|  |  |
| --- | --- |
|  | ***Patuakhali Science and Technology University*** |

Assignment on

***“***w3resourse Problem (151-249) solve***”***

Course Code: CCE-122

Course Title: Object Oriented Programming

Level - I; Semester - II

|  |
| --- |
| **Submitted By**  **Name: M.D. Sakibul Islam Shovon**  **ID:** 2302056  **REG:** 11834  **Session:** 2023-2024  Faculty of Computer Science and Engineering |

|  |
| --- |
| **Submitted To**  **Prof. Dr. Md. Samsuzzaman**  Professor of Computer and Communication Engineering Department  Faculty of Computer Science and Engineering |

**w3resourse Problem (151-249) solve:**

151. Evaluate Expressions

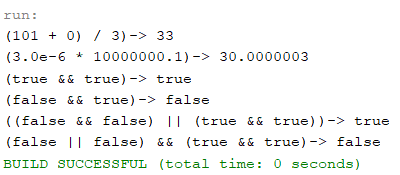
Write a Java program to find the value of a specified expression.

a) 101 + 0) / 3  
b) 3.0e-6 \* 10000000.1  
c) true && true  
d) false && true  
e) (false && false) || (true && true)  
f) (false || false) && (true && true)

**Code:**

|  |
| --- |
| package Lab\_p2;  public class P\_151 {  public static void main(String[] args){  System.out.println("(101 + 0) / 3)-> " + ((101 + 0) / 3));  System.out.println("(3.0e-6 \* 10000000.1)-> " + (3.0e-6 \* 10000000.1));  System.out.println("(true && true)-> " + (true && true));  System.out.println("(false && true)-> " + (false && true));  System.out.println("((false && false) || (true && true))-> " + ((false && false) || (true && true)));  System.out.println("(false || false) && (true && true)-> " + ((false || false) && (true && true)));  }  } |

**Output:**



152. Check Four Numbers Equal

Write a Java program that accepts four integers from the user and prints equal if all four are equal, and not equal otherwise.

*Sample Output*:  
Input first number: 25  
Input second number: 37  
Input third number: 45  
Input fourth number: 23  
Numbers are not equal!

**Code:**

|  |
| --- |
| package Lab\_p2;  import java.util.Scanner;  public class P\_152 {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  System.out.print("Input first number: ");  int num1 = scanner.nextInt();  System.out.print("Input second number: ");  int num2 = scanner.nextInt();  System.out.print("Input third number: ");  int num3 = scanner.nextInt();  System.out.print("Input fourth number: ");  int num4 = scanner.nextInt();  if (num1 == num2 && num2 == num3 && num3 == num4) {  System.out.println("Numbers are equal!");  } else {  System.out.println("Numbers are not equal!");  }  }  } |

**Output:**

A screenshot of a computer code

AI-generated content may be incorrect.

153. Test Doubles Between 0 and 1

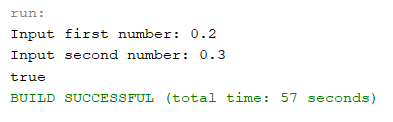
Write a Java program that accepts two double variables and test if both strictly between 0 and 1 and false otherwise.

*Sample Output:*  
Input first number: 5  
Input second number: 1  
false

**Code:**

|  |
| --- |
| package Lab\_p2;  import java.util.Scanner;  public class P\_153 {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  System.out.print("Input first number: ");  double num1 = scanner.nextDouble();  System.out.print("Input second number: ");  double num2 = scanner.nextDouble();  boolean isInRange = (num1 > 0 && num1 < 1) && (num2 > 0 && num2 < 1);  System.out.println(isInRange);  }  } |

**Output:**



154. Print Boolean Matrix

Write a Java program to print the contents of a two-dimensional Boolean array where t represents true and f represents false.

Sample array:  
array = {{true, false, true},  
{false, true, false}};  
*Expected Output* :  
t f t  
f t f

**Code:**

|  |
| --- |
| package Lab\_p2;  public class P\_154 {  public static void main(String[] args) {  boolean[][] array = {  {true, false, true},  {false, true, false}  };  printBooleanArray(array);  }  public static void printBooleanArray(boolean[][] array) {  for (int i = 0; i < array.length; i++) {  for (int j = 0; j < array[i].length; j++) {  System.out.print(array[i][j] ? "t " : "f ");  }  System.out.println();  }  }  } |

**Output:**

A white background with black text

AI-generated content may be incorrect.

155. Transpose 2D Array

Write a Java program to print an array after changing the rows and columns of a two-dimensional array.

