Day-5(Miscellaneous Problem Solving):

Introduction to Dynamic Programming

Fibonacci Series using Dynamic Programming

Solving Problems using Regex.

Java-8-17 Overview

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

Q1.Egg Dropping Puzzle

Imagine that you have N eggs and you have to determine from which K-floor of a building, you can drop an egg such that it doesn't break. You have to determine the minimum number of attempts. Which you need in order to find the critical floor in the worst case while using the best strategy. There are few rules given below:

An egg that survives a fall can be used again.

A broken egg must be discarded.

The effect of a fall is the same for all the eggs.

If the egg doesn't break on a certain floor, it will not break on any of the floors below the present floor.

If the eggs break at a certain floor, they will break at any of the floors above the present floor.

Format:

Input:

The first line of input is T denoting the number of test cases. Then each of the T lines contains two positive integers N and K. Where 'N' is the number of eggs and 'K' is the number of floors in a building.

Output:

For each test case, print a single line containing one integer. Which gives the minimum number of attempts you need in order to find a critical floor.

Constraints:

1 <= T <= 30

1 <= N <= 10

1 <= K <= 50

﻿Example:

Input:

2

2 10

3 5

Output:

4

3

Case 1

Case 2

Input (stdin)

2

2 10

3 5

Output (stdout)

4

3

Input (stdin)

1

9 49

Output (stdout)

6

Solution:

import java.util.Scanner;

public class EggDrop {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

int T = scanner.nextInt(); // number of test cases

int count = 1;

while (T-- > 0) {

System.out.println("Test case "+count);

int N = scanner.nextInt(); // number of eggs

int K = scanner.nextInt(); // number of floors

int result = eggDrop(N, K);

System.out.println("Result "+count++ +": "+result);

}

scanner.close();

}

public static int eggDrop(int N, int K) {

// dp[n][k] will store minimum number of attempts needed with n eggs and k floors

int[][] dp = new int[N + 1][K + 1];

// With 0 floors, 0 attempts needed; with 1 floor, 1 attempt needed

for (int n = 1; n <= N; n++) {

dp[n][0] = 0;

dp[n][1] = 1;

}

// With 1 egg, k attempts needed for k floors

for (int k = 1; k <= K; k++) {

dp[1][k] = k;

}

// Fill the dp table

for (int n = 2; n <= N; n++) {

for (int k = 2; k <= K; k++) {

dp[n][k] = Integer.MAX\_VALUE;

for (int x = 1; x <= k; x++) {

// Case 1: Egg breaks -> Search in floors below x

// Case 2: Egg doesn't break -> Search in floors above x

int worst\_case\_attempts = 1 + Math.max(dp[n - 1][x - 1], dp[n][k - x]);

dp[n][k] = Math.min(dp[n][k], worst\_case\_attempts);

