Day-4(Advance Concept):

Recursion

Searching & Sorting technique using Collection Framework

Hashing Techniques

----------------------------------------------------------------------------------------------------------------------------------------------------------

Q1.Write a program to compute the factorial of a number using recursion.

Input Format:

Input consists of an integer.

Output Format:

Print the factorial of the given number

Refer sample input and output for formatting specifications.

Sample Input:

5

Sample Output:

The factorial of 5 is 120

Case 1

Input (stdin)

5

Output (stdout)

The factorial of 5 is 120

Solution:

import java.util.\*;

class HelloWorld {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int num = sc.nextInt();

int result = fact(num);

System.out.println("The factorial of "+num+" is: "+ result);

}

static int fact(int num){

if(num==1){

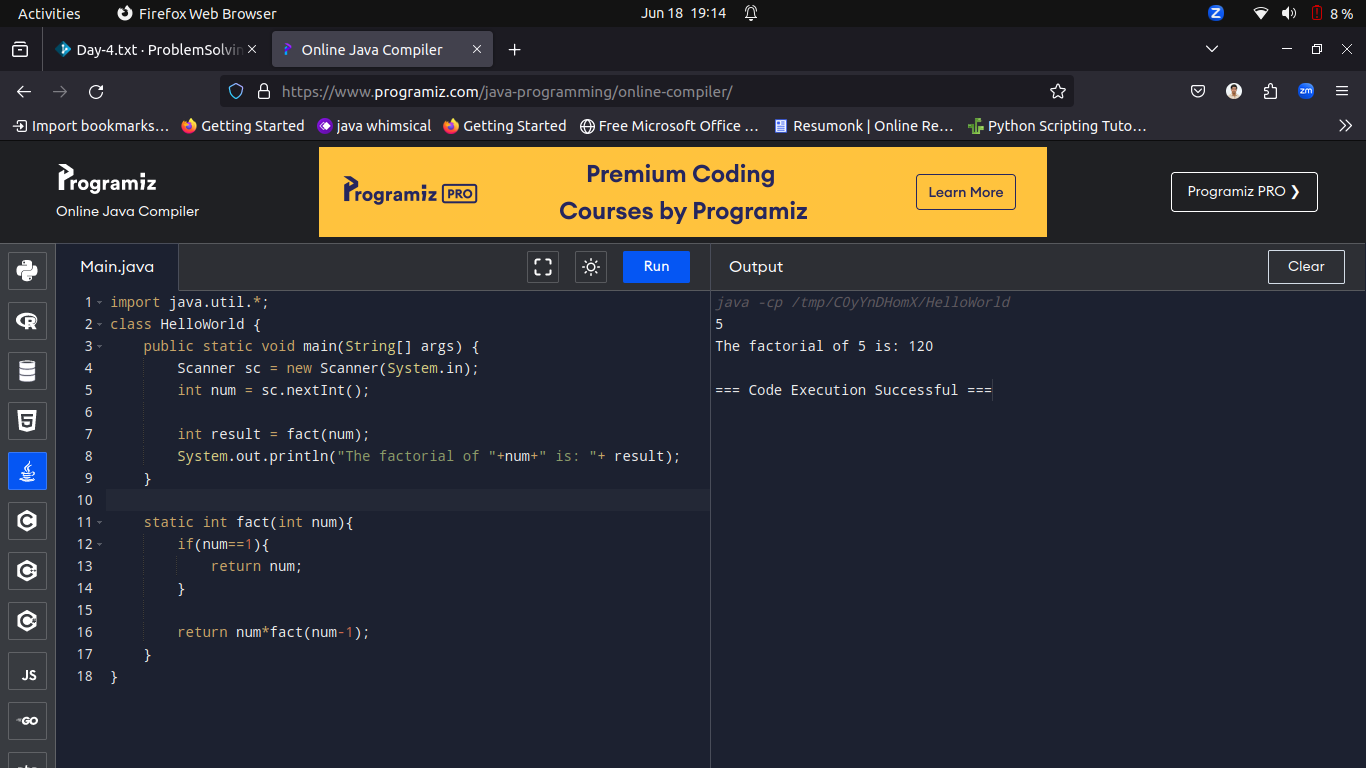
return num;

}

return num\*fact(num-1);

}

}



----------------------------------------------------------------------------------------------------------------------------------------------------------

Q2.Write a program to find the nth term in the Fibonacci series using recursion. Note that the first 2 terms in the Fibonacci Series are 0 and 1.

Input and Output Format:

Input consists of an integer.

Refer sample input and output for formatting specifications.

All text in bold corresponds to input and the rest corresponds to output.