Original Array:  
10 20 30  
40 50 60  
After changing the rows and columns of the said array:10 40  
20 50  
30 60

**Code:**

|  |
| --- |
| package Lab\_p2;  public class P\_155 {  public static void main(String[] args) {  int[][] originalArray = {  {10, 20, 30},  {40, 50, 60}  };  int rows = originalArray.length;  int cols = originalArray[0].length;  int[][] transposedArray = new int[cols][rows];  for (int i = 0; i < rows; i++) {  for (int j = 0; j < cols; j++) {  transposedArray[j][i] = originalArray[i][j];  }  }  System.out.println("Transpose array: ");  for (int j = 0; j < cols; j++) {  for (int i = 0; i < rows; i++) {  System.out.print(transposedArray[j][i] + " ");  }  System.out.println();  }  }  } |

**Output:**

A white background with black text

AI-generated content may be incorrect.

158. 2D Prime Matrix

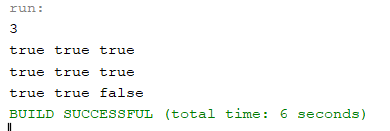
Write a Java program to create a two-dimensional array (m x m) A[][] such that A[i][j] is false if I and j are prime otherwise A[i][j] becomes true.

*Sample Output:*  
true true true  
true true true  
true true false

**Code:**

|  |
| --- |
| package Lab\_p2;  public class P\_155 {  public static void main(String[] args) {  int[][] originalArray = {  {10, 20, 30},  {40, 50, 60}  };  int rows = originalArray.length;  int cols = originalArray[0].length;  int[][] transposedArray = new int[cols][rows];  for (int i = 0; i < rows; i++) {  for (int j = 0; j < cols; j++) {  transposedArray[j][i] = originalArray[i][j];  }  }  System.out.println("Transpose array: ");  for (int j = 0; j < cols; j++) {  for (int i = 0; i < rows; i++) {  System.out.print(transposedArray[j][i] + " ");  }  System.out.println();  }  }  } |

**Output:**



159. Find K Largest Elements

Write a Java program to find the k largest elements in a given array. Elements in the array can be in any order.

*Expected Output:*  
Original Array:  
[1, 4, 17, 7, 25, 3, 100]  
3 largest elements of the said array are:  
100 25 17

**Code:**

|  |
| --- |
| import java.util.Arrays;  import java.util.Scanner;  public class P\_159 {  public static void Largest(int[] arr, int k) {  Arrays.sort(arr);  for (int i = arr.length - 1; i >= arr.length - k; i--) {  System.out.print(arr[i] + " ");  }  System.out.println("");  }  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  System.out.print("Enter the size of the array: ");  int size = scanner.nextInt();  int arr[] = new int[size];  System.out.println("Enter the elements of the array:");  for (int i = 0; i < size; i++) {  arr[i] = scanner.nextInt();  }  System.out.print("Enter the value of k: ");  int k = scanner.nextInt();  if (k <= 0 || k > size) {  System.out.println("Invalid value of k. It should be between 1 and " + size);  return;  }  Largest(arr, k);  scanner.close();  }  } |

**Output:**

A white background with black text

AI-generated content may be incorrect.

160. Find K Smallest Elements

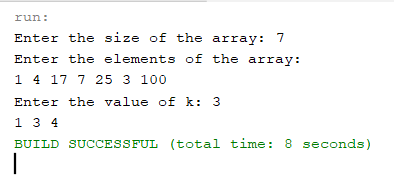
Write a Java program to find the k smallest elements in a given array. Elements in the array can be in any order.

*Expected Output:*  
Original Array:  
[1, 4, 17, 7, 25, 3, 100]  
3 largest elements of the said array are:  
100 25 17

**Code:**

|  |
| --- |
| import java.util.Arrays;  import java.util.Scanner;  public class P\_160 {  public static void smalest(int[] arr, int k) {  Arrays.sort(arr);  for (int i = 0; i <=k-1; i++) {  System.out.print(arr[i] + " ");  }  System.out.println("");  }  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  System.out.print("Enter the size of the array: ");  int size = scanner.nextInt();  int arr[] = new int[size];  System.out.println("Enter the elements of the array:");  for (int i = 0; i < size; i++) {  arr[i] = scanner.nextInt();  }  System.out.print("Enter the value of k: ");  int k = scanner.nextInt();  if (k <= 0 || k > size) {  System.out.println("Invalid value of k. It should be between 1 and " + size);  return;  }  smalest(arr, k);  scanner.close();  }  } |

**Output:**



161. Kth Smallest and Largest

Write a Java program to find the kth smallest and largest element in a given array. Elements in the array can be in any order.