}

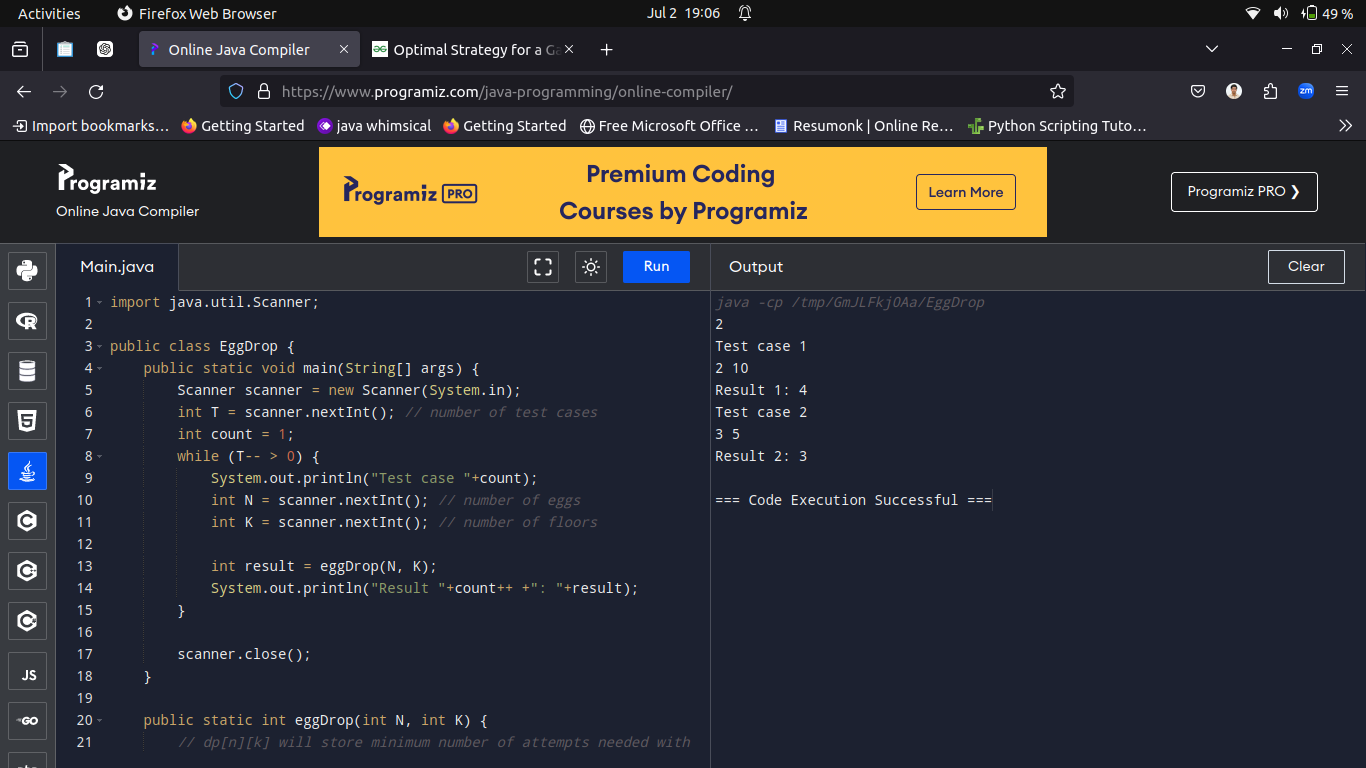
}

}

return dp[N][K];

}

}



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

Q2.Coin Turn Game

Consider a row of n coins. We play a game against an opponent by alternative turns. In each turn, a player selects either the first or last coin from the row. Now remove it from the row permanently and take the value of a coin. Find the maximum possible amount of money.

Example:

Input:

4

5 3 7 10

Output:

15

Case 1

Case 2

Case 3

Case 6

Case 7

Case 8

Case 9

Case 10

Input (stdin)

4

5 3 7 10

Output (stdout)

15

Input (stdin)

7

8 15 3 7 10 22 5

Output (stdout)

26

Input (stdin)

8

10 3 8 2 6 7 15 1

Output (stdout)

39

Input (stdin)

5

1 2 3 4 5

Output (stdout)

9

Input (stdin)

7

11 22 33 44 55 66 88

Output (stdout)

187

Solution:

import java.util.\*;

public class CoinTurnGame {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

int [] coins = new int[n];

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

coins[i] = sc.nextInt();

}

int[][] dp = new int[n][n];

// Fill dp array

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

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

int x = (i+2 <= j) ? dp[i+2][j] : 0;

int y = (i+1 <= j-1) ? dp[i+1][j-1] : 0;

int z = (i <= j-2) ? dp[i][j-2] : 0;

dp[i][j] = Math.max(coins[i] + Math.min(x, y),

coins[j] + Math.min(y, z));

}

}

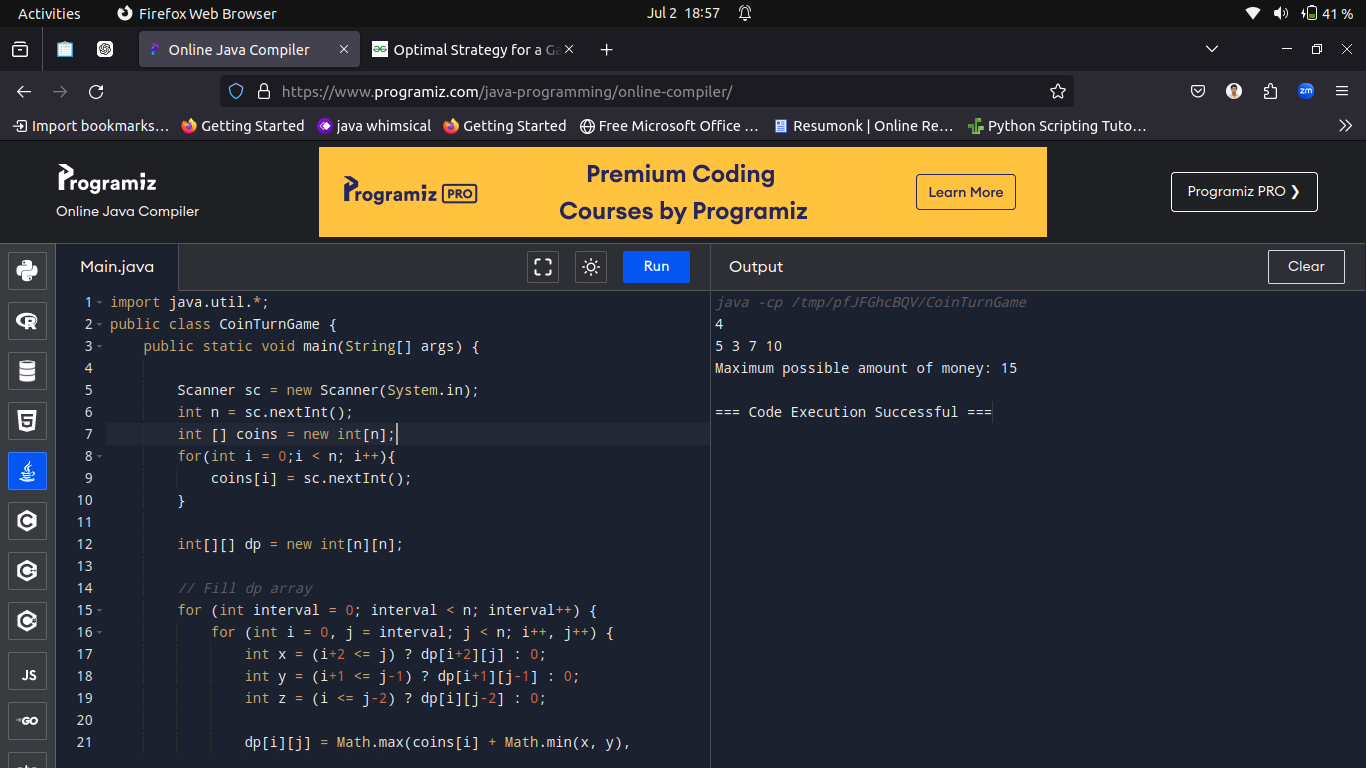
// The result will be in dp[0][n-1]

int maxMoney = dp[0][n-1];