Sample Input and Output:

4

Sample Output:

The term 4 in the fibonacci series is 2

Case 1

Input (stdin)

4

Output (stdout)

The term 4 in the fibonacci series is 2

Solution:

import java.util.\*;

class HelloWorld {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int num = sc.nextInt();

int result = fib(num-1);

System.out.println("The term "+num+" in fibonacci series: "+ result);

}

// 0 first term

// 1 second term

// so on

static int fib(int num){

if(num==1||num==0){

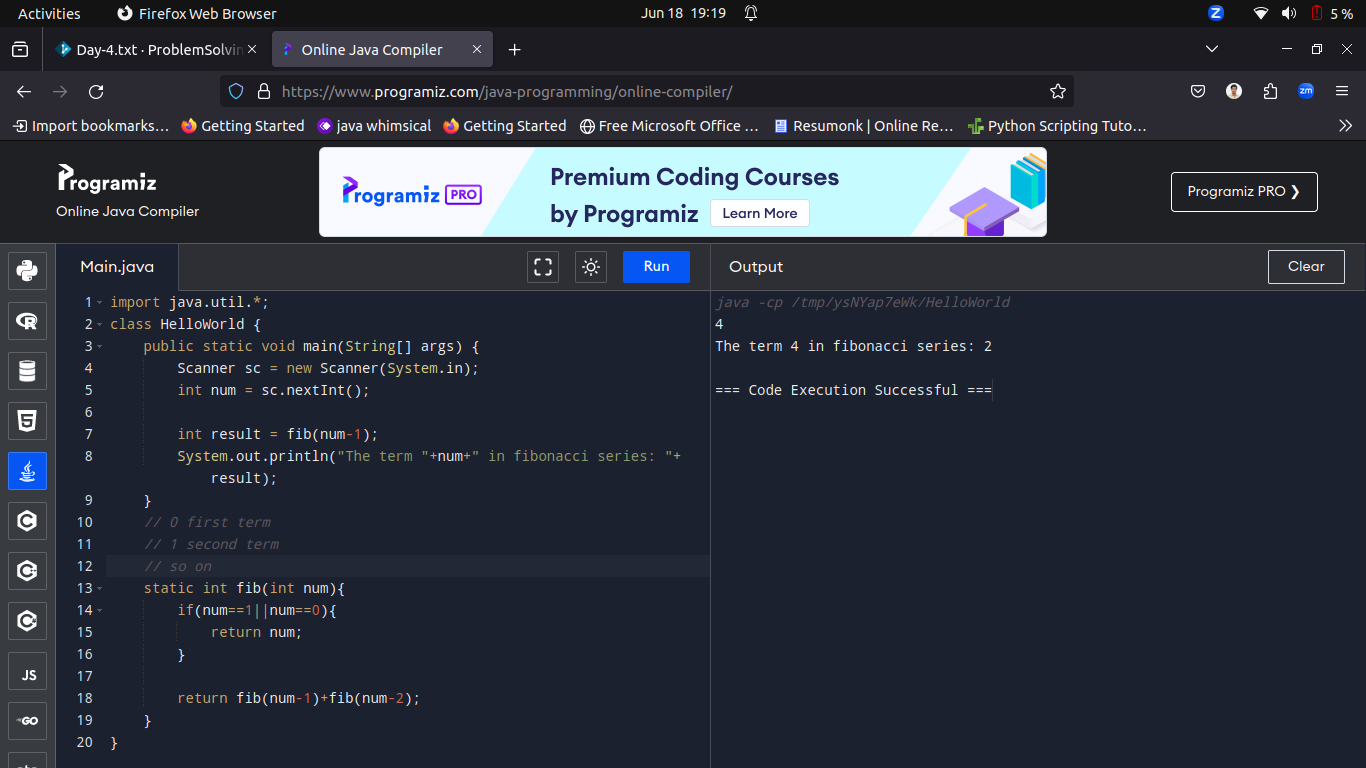
return num;

}

return fib(num-1)+fib(num-2);

}

}



----------------------------------------------------------------------------------------------------------------------------------------------------------

Q3.

Write a program to compute a^n (a power n) using recursion.

Input and Output Format:

Input consists of an integer.

Refer sample input and output for formatting specifications.

Sample Input and Output:

2

8

Sample Output:

The value of 2 power 8 is 256

Case 1

Input (stdin)

2

8

Output (stdout)

The value of 2 power 8 is 256

Solution:

import java.util.\*;

class HelloWorld {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int num = sc.nextInt();

int pow = sc.nextInt();

System.out.println("The value of "+ num +" power "+ pow +" is "+power(num, pow));

}

public static int power(int x, int n)

{

// x^0 return 1

if (n == 0)

return 1;

// 0^y is always 0

if (x == 0)