*Expected Output:*  
Original Array:  
[1, 4, 17, 7, 25, 3, 100]  
K'th smallest element of the said array:  
3  
K'th largest element of the said array:  
25

**Code:**

|  |
| --- |
| package Lab\_p2;  import java.util.Arrays;  import java.util.Scanner;  public class P\_161 {  public static void smalestLargest(int[] arr, int k) {  Arrays.sort(arr);  System.out.println(k + "th smallest element of the said array: " + arr[k-1]);  System.out.println(k + "th smallest element of the said array: " + arr[arr.length-k+1]);  System.out.println("");  }  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  System.out.print("Enter the size of the array: ");  int size = scanner.nextInt();  int arr[] = new int[size];  System.out.println("Enter the elements of the array:");  for (int i = 0; i < size; i++) {  arr[i] = scanner.nextInt();  }  System.out.print("Enter the value of k: ");  int k = scanner.nextInt();  if (k <= 0 || k > size) {  System.out.println("Invalid value of k. It should be between 1 and " + size);  return;  }  smalestLargest(arr, k);  scanner.close();  }  } |

**Output:**

A white background with black text

AI-generated content may be incorrect.

162. Numbers Greater Than Average

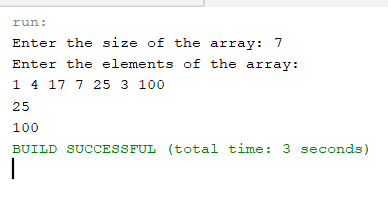
Write a Java program that finds numbers greater than the average of an array.

*Expected Output:*  
Original Array:  
[1, 4, 17, 7, 25, 3, 100]  
The average of the said array is: 22.0  
The numbers in the said array that are greater than the average are:  
25  
100

**Code:**

|  |
| --- |
| package Lab\_p2;  import java.util.Arrays;  import java.util.Scanner;  public class P\_162 {  public static double calavg(int[] array) {  if (array.length == 0) return 0;  int sum = 0;  for (int num : array) {  sum += num;  }  return (double) sum / array.length;  }  public static void greater\_than\_avg(int[] array, double average) {  for (int num : array) {  if (num > average) {  System.out.println(num);  }  }  }  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  System.out.print("Enter the size of the array: ");  int size = scanner.nextInt();  int arr[] = new int[size];  System.out.println("Enter the elements of the array:");  for (int i = 0; i < size; i++) {  arr[i] = scanner.nextInt();  }  double average = calavg(arr);  greater\_than\_avg(arr, average);  }  } |

**Output:**



163. Binary Zeros Count

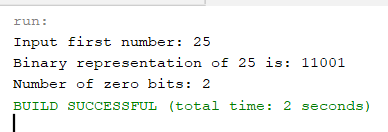
Write a Java program that will accept an integer and convert it into a binary representation. Now count the number of bits equal to zero in this representation.

*Expected Output:*  
Input first number: 25  
Binary representation of 25 is: 11001  
Number of zero bits: 2

**Code:**

|  |
| --- |
| package Lab\_p2;  import java.util.Scanner;  public class P\_163 {  public static void main(String[] args) {  Scanner scan = new Scanner(System.in);    System.out.print("Input first number: ");  int number = scan.nextInt();    String binaryString = Integer.toBinaryString(number);  System.out.println("Binary representation of " + number + " is: " + binaryString);    int zeroCount = countZeroBits(binaryString);  System.out.println("Number of zero bits: " + zeroCount);    }    public static int countZeroBits(String binaryString) {  int count = 0;  for (int i = 0; i < binaryString.length(); i++) {  if (binaryString.charAt(i) == '0') {  count++;  }  }  return count;  }  } |

**Output:**



164. Divide Using Subtraction

Write a Java program to divide the two given integers using the subtraction operator.

*Expected Output:*  
Input the dividend: 625  
Input the divider: 25  
Result: 25.0

**Code:**

|  |
| --- |
| import java.util.Scanner;  import java.util.Scanner;  public class P\_164{  public static void main(String[] args) {  Scanner scan = new Scanner(System.in);    System.out.print("Input the dividend: ");  int dividend = scan.nextInt();    System.out.print("Input the divider: ");  int divider = scan.nextInt();    if (divider == 0) {  System.out.println("Cannot divide by zero!");  } else {  int count = 0;  int remainder = dividend;    while (remainder >= divider) {  remainder -= divider;  count++;  }    System.out.println("Result: " + count);  }    }  } |

**Output:**

A white background with black text

AI-generated content may be incorrect.

165. Move Positives Right

Write a Java program to move every positive number to the right and every negative number to the left of a given array of integers.