System.out.println("Maximum possible amount of money: " + maxMoney);

}

}



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

Q3.You are given a string representing a phone number in the format (XXX) XXX-XXXX. Your task is to validate whether the given string is a

valid phone number or not.

Input:

Enter a phone number:

(123) 456-7890

Output:

true

Solution:

import java.util.regex.\*;

import java.util.\*;

public class PhoneNumberValidator {

public static void main(String[] args) {

// String phoneNumber = "(123) 456-7890";

Scanner sc = new Scanner(System.in);

String phoneNumber = sc.nextLine();

boolean isValid = isValidPhoneNumber(phoneNumber);

System.out.println(isValid); // Output: true

}

public static boolean isValidPhoneNumber(String phoneNumber) {

String regex = "^\\(\\d{3}\\) \\d{3}-\\d{4}$";

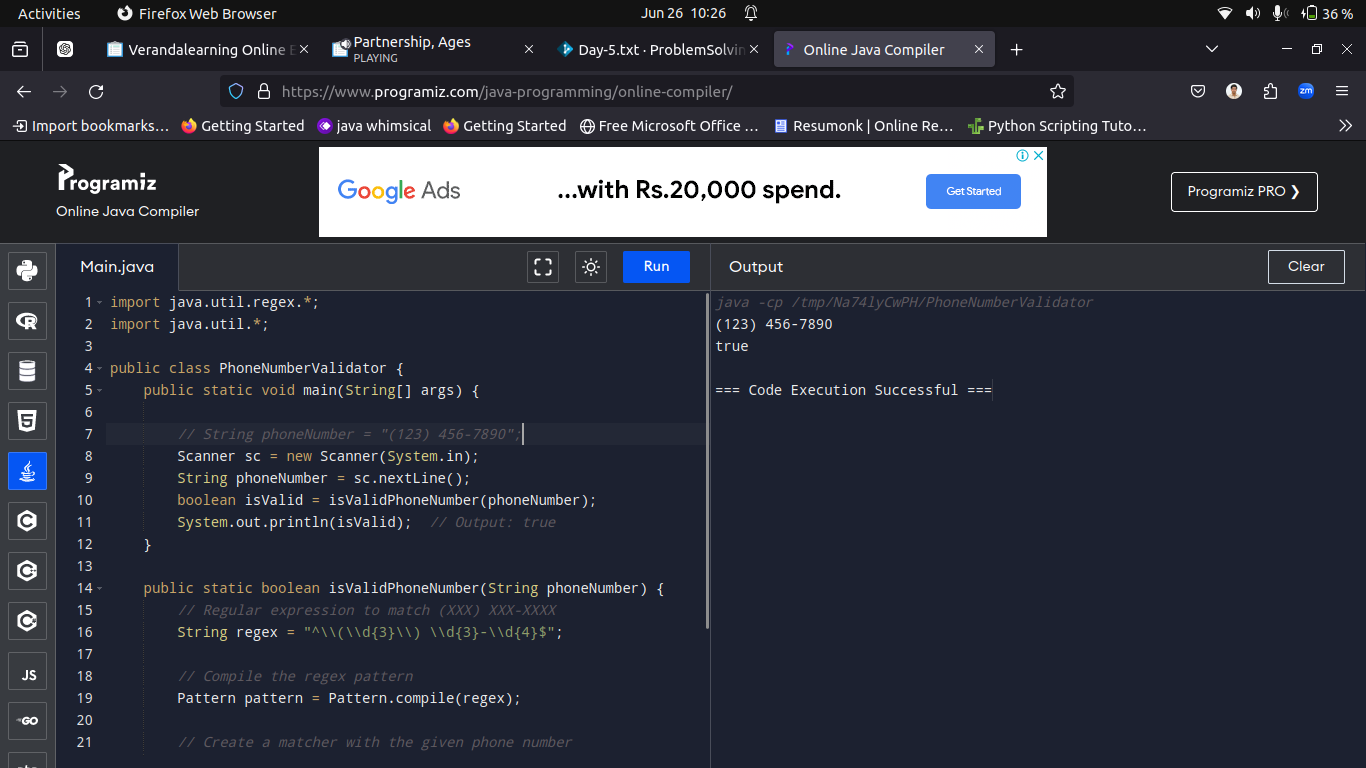
Pattern pattern = Pattern.compile(regex);

Matcher matcher = pattern.matcher(phoneNumber);

return matcher.matches();