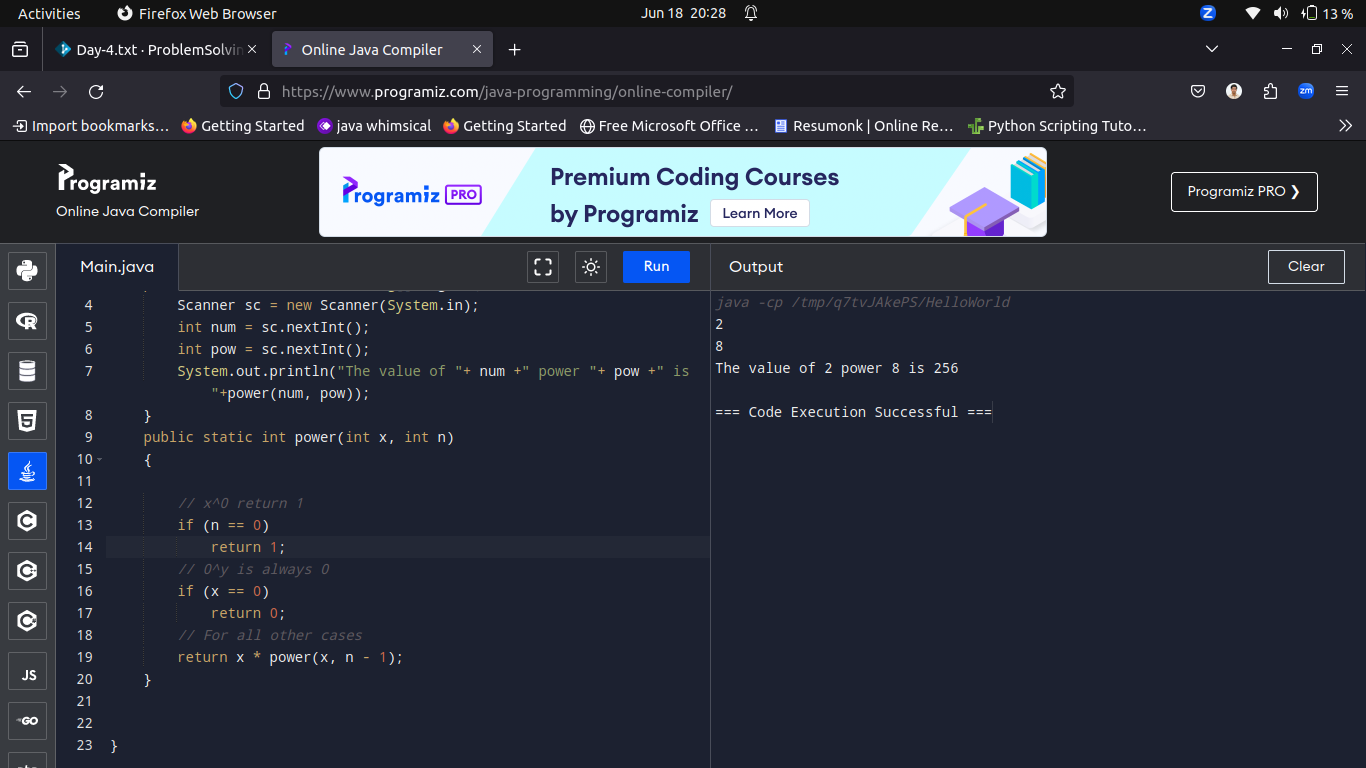
return 0;

// For all other cases

return x \* power(x, n - 1);

}

}



----------------------------------------------------------------------------------------------------------------------------------------------------------

Q4.Write a program to find the sum of digits in a number using recursion.

Input Format:

Input consists of a non - negative integer.

Output Format:

Print the sum of digits of a given number.

Refer sample input and output for formatting specifications.

Sample Input:

432

Sample Output:

The sum of digits in 432 is 9

Case 1

Input (stdin)

432

Output (stdout)

The sum of digits in 432 is 9

solution:

import java.util.\*;

class HelloWorld {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int num = sc.nextInt();

System.out.println("The sum of digits in "+ num +" is "+sum\_of\_digits(num));

}

public static int sum\_of\_digits(int n)

{

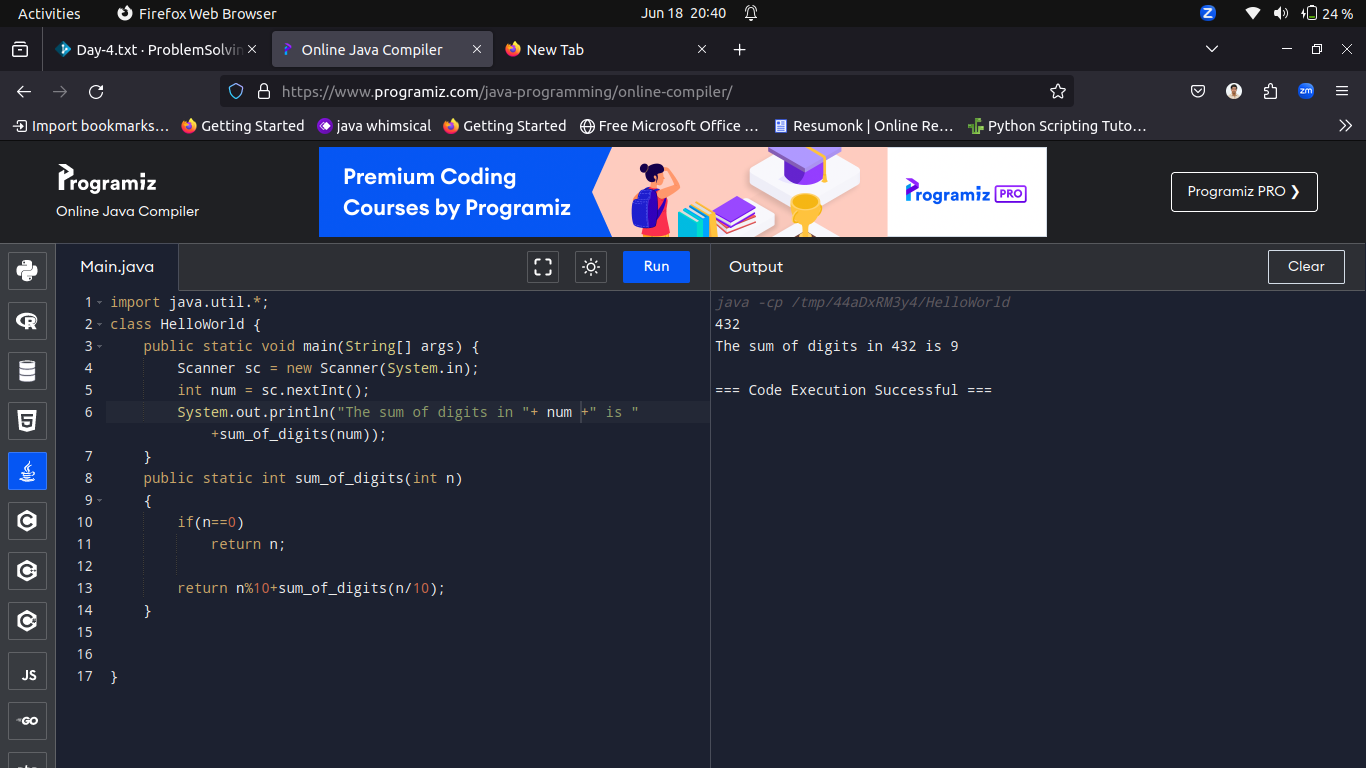
if(n==0)

