Return l
```

The screenshot shows a Notepad++ editor with a Java file named `DuplicateChar.java`. The code implements a recursive method `show` to find repeated characters in a string. The `main` method uses a `Scanner` to take input and prints the result. A terminal window in the foreground shows the compilation and execution of the program. The input "programming" results in the output "[r, m, g]" and the input "hello" results in the output "[l]".

```
1 import java.util.*;
2
3 class DuplicateChar{
4     static void show(String str, int index, List<Character> result){
5         if(index >= str.length())
6             return;
7
8         char current = str.charAt(index);
9         if(str.indexOf(current) != index && !result.contains(current)){
10             result.add(current);
11         }
12
13         show(str, index + 1, result);
14     }
15
16     public static void main(String[] args){
17         Scanner sc = new Scanner(System.in);
18         System.out.print("Enter a world : ");
19         String input = sc.nextLine();
20
21         List<Character> repeated = new ArrayList<>();
22         show(input, 0, repeated);
23         System.out.print(repeated);
24     }
25
26 }
```

```
D:\Module 3\Day_1\Assignment1>javac DuplicateChar.java
D:\Module 3\Day_1\Assignment1>java DuplicateChar
Enter a world : programming
[r, m, g]
D:\Module 3\Day_1\Assignment1>java DuplicateChar
Enter a world : hello
[l]
D:\Module 3\Day_1\Assignment1>
```

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 NonReapedted{
    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(""));
}
}

```

The screenshot shows a Notepad++ window with the following Java code for `NonReaped.java`:

```

1  import java.util.*;
2
3  class NonReaped{
4      static String show(String cc){
5          Scanner sc = new Scanner(System.in);
6          System.out.print("Enter a String : ");
7          String s = sc.nextLine();
8
9          Map<Character, Integer> m = new LinkedHashMap<>();
10         for(Character c : s.toCharArray()){
11             m.put(c, m.get(c) == null ? 1 : m.get(c)+1);
12         }
13
14         for(Character c : m.keySet()){
15             if(m.get(c) == 1){
16                 System.out.print(c + " ");
17             }
18         }
19
20
21         return cc;
22     }
23
24     public static void main(String[] args){
25         System.out.println(show(""));
26     }
27 }
28

```

Overlaid on the right is a Windows command prompt window showing the execution of the program:

```

D:\Module 3\Day_1\Assignment1>javac NonReaped.java
D:\Module 3\Day_1\Assignment1>java NonReaped
Enter a String : stress
t
r
e

D:\Module 3\Day_1\Assignment1>javac NonReaped.java
D:\Module 3\Day_1\Assignment1>java NonReaped
Enter a String : stress
t r e

D:\Module 3\Day_1\Assignment1>