}

}



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

Q4.You are given a string containing a list of email addresses separated by commas. Your task is to extract all the email addresses from the string.

input:

Contact us at test@guvi.in or lavish@guvi.in

Output:

[test@guvi.in, lavish@guvi.in]

Solution:

import java.util.ArrayList;

import java.util.List;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import java.util.\*;

public class EmailExtractor {

public static void main(String[] args) {

// String input = "Contact us at test@guvi.in or lavish@guvi.in";

Scanner sc = new Scanner(System.in);

String input = sc.nextLine();

List<String> emails = extractEmails(input);

System.out.println(emails);

}

public static List<String> extractEmails(String text) {

List<String> emails = new ArrayList<>();

String regex = "\\b[A-Za-z0-9.\_%+-]+@[A-Za-z0-9.-]+\\.[A-Z|a-z]{2,}\\b";

Pattern pattern = Pattern.compile(regex);

Matcher matcher = pattern.matcher(text);

while (matcher.find()) {

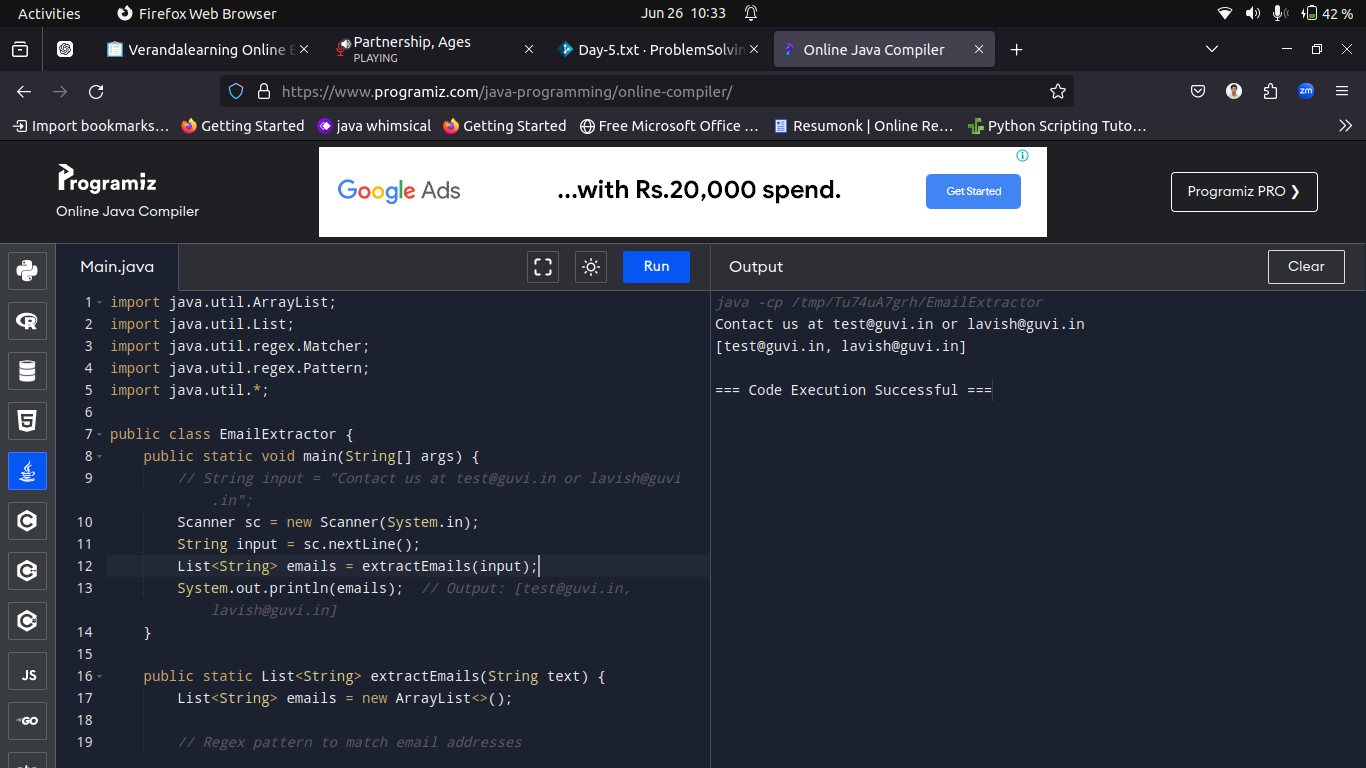
emails.add(matcher.group());

}

return emails;

}

}



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

Q5.You are given a string containing a word. Your task is to check if the word starts with the letter 'A' or 'a'.