return n;

return n%10+sum\_of\_digits(n/10);

}

}



----------------------------------------------------------------------------------------------------------------------------------------------------------

Q5.Write a program to find the number of digits in a number using recursion.

Input Format:

Input consists of a non - negative integer.

Output Format:

Print number of digits of a given number

Refer sample input and output for formatting specifications.

Sample Input:

432

Sample Output:

The number of digits in 432 is 3

Case 1

Input (stdin)

432

Output (stdout)

The number of digits in 432 is 3

Solution:

import java.util.\*;

class HelloWorld {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int num = sc.nextInt();

System.out.println("The sum of digits in "+ num +" is "+sum\_of\_digits(num));

}

public static int sum\_of\_digits(int n)

{

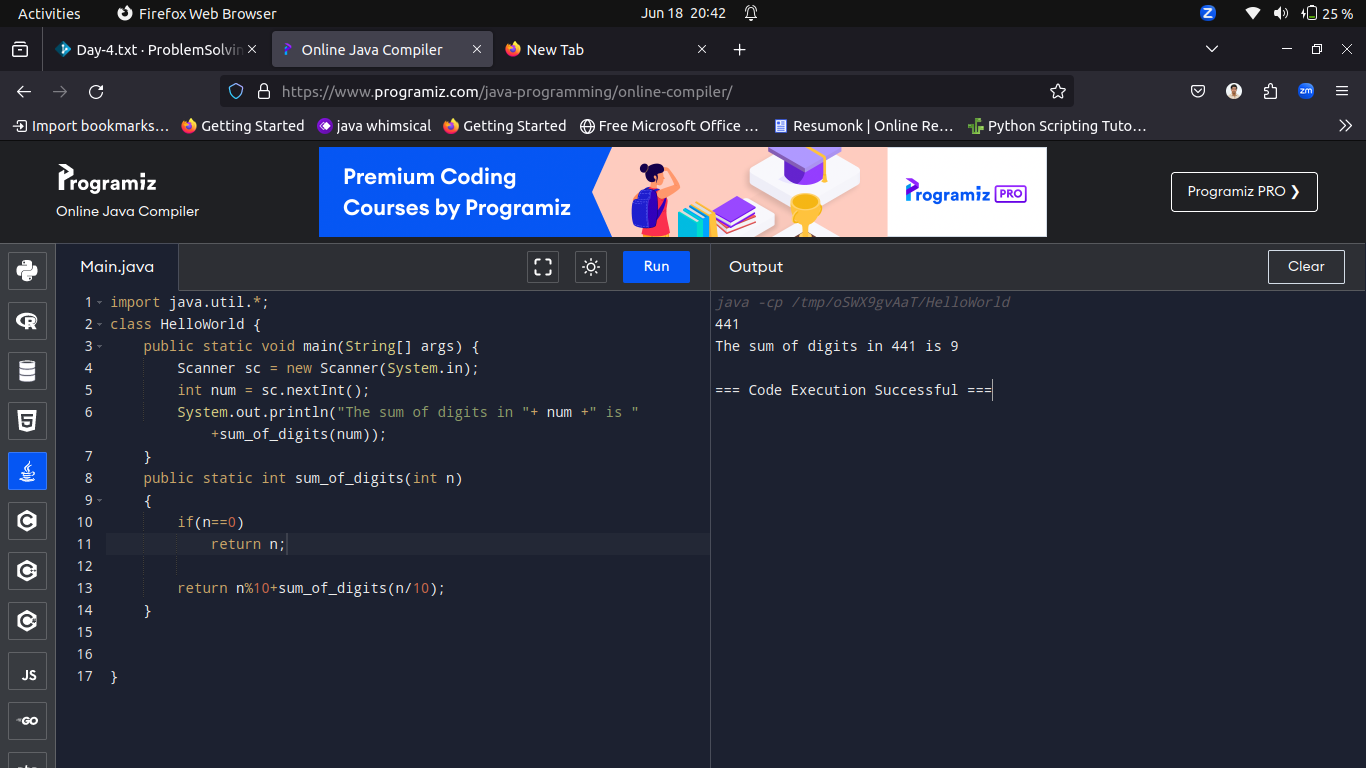
if(n==0)

return n;

return n%10+sum\_of\_digits(n/10);

}

}



----------------------------------------------------------------------------------------------------------------------------------------------------------

Q6.Write a Program to peform Binary Search using ArrayList Collection.