*Expected Output:*  
Original array: [-2, 3, 4, -1, -3, 1, 2, -4, 0]  
Result: [-4, -3, -2, -1, 0, 1, 2, 3, 4]

**Code:**

|  |
| --- |
| package Lab\_p2;  import java.util.Arrays;  import java.util.Scanner;  public class P\_165 {  public static void main(String[] args) {  Scanner scan = new Scanner(System.in);  System.out.println("Enter the array number: ");  int n = scan.nextInt();  System.out.println("Enter the elements of the array:");  int arr[] = new int[n];  for (int i = 0; i < n; i++) arr[i] = scan.nextInt();  Arrays.sort(arr);  System.out.println(Arrays.toString(arr));  }  } |

**Output:**

A white background with black text

AI-generated content may be incorrect.

166. Integer to String Format

Write a Java program to transform a given integer into String format.

*Expected Output:*  
Input an integer: 35  
String format of the said integer: 35

**Code:**

|  |
| --- |
| import java.util.Scanner;  public class p\_166 {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);    System.out.print("Input an integer: ");  int number = scanner.nextInt();    System.out.println("String format of the said integer: " + Integer.toString(number));    }  } |

**Output:**

A text on a white background

AI-generated content may be incorrect.

167. Move Zeros to Right

Write a Java program to move every zero to the right side of a given array of integers.

Original array: [0, 3, 4, 0, 1, 2, 5, 0]  
Result: [3, 4, 1, 2, 5, 0, 0, 0]

**Code:**

|  |
| --- |
| package Lab\_p2;  import java.util.Arrays;  import java.util.Scanner;  public class P\_167 {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  System.out.print("Enter the size of the array: ");  int size = scanner.nextInt();  int arr[] = new int[size];  System.out.println("Enter the elements of the array:");  for (int i = 0; i < size; i++) arr[i] = scanner.nextInt();  int[] result = moveZeros(arr);    System.out.println("Result: " + Arrays.toString(result));  }  public static int[] moveZeros(int[] nums) {  int[] result = new int[nums.length];  int index = 0;  for (int num : nums) {  if (num != 0) {  result[index++] = num;  }  }  return result;  }  } |

**Output:**

A white background with black text

AI-generated content may be incorrect.

168. Multiply Without Operator

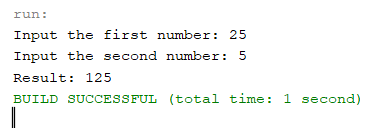
Write a Java program to multiply two given integers without using the multiply operator (\*).

Input the first number: 25  
Input the second number: 5  
Result: 125

**Code:**

|  |
| --- |
| import java.util.Scanner;  import java.util.Scanner;  public class P\_168{  public static void main(String[] args) {  Scanner scan = new Scanner(System.in);  System.out.print("Input the first number: ");  int first = scan.nextInt();  System.out.print("Input the second number: ");  int second = scan.nextInt();  int check = second;  int multiple = 0;  while (check>0) {  multiple += first ;  check--;  }  System.out.println("Result: " + multiple);  }  } |

**Output:**



169. Reverse Sentence Without Words

Write a Java program to reverse a sentence (assume a single space between two words) without reverse every word.

Input a string: The quick brown fox jumps over the lazy dog  
Result: dog lazy the over jumps fox brown quick The

**Code:**

|  |
| --- |
| package Lab\_p2;  import java.util.Scanner;  public class P\_169 {  public static void main(String[] args) {  Scanner scan = new Scanner(System.in);  System.out.print("Input the string : ");  String s = scan.nextLine();    System.out.print("The reverse string is: ");  for(int i=s.length()-1; i>=0; i--) System.out.print(s.charAt(i));  System.out.println("");  }  } |

**Output:**

A close-up of a white background

AI-generated content may be incorrect.

171. Test Substring in String

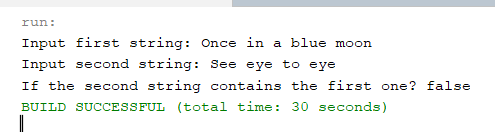
Write a Java program to accept two strings and test if the second string contains the first one.

Input first string: Once in a blue moon  
Input second string: See eye to eye  
If the second string contains the first one? false

**Code:**

|  |
| --- |
| package Lab\_p2;  import java.util.Scanner;  public class P\_171 {  public static void main(String[] args) {  Scanner scanner = new Scanner(System.in);  System.out.print("Input first string: ");  String firstString = scanner.nextLine();  System.out.print("Input second string: ");  String secondString = scanner.nextLine();  boolean contains = secondString.contains(firstString);  System.out.println("If the second string contains the first one? " + contains);  }  } |

**Output:**



172. Write a Java program to get the number of elements in a given array of integers that are smaller than the integer in another given array of integers.