Input:

apple

Output:

True

Solution:

import java.util.regex.\*;

import java.util.\*;

public class StartsWithRegexExample {

public static void main(String[] args) {

// String word = "apple";

Scanner sc = new Scanner(System.in);

String word = sc.nextLine();

boolean startsWithA = startsWithAorLowerCaseARegex(word);

System.out.println(startsWithA);

}

public static boolean startsWithAorLowerCaseARegex(String word) {

String regex = "^[Aa].\*";

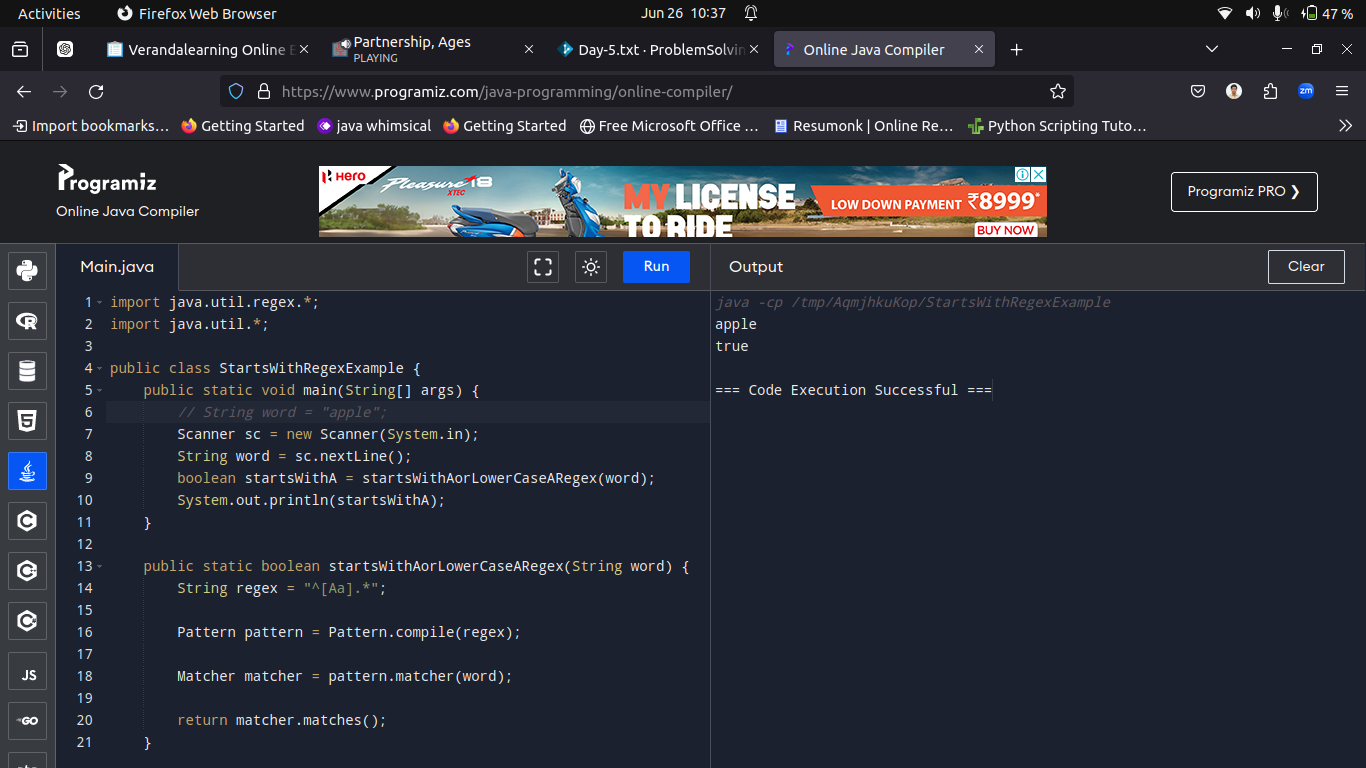
Pattern pattern = Pattern.compile(regex);

Matcher matcher = pattern.matcher(word);

return matcher.matches();

}

}



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

Q6.You are given a string representing a sentence. Your task is to check if the sentence ends with a question mark.

Input:

How are you?

Output:

true

SOlution:

import java.util.regex.\*;

import java.util.\*;

public class SentenceQuestionCheckerRegex {

public static void main(String[] args) {

// String sentence = "How are you?";

Scanner sc = new Scanner(System.in);

String sentence = sc.nextLine();

boolean endsWithQuestionMark = endsWithQuestionMarkRegex(sentence);

System.out.println(endsWithQuestionMark);

}

public static boolean endsWithQuestionMarkRegex(String sentence) {

String regex = "\\?$";