Input:

6

10

20

30

40

50

60

40

Output:

3

Solution:

import java.util.\*;

public class HelloWorld {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int n = scanner.nextInt();

ArrayList<Integer> arrayList = new ArrayList<>();

for (int i = 0; i < n; i++) {

arrayList.add(scanner.nextInt());

}

System.out.println("Search element");

int search = scanner.nextInt();

// Sort the arraylist to perform binary search

Collections.sort(arrayList);

int index = binarySearch(arrayList, search);

System.out.println(index);

scanner.close();

}

public static int binarySearch(ArrayList<Integer> list, int search) {

int low = 0;

int high = list.size() - 1;

while (low <= high) {

int mid = low + (high - low) / 2;

int midVal = list.get(mid);

if (midVal < search) {

low = mid + 1;

} else if (midVal > search) {

high = mid - 1;

} else {

return mid;

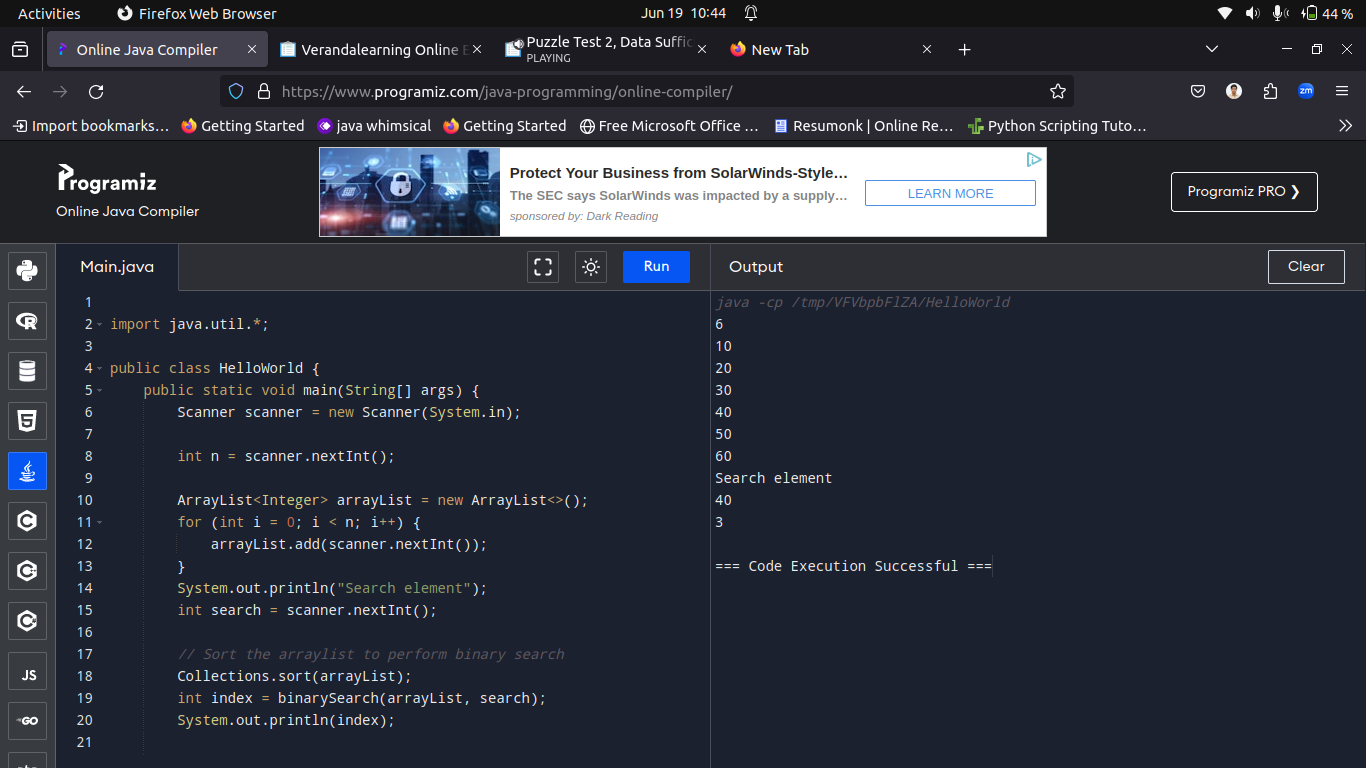
}

}

return -1; // element not found

}

}



----------------------------------------------------------------------------------------------------------------------------------------------------------

Q7.You are given a list of integers. Your task is to find the sum of all the elements in the list.

input:

5

1 2 3 4 5

Ouptut:

Sum of the integers: 15

Solution:

import java.util.\*;

class HelloWorld {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int num = sc.nextInt();

List<Integer> li = new ArrayList<>();

int sum =0;

for(int i=0;i<num;i++){

int temp = sc.nextInt();

li.add(temp);