*Expected Output:*  
0  
3  
7

Code:

|  |
| --- |
| // Importing necessary Java utilities  import java.util.ArrayList;  import java.util.Arrays;  // Defining a class named Code  public class Code {    // The main method of the program  public static void main(String[] args) {  // Initializing arrays for main and query data  int[] main\_arra = {1, 2, 3, 4, 5, 6, 7, 8};  int[] query\_arra = {1, 4, 8};    // Getting the result by counting smaller numbers from the main array for query elements  ArrayList<Integer> result = count\_smaller\_number(main\_arra, query\_arra);    // Displaying the result  for (int i = 0; i < result.size(); i++) {  System.out.println(result.get(i));  }  }    // Method to count smaller numbers in the main array for query elements  public static ArrayList<Integer> count\_smaller\_number(int[] main\_arra, int[] query\_arra) {  // Initializing an ArrayList to store the result  ArrayList<Integer> result = new ArrayList<>();    // Sorting the main array in ascending order  Arrays.sort(main\_arra);    // Looping through the query array elements  for (int i = 0; i < query\_arra.length; i++) {  // Adding the count of smaller numbers for each query element to the result ArrayList  result.add(temp(main\_arra, query\_arra[i]));  }    return result; // Returning the result ArrayList  }    // Helper method to count smaller numbers in the main array  private static int temp(int[] main\_arra, int num) {  int ctr = 0; // Counter to track the number of smaller elements    // Looping through the main array  for (int i = 0; i < main\_arra.length; i++) {  // Checking if the current element in the main array is smaller than the given number  if (main\_arra[i] < num) {  ctr++; // Incrementing the counter for smaller numbers  } else {  break; // Exiting the loop if the current element is greater than or equal to the given number  }  }    return ctr; // Returning the count of smaller numbers  }  } |

Output:

A white rectangular object with a black border

Description automatically generated

173. Write a Java program to find the median of the numbers inside the window (size k) at each step in a given array of integers with duplicate numbers. Move the window to the array start.

Sample Output:  
{|1, 2, 3|, 4, 5, 6, 7, 8, 8} -> Return median 2  
{1, |2, 3, 4|, 5, 6, 7, 8, 8} -> Return median 3  
{1, 2, |3, 4, 5|, 6, 7, 8, 8} -> Return median 4  
{1, 2, 3, |4, 5, 6|, 7, 8, 8} -> Return median 5  
{1, 2, 3, 4, |5, 6, 7|, 8, 8} -> Return median 6  
{1, 2, 3, 4, 5, |6, 7, 8|, 8} -> Return median 7  
{1, 2, 3, 4, 5, 6, |7, 8, 8|} -> Return median 8  
Result array {2, 3, 4, 5, 6, 7, 8}

*Expected Output:*

Original array: [1, 2, 3, 4, 5, 6, 7, 8, 8]

Value of k: 3

Result:

2

3

4

5

6

7

8

Code:

|  |
| --- |
| // Importing necessary Java utilities  import java.util.\*;  import java.util.Arrays;  import java.util.LinkedList;  // Defining a class named Code  public class Code {    // The main method of the program  public static void main(String[] args) {  // Initializing an array and window size 'k'  int[] main\_array = {1, 2, 3, 4, 5, 6, 7, 8, 8};  int k = 3;    // Displaying the original array and value of 'k'  System.out.println("\nOriginal array: " + Arrays.toString(main\_array));  System.out.println("\nValue of k: " + k);  System.out.println("\nResult: ");    // Getting the result of the median sliding window operation  ArrayList<Integer> result = median\_slide\_window(main\_array, k);    // Displaying the result  for (int i = 0; i < result.size(); i++) {  System.out.println(result.get(i));  }  }    // Method to compute the median in a sliding window of size 'k'  public static ArrayList<Integer> median\_slide\_window(int[] main\_array, int k) {  ArrayList<Integer> result = new ArrayList<>();    // If 'k' is 0 or greater than the length of the array, return an empty result  if (k == 0 || main\_array.length < k) {  return result;  }    // PriorityQueues to store elements on the right and left side of the window  PriorityQueue<Integer> right\_num = new PriorityQueue<>(k);  PriorityQueue<Integer> left\_num = new PriorityQueue<>(k, Collections.reverseOrder());  // Adding elements to the queues for initial window  for (int i = 0; i < k - 1; ++i) {  add(right\_num, left\_num, main\_array[i]);  }  // Sliding the window and computing median  for (int i = k - 1; i < main\_array.length; ++i) {  add(right\_num, left\_num, main\_array[i]);  int median = compute\_median(right\_num, left\_num);  result.add(median);  remove(right\_num, left\_num, main\_array[i - k + 1]);  }    return result; // Returning the result containing medians of the sliding window  }  // Method to compute the median from the PriorityQueues  private static int compute\_median(PriorityQueue<Integer> right\_num, PriorityQueue<Integer> left\_num) {  if (left\_num.isEmpty() && right\_num.isEmpty()) {  return 0; // Return 0 if both queues are empty  }    // Balancing the queues to get the median  while (left\_num.size() < right\_num.size()) {  left\_num.add(right\_num.poll());  }  while (left\_num.size() - right\_num.size() > 1) {  right\_num.add(left\_num.poll());  }    return left\_num.peek(); // Returning the median element  }  // Method to add elements to the PriorityQueues maintaining the order  private static void add(PriorityQueue<Integer> right\_num, PriorityQueue<Integer> left\_num, int num) {  if (left\_num.isEmpty() && right\_num.isEmpty()) {  left\_num.add(num);  return;  } else {  if (num <= compute\_median(right\_num, left\_num)) {  left\_num.add(num);  } else {  right\_num.add(num);  }  }  }  // Method to remove elements from the PriorityQueues  private static void remove(PriorityQueue<Integer> right\_num, PriorityQueue<Integer> left\_num, int num) {  if (num <= compute\_median(right\_num, left\_num)) {  left\_num.remove(num);  } else {  right\_num.remove(num);  }  }  } |