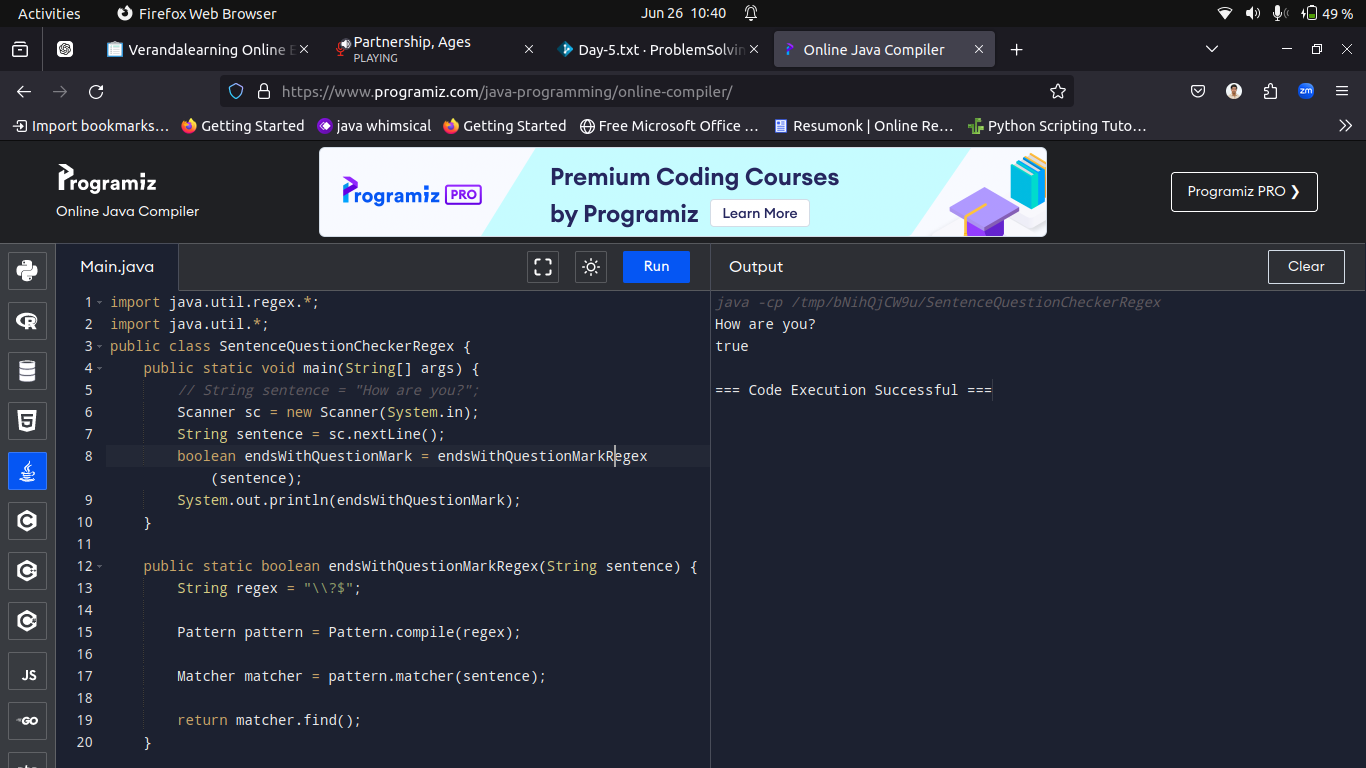
Pattern pattern = Pattern.compile(regex);

Matcher matcher = pattern.matcher(sentence);

return matcher.find();

}

}



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

Q7.Write a program to find the nth Fibonacci number using dynamic programming.

Input:6

Output:8

Solution:

import java.util.\*;

public class FibonacciDynamicProgramming {

public static void main(String[] args) {

// int n = 6;

Scanner sc = new Scanner(System.in);

int n = sc.nextInt();

int nthFibonacci = findNthFibonacci(n);

System.out.println("The " + n + "th Fibonacci number is: " + nthFibonacci);

}

public static int findNthFibonacci(int n) {

if (n <= 1) {

return n;

}

int[] fib = new int[n + 1];

fib[0] = 0;

fib[1] = 1;

for (int i = 2; i <= n; i++) {

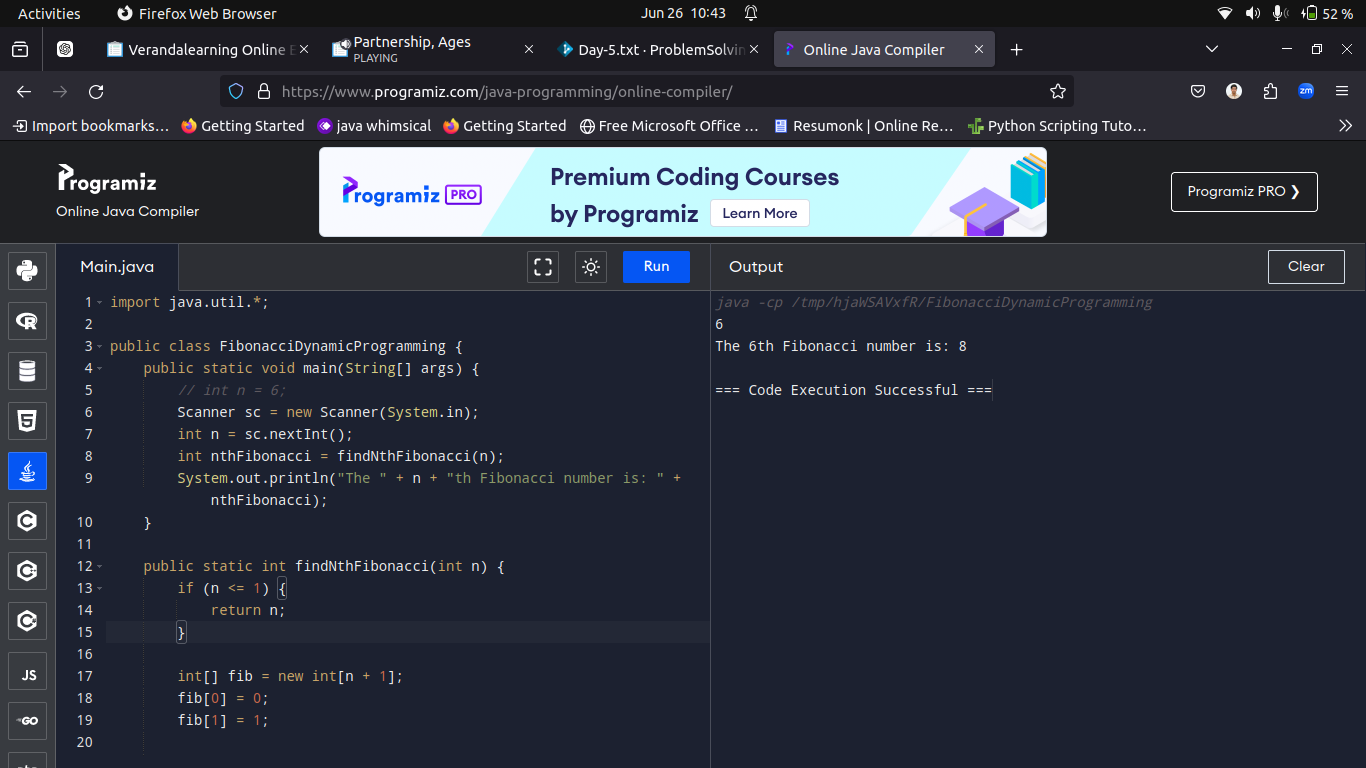
fib[i] = fib[i - 1] + fib[i - 2];