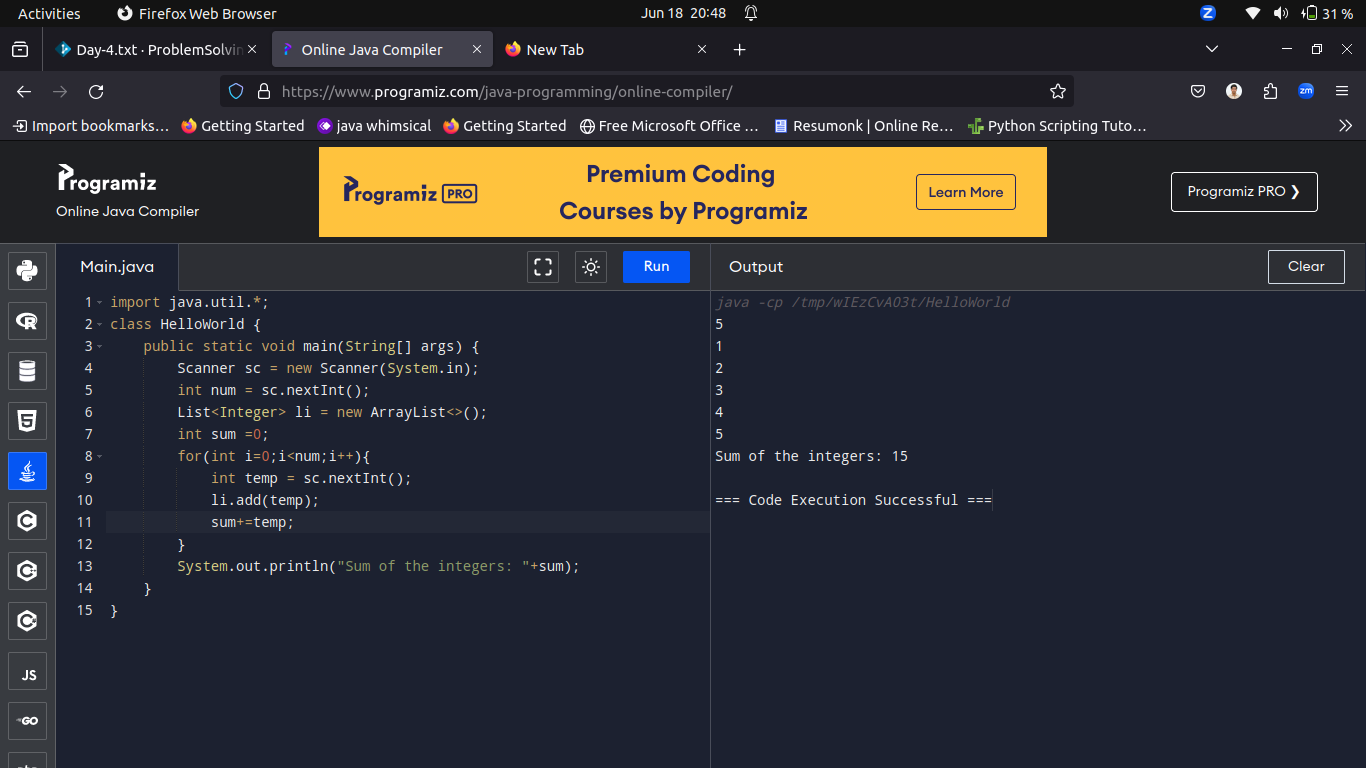
sum+=temp;

}

System.out.println("Sum of the integers: "+sum);

}

}



----------------------------------------------------------------------------------------------------------------------------------------------------------

Q8.You are given a list of strings. Your task is to find the length of the longest string in the list.

Input:

5

apple banana kiwi orange strawberry

Output:

Length of the longest string: 10

Solution:

import java.util.\*;

class HelloWorld {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

sc.nextLine();

String inputLine = sc.nextLine();

String[] input = inputLine.split(" ");

int maxLen = 0;

for(int i=0;i<n;i++){

if(maxLen<input[i].length())