Output:

A white background with blue text

Description automatically generated

174. Write a Java program to find the maximum number inside the number in the window (size k) at each step in a given array of integers with duplicate numbers. Move the window to the top of the array.

Sample output:  
{|1, 2, 3|, 4, 5, 6, 7, 8, 8} -> Return maximum 3  
{1, |2, 3, 4|, 5, 6, 7, 8, 8} -> Return maximum 4  
{1, 2, |3, 4, 5|, 6, 7, 8, 8} -> Return maximum 5  
{1, 2, 3, |4, 5, 6|, 7, 8, 8} -> Return maximum 6  
{1, 2, 3, 4, |5, 6, 7|, 8, 8} -> Return maximum 7  
{1, 2, 3, 4, 5, |6, 7, 8|, 8} -> Return maximum 8  
{1, 2, 3, 4, 5, 6, |7, 8, 8|} -> Return maximum 8  
Result array {3, 4, 5, 6, 7, 8, 8}

*Expected Output:*

Original array: [1, 2, 3, 4, 5, 6, 7, 8, 8]

Value of k: 3

Result:

2

3

4

5

6

7

8

Code:

|  |
| --- |
| // Import necessary classes from java.util package  import java.util.\*;  import java.util.Arrays;  import java.util.LinkedList;  // Main class to demonstrate max sliding window  public class Main {  // Main method to execute the sliding window algorithm  public static void main(String[] args) {  // Sample array and value of k for testing  int[] main\_array = {1, 2, 3, 4, 5, 6, 7, 8, 8};  int k = 3;  // Display the original array and the value of k  System.out.println("\nOriginal array: " + Arrays.toString(main\_array));  System.out.println("\nValue of k: " + k);  System.out.println("\nResult: ");  // Call the method to find maximums in the sliding window  ArrayList result = max\_slide\_window(main\_array, k);  // Display the result  for (int i = 0; i < result.size(); i++) {  System.out.println(result.get(i));  }  }  // Method to find maximums in a sliding window  public static ArrayList max\_slide\_window(int[] main\_array, int k) {  // Initialize an ArrayList to store the result  ArrayList rst\_arra = new ArrayList();  // Checking for invalid inputs  if (main\_array == null || main\_array.length == 0 || k < 0) {  return rst\_arra;  }  // Using a Deque to store indexes of elements  Deque<Integer> deque\_num = new LinkedList<>();  // Processing the first k elements separately  for (int i = 0; i < k; i++) {  // Removing smaller elements from the Deque  while (!deque\_num.isEmpty() && main\_array[deque\_num.peekLast()] <= main\_array[i]) {  deque\_num.pollLast();  }  deque\_num.offerLast(i); // Adding the current index to the Deque  }  // Processing the rest of the elements  for (int i = k; i < main\_array.length; i++) {  rst\_arra.add(main\_array[deque\_num.peekFirst()]); // Adding the maximum from the window to result  // Removing elements that are out of the window range  if (!deque\_num.isEmpty() && deque\_num.peekFirst() <= i - k) {  deque\_num.pollFirst();  }  // Removing smaller elements from the Deque  while (!deque\_num.isEmpty() && main\_array[deque\_num.peekLast()] <= main\_array[i]) {  deque\_num.pollLast();  }  deque\_num.offerLast(i); // Adding the current index to the Deque  }  rst\_arra.add(main\_array[deque\_num.peekFirst()]); // Adding the maximum of the last window  return rst\_arra; // Returning the result ArrayList containing maximums  }  } |

Output:

A screenshot of a computer

Description automatically generated

175. Write a Java program to delete a specified node in the middle of a singly linked list.

Sample Singly linked list: 10->20->30->40->50  
Delete the fourth node i.e. 40  
Result: 10->20->30->50  
*Expected Output:*