}

return fib[n];

}

}



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

Q8.Write a program to check if a given string contains only digits

Sample Input:

12345

Output:

true

Solution:

import java.util.regex.\*;

import java.util.\*;

public class DigitCheckerRegex {

public static void main(String[] args) {

// String input = "12345";

Scanner sc = new Scanner(System.in);

String input = sc.next();

boolean containsOnlyDigits = checkIfAllDigitsRegex(input);

System.out.println(containsOnlyDigits);

}

public static boolean checkIfAllDigitsRegex(String input) {

String regex = "^\\d+$";

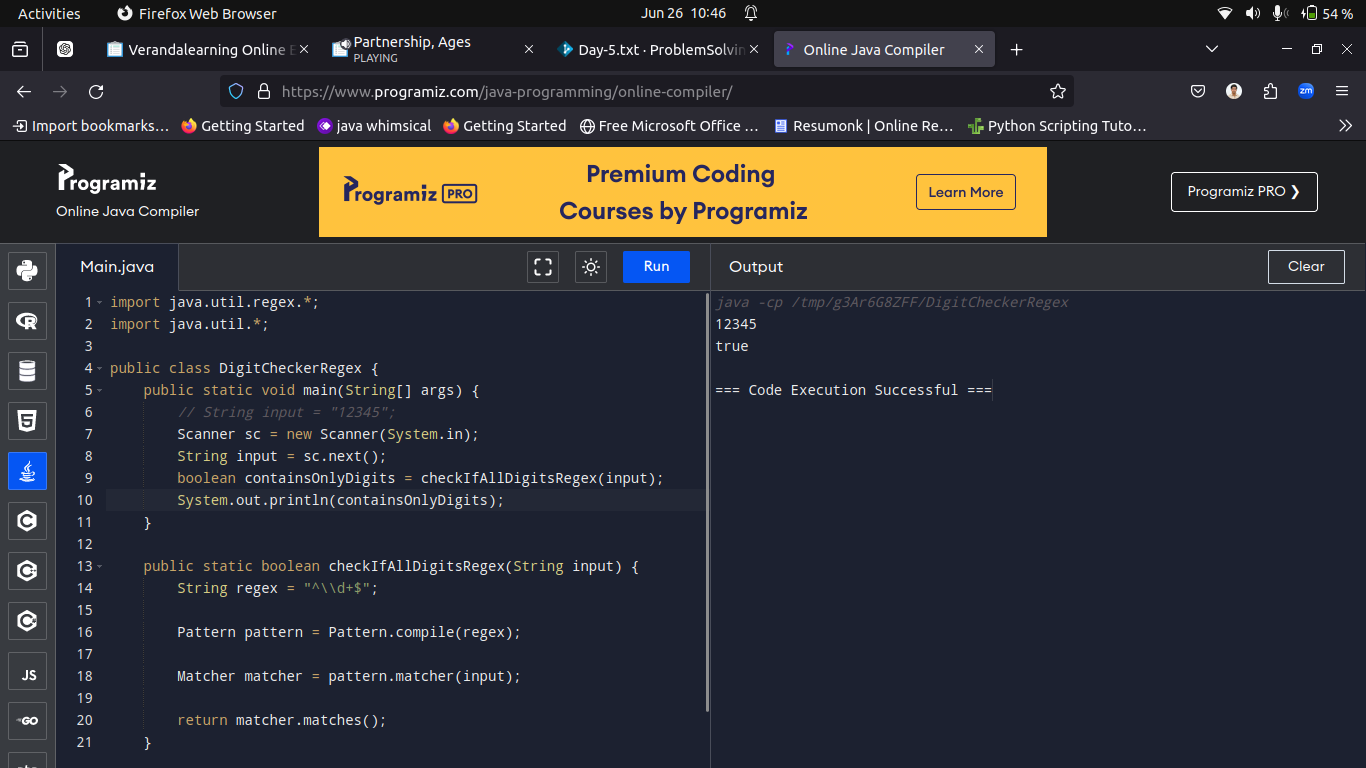
Pattern pattern = Pattern.compile(regex);

Matcher matcher = pattern.matcher(input);

return matcher.matches();

}

}



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

Q9.Write a program to extract all digits from a given string.