```

Explanation:-

- 1) Use map to check unique 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"));
    }
}
```

Explanation:-

- 1) Did some mathematical steps to get the reverse no which I have written below
- 2) Then check whether 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 = $O(1)$

$$\text{digit} = 121 \% 10 \\ = 1$$

$$p = 0 \times 10 + 1 \\ = \underline{\underline{1}}$$

$$n = 121 / 10 \\ = 12$$

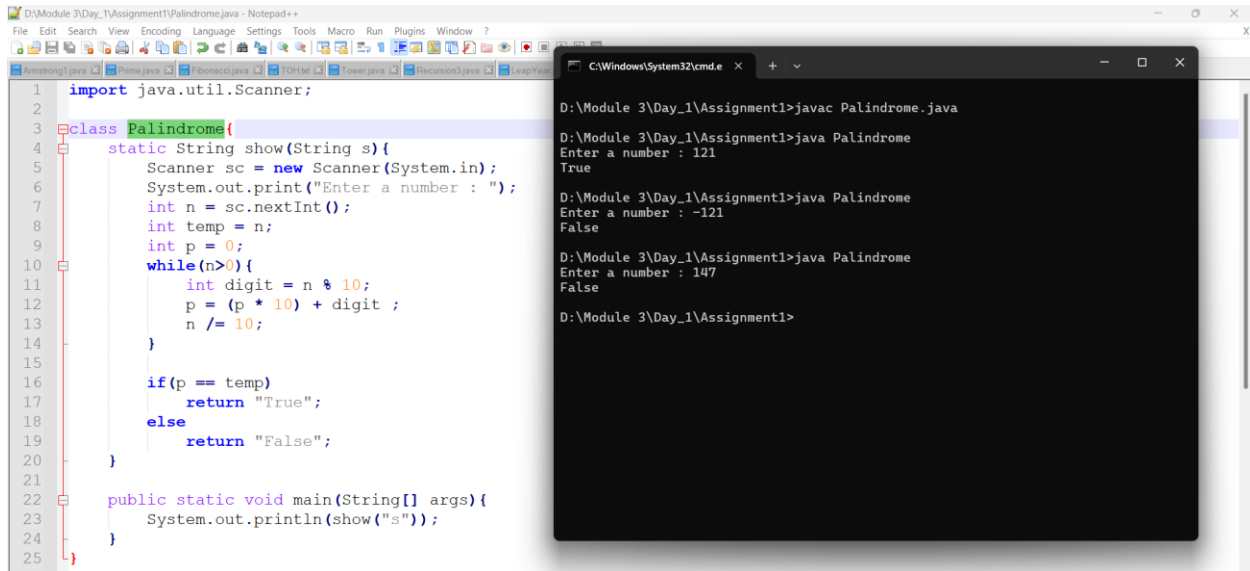
$$\text{digit} = 12 \% 10 \\ = 2$$

$$p = 1 \times 10 + 2 \\ = 12$$

$$n = 12 / 10 \\ = 1$$

$$\text{digit} = 1 \% 10 \\ = 1$$

$$p = 12 \times 10 + 1 \\ = \underline{\underline{121}}$$



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

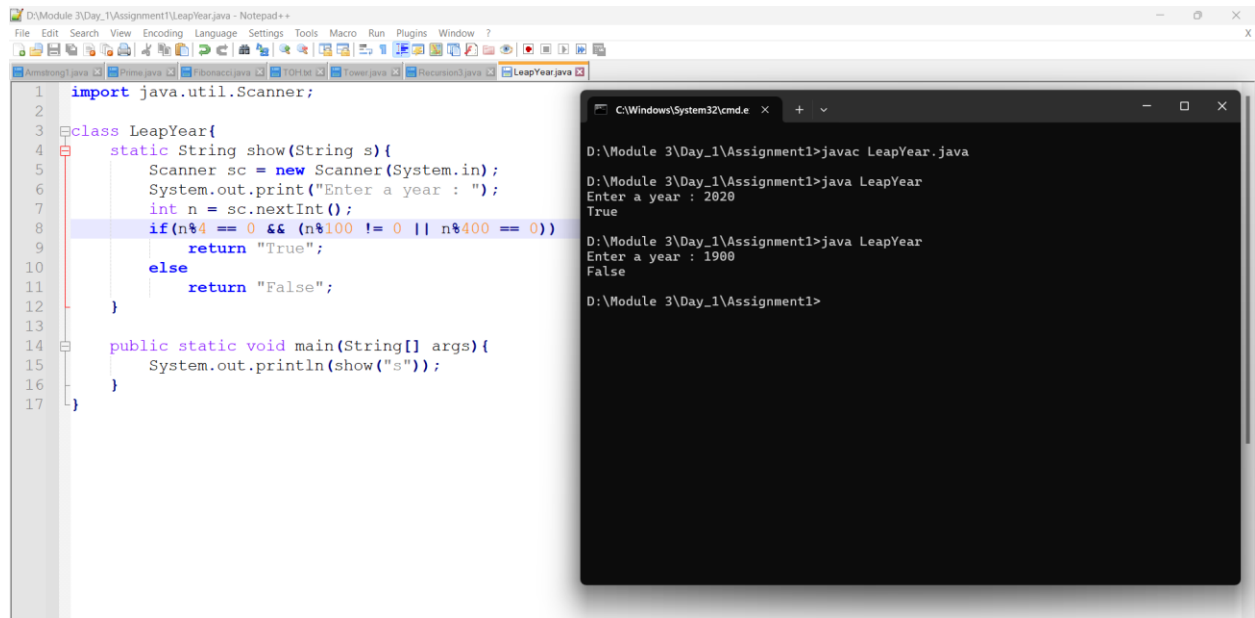
```
import java.util.Scanner;
```

```
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"));
    }
}
```

Explanation:-

- 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 = $O(1)$



```
1 import java.util.Scanner;
2
3 class LeapYear{
4     static String show(String s){
5         Scanner sc = new Scanner(System.in);
6         System.out.print("Enter a year : ");
7         int n = sc.nextInt();
8         if(n%4 == 0 && (n%100 != 0 || n%400 == 0))
9             return "True";
10        else
11            return "False";
12    }
13
14    public static void main(String[] args){
15        System.out.println(show("s"));
16    }
17 }
```

```
C:\Windows\System32\cmd.e x + v
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>
```