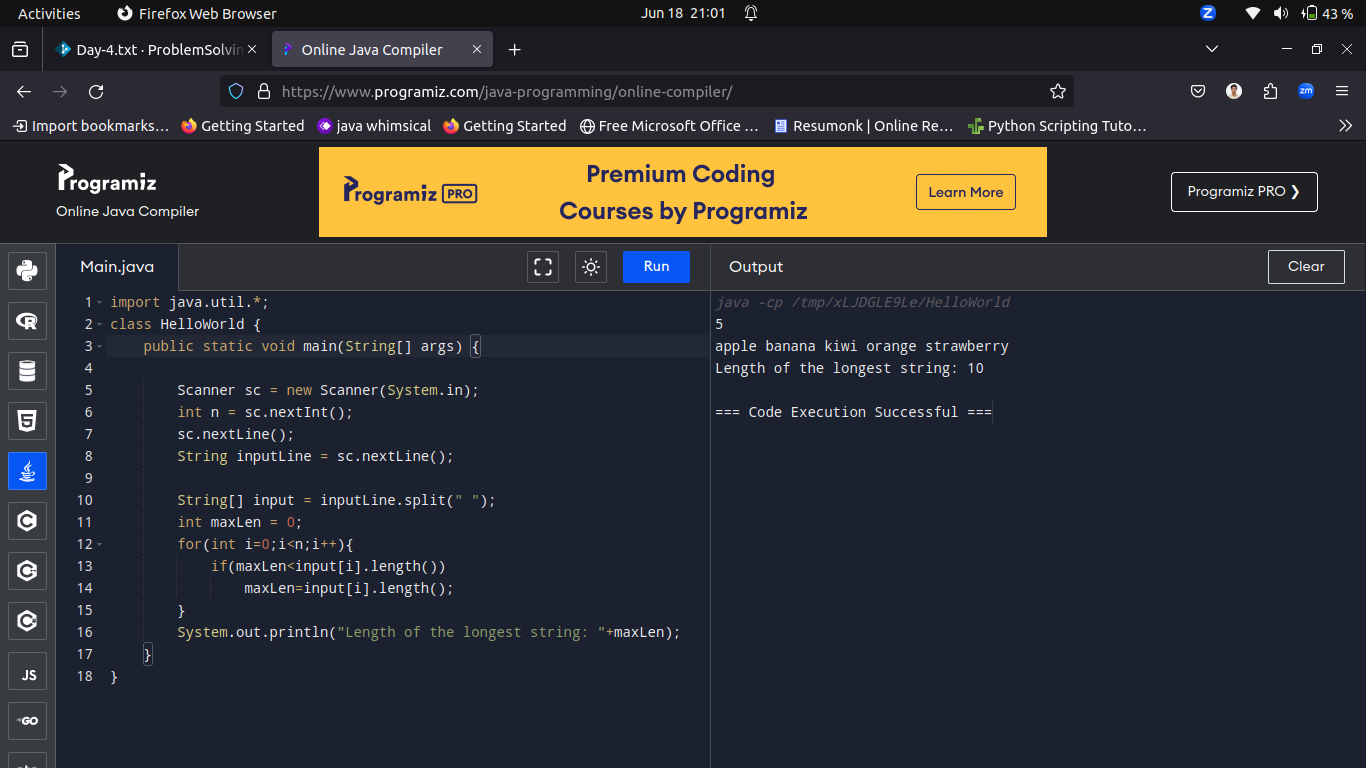
maxLen=input[i].length();

}

System.out.println("Length of the longest string: "+maxLen);

}

}



----------------------------------------------------------------------------------------------------------------------------------------------------------

Q9.You are given two lists of integers. Your task is to find the common elements present in both lists.

Input:

5

1 2 3 4 5

4

4 5 6 7

Output: [4, 5]

Solution:

import java.util.\*;

class HelloWorld {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n1 = sc.nextInt();

sc.nextLine();

String[] input1 = sc.nextLine().split(" ");

List<Integer> list1 = new ArrayList<>();

for(String e: input1){

list1.add(Integer.parseInt(e));

}

int n2 = sc.nextInt();

sc.nextLine();

String[] input2 = sc.nextLine().split(" ");

List<Integer> list2 = new ArrayList<>();

for(String e: input2){

list2.add(Integer.parseInt(e));