Sample Input:

abc123xyz

Sample Output:

123

Solution:

import java.util.regex.\*;

import java.util.ArrayList;

import java.util.List;

import java.util.\*;

public class DigitExtractor {

public static void main(String[] args) {

// String input = "abc123xyz234";

Scanner sc = new Scanner(System.in);

String input = sc.next();

List<Integer> digits = extractDigits(input);

System.out.println("Digits extracted from input: " + digits);

}

public static List<Integer> extractDigits(String input) {

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

Pattern pattern = Pattern.compile("\\d+");

Matcher matcher = pattern.matcher(input);

while (matcher.find()) {

String digitStr = matcher.group();

int digit = Integer.parseInt(digitStr);

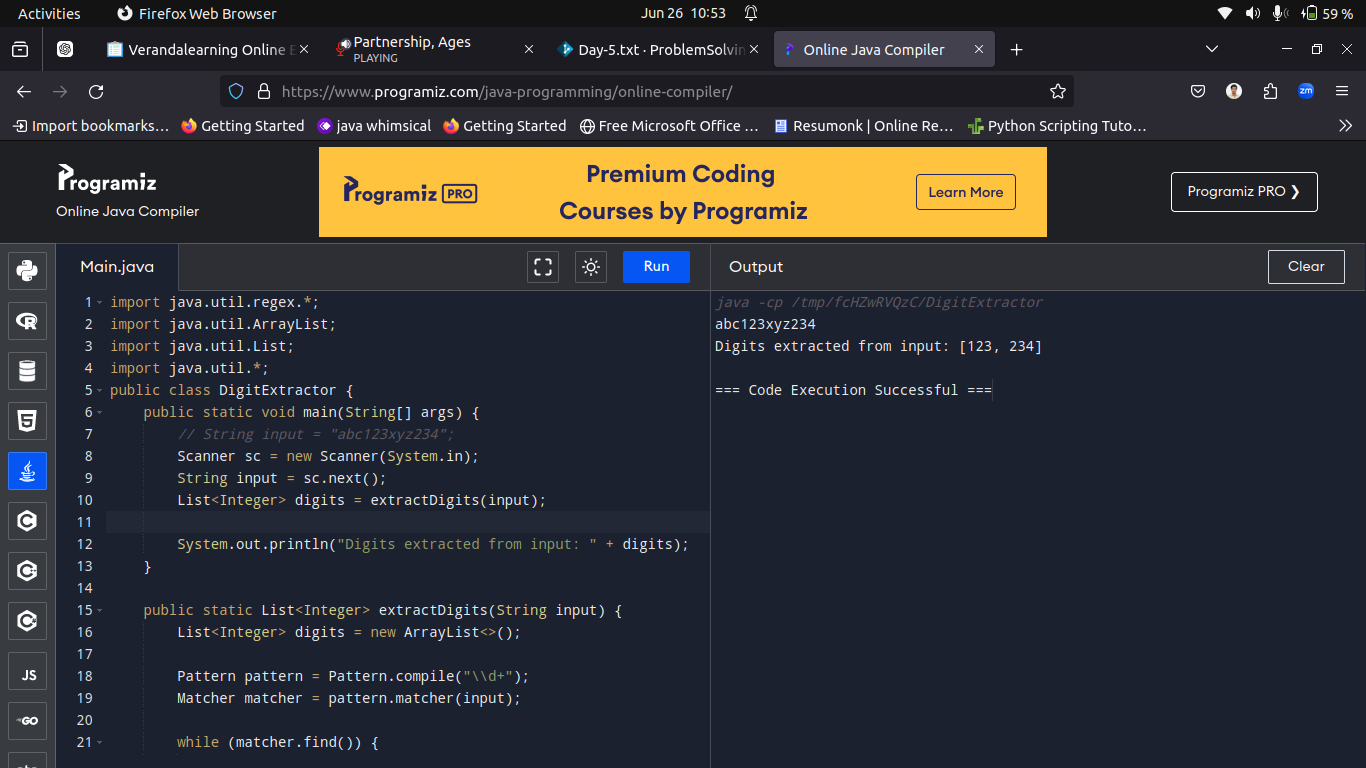
digits.add(digit);

}

return digits;

}

}



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

Q10.Write a program to replace all vowels in a given string with the character '\*'.

Sample input:

guvi geek

Sample Output:

g\*v\* g\*\*k

Solution:

import java.util.\*;

public class VowelReplacement {

public static void main(String[] args) {

// String input = "guvi geek";

Scanner sc = new Scanner(System.in);

String input = sc.nextLine();

String replacedString = replaceVowels(input);

System.out.println("Input: " + input);

System.out.println("Output: " + replacedString);

}

public static String replaceVowels(String input) {

String regex = "[aeiouAEIOU]";

String replacedString = input.replaceAll(regex, "\*");

return replacedString;

}

}