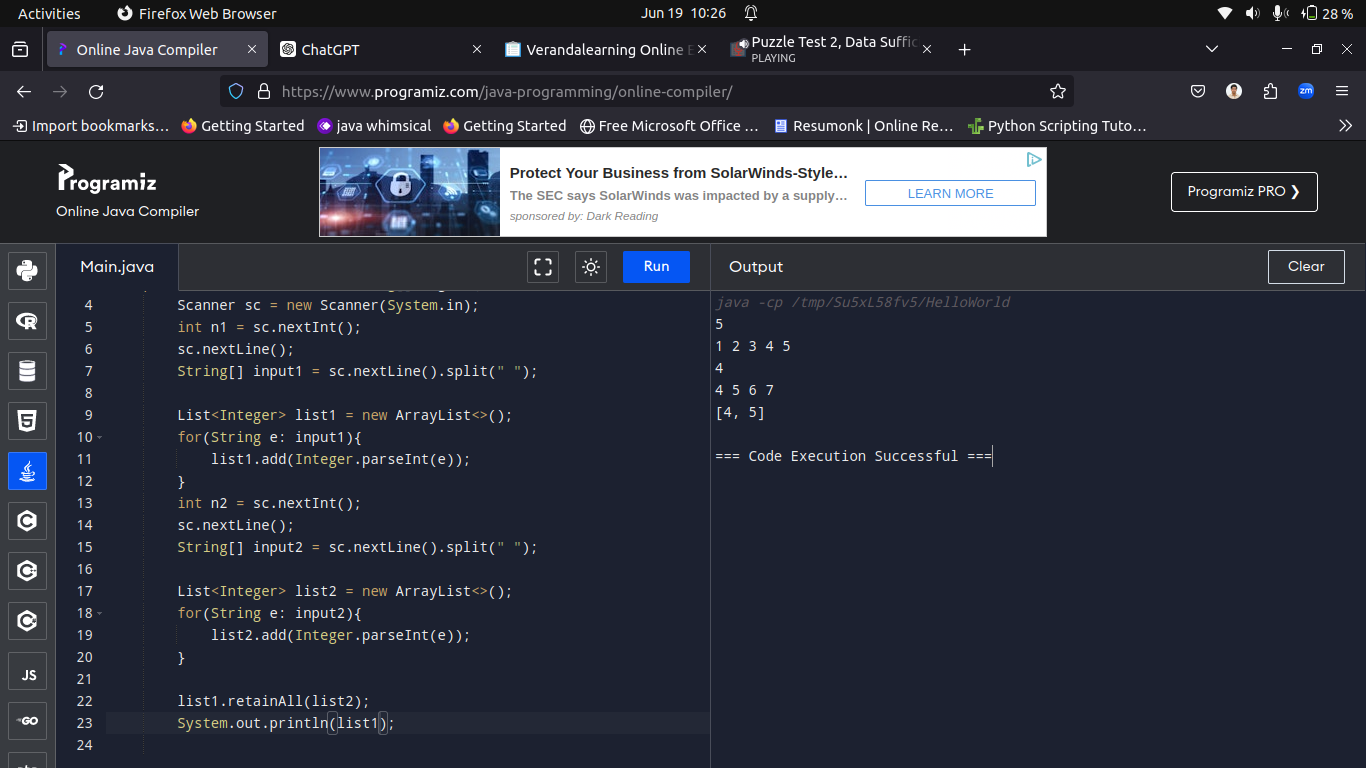
}

list1.retainAll(list2);

System.out.println(list1);

}

}



----------------------------------------------------------------------------------------------------------------------------------------------------------

Q10.You are given a list of integers. Your task is to remove duplicates from the list while preserving the order of the elements

Input:

8

1 2 3 2 4 5 3 6

output:

[1, 2, 3, 4, 5, 6]

Solution:

import java.util.\*;

class HelloWorld {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n1 = sc.nextInt();

sc.nextLine();

String[] input1 = sc.nextLine().split(" ");

List<Integer> list = new ArrayList<>();

for(String e: input1){

list.add(Integer.parseInt(e));

}

Set<Integer> set = new LinkedHashSet<>();

// add list to set

set.addAll(list);

// Clear the list

list.clear();

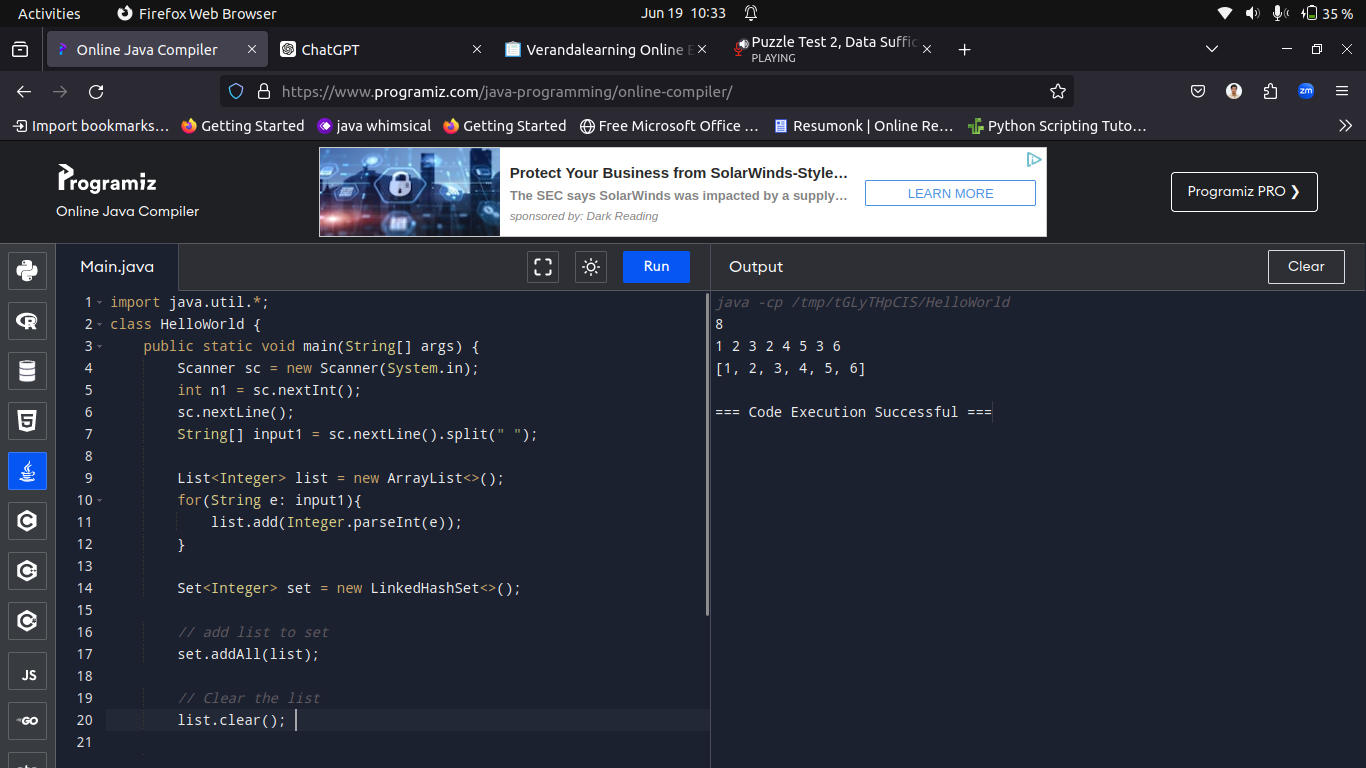
// set removes the duplicates

list.addAll(set);

System.out.println(list);

}

}



----------------------------------------------------------------------------------------------------------------------------------------------------------