Original Linked list:

10->20->30->40->50

After deleting the fourth node, Linked list becomes:

10->20->30->50

Code:

|  |
| --- |
| // Importing necessary Java utilities  import java.util.\*;  import java.util.Arrays;  import java.util.LinkedList;  // ListNode class definition representing each node of the linked list  class ListNode {  int val;  ListNode next;  // Constructor to initialize the ListNode  ListNode(int val) {  this.val = val;  this.next = null;  }  }  // Main class Code  public class Code {  // Initializing the head of the linked list with a node containing value 10  public static ListNode head = new ListNode(10);  // Main method  public static void main(String[] args) {  // Creating a linked list with nodes containing values 20, 30, 40, 50  head.next = new ListNode(20);  head.next.next = new ListNode(30);  head.next.next.next = new ListNode(40);  head.next.next.next.next = new ListNode(50);  ListNode p = head; // Creating a reference 'p' to the head node  System.out.println("Original Linked list:");  printList(p); // Printing the original linked list  System.out.println("\nAfter deleting the fourth node, Linked list becomes:");  deleteNode(head.next.next.next); // Deleting the fourth node in the list  p = head; // Updating reference 'p' to the head node after deletion  printList(p); // Printing the updated linked list  }  // Method to delete a node from the linked list  public static void deleteNode(ListNode node) {  // Check if the node to be deleted is not the last node in the list  if (node.next != null) {  int temp = node.val;  node.val = node.next.val;  node.next.val = temp;  node.next = node.next.next; // Skip the next node effectively deleting the current node  } else {  // If the node to be deleted is the last node, traverse to the previous node and delete it  ListNode p = head;  while (p.next.val != node.val) {  p = p.next;  }  p.next = null; // Set the next of the previous node to null  }  }  // Method to print the linked list  static void printList(ListNode p) {  while (p != null) {  System.out.print(p.val); // Printing the value of the current node  if (p.next != null) {  System.out.print("->"); // Adding an arrow for non-last nodes  }  p = p.next; // Move to the next node  }  }  } |

Output:

A white background with blue text

Description automatically generated

176. Write a Java program that partitions an array of integers into even and odd numbers.

*Expected Output*

Original array: [7, 2, 4, 1, 3, 5, 6, 8, 2, 10]

After partition the said array becomes: [10, 2, 4, 2, 8, 6, 5, 3, 1, 7]

Code:

|  |
| --- |
| // Importing necessary Java utilities  import java.util.\*;  // Main class Code  public class Code {    // Main method  public static void main(String[] args) {  int[] nums = {7, 2, 4, 1, 3, 5, 6, 8, 2, 10};    // Printing the original array  System.out.println("Original array: " + Arrays.toString(nums));    // Calling the partitionArray2 method to partition the array  int[] result = partitionArray2(nums);    // Printing the resulting array after partitioning  System.out.println("After partition the said array becomes: " + Arrays.toString(result));  }  // Method to partition the array based on odd and even numbers  public static int[] partitionArray2(int[] nums) {  int i = 0; // Initializing pointer i to the start of the array  int j = nums.length - 1; // Initializing pointer j to the end of the array    // Looping until pointers i and j meet or cross each other  while (i < j) {  // Moving pointer i until it finds an odd number  while (nums[i] % 2 == 0) {  i++;  }    // Moving pointer j until it finds an even number  while (nums[j] % 2 != 0) {  j--;  }    // Swapping the odd and even numbers if i is less than j  if (i < j) {  int temp = nums[i];  nums[i] = nums[j];  nums[j] = temp;  }  }    // Returning the partitioned array  return nums;  }  } |

Output:

A close-up of a computer screen

Description automatically generated

177. Write a Java program to get an updated binary tree with the same structure and value as a given binary tree.

*Expected Output:*

Original Treenode:

4

5

2

3

1

Clone of the said Treenode:

4

5

2

3

1

Code:

|  |
| --- |
| // Importing necessary Java utilities  import java.util.\*;  // Main class Code  public class Code {  // Main method  public static void main(String[] args) {  // Creating TreeNode instances  TreeNode t1 = new TreeNode(1);  TreeNode t2 = new TreeNode(2);  TreeNode t3 = new TreeNode(3);  TreeNode t4 = new TreeNode(4);  TreeNode t5 = new TreeNode(5);  // Creating a tree structure  t1.left = t2;  t1.right = t3;  t2.left = t4;  t2.right = t5;  // Printing the original TreeNode  System.out.println("Original Treenode:");  traverseTree(t1);  // Cloning the TreeNode and printing the clone  System.out.println("\nClone of the said Treenode:");  TreeNode result = cloneTree(t1);  traverseTree(result);  }  // Method to clone a given TreeNode  public static TreeNode cloneTree(TreeNode root) {  // Checking if the root is null  if (root == null) {  return null;  }  // Creating a duplicate TreeNode with the same value as the original root  TreeNode dup = new TreeNode(root.val);  // Recursively cloning left and right subtrees  dup.left = cloneTree(root.left);  dup.right = cloneTree(root.right);  return dup; // Returning the cloned TreeNode  }  // Method to traverse the TreeNode in post-order traversal (Left, Right, Root)  private static void traverseTree(TreeNode root) {  // Checking if the root is not null  if (root != null) {  // Traversing the left subtree  traverseTree(root.left);  // Traversing the right subtree  traverseTree(root.right);  // Printing the value of the current TreeNode  System.out.println(root.val);  }  }  }  // Definition of TreeNode class  class TreeNode {  public int val;  public TreeNode left, right;  // Constructor to initialize TreeNode with a value  public TreeNode(int val) {  this.val = val;  this.left = this.right = null;  }  } |

Output:

A screenshot of a computer

Description automatically generated

178. Write a Java program to find the longest increasing continuous subsequence in a given array of integers.

*Expected Output:*

Original array: [10, 11, 12, 13, 14, 7, 8, 9, 1, 2, 3]

Size of longest increasing continuous subsequence: 5

Code:

|  |
| --- |
| // Importing necessary Java utilities  import java.util.\*;  // Main class Code  public class Code {  // Main method  public static void main(String[] args) {  // Initializing an array of integers  int[] nums = { 10, 11, 12, 13, 14, 7, 8, 9, 1, 2, 3 };    // Printing the original array  System.out.println("Original array: " + Arrays.toString(nums));    // Finding the size of the longest increasing continuous subsequence and printing it  System.out.println("Size of longest increasing continuous subsequence: " + longest\_seq(nums));  }  // Method to find the size of the longest increasing continuous subsequence  public static int longest\_seq(int[] nums) {  int max\_sequ = 0; // Initializing the variable to hold the maximum sequence length    // Handling the case when the array contains only one element  if (nums.length == 1)  return 1; // If only one element is present, the longest sequence is of length 1  // Looping through the array to find the longest increasing or decreasing sequence  for (int i = 0; i < nums.length - 1; i++) {  int ctr = 1; // Counter to track the sequence length  int j = i; // Initializing j to the current index i    // Checking for an increasing sequence  if (nums[i + 1] > nums[i]) {  while (j < nums.length - 1 && nums[j + 1] > nums[j]) {  ctr++; // Incrementing the counter for each increasing element  j++;  }  }  // Checking for a decreasing sequence  else if (nums[i + 1] < nums[i]) {  while (j < nums.length - 1 && nums[j + 1] < nums[j]) {  ctr++; // Incrementing the counter for each decreasing element  j++;  }  }    // Updating the maximum sequence length encountered so far  if (ctr > max\_sequ) {  max\_sequ = ctr;  }    // Moving the index i ahead by the sequence length minus 2 to avoid rechecking elements  i += ctr - 2;  }    return max\_sequ; // Returning the size of the longest sequence found  }  } |

Output:

A close-up of a number

Description automatically generated

179. Write a Java program to add one to a positive number represented as an array of digits.

Sample array: [9, 9, 9, 9] which represents 9999  
Output: [1, 0, 0, 0, 0].

*Expected Output:*

Original array: [9, 9, 9, 9]

Array of digits: [1, 0, 0, 0, 0]

Code:

|  |
| --- |
| // Importing necessary Java utilities  import java.util.\*;  // Main class Code  public class Code {  // Main method  public static void main(String[] args) {  // Initializing an array of integers  int[] nums = {9, 9, 9, 9};    // Printing the original array  System.out.println("Original array: " + Arrays.toString(nums));    // Printing the array of digits after adding one to the input array  System.out.println("Array of digits: " + Arrays.toString(plus\_One\_digit(nums)));  }    // Method to add one to the last digit of the input array  public static int[] plus\_One\_digit(int[] digits\_nums) {  // Looping through the array from the end to the start  for (int i = digits\_nums.length - 1; i > -1; --i) {  // Checking if the digit is not 9  if (digits\_nums[i] != 9) {  digits\_nums[i] += 1; // Incrementing the digit by 1    // Setting the digits after the incremented digit to 0  for (int j = i + 1; j < digits\_nums.length; ++j) {  digits\_nums[j] = 0;  }    return digits\_nums; // Returning the updated array  }  }    // If all digits are 9, creating a new array with an additional digit for carrying over 1  int[] result = new int[digits\_nums.length + 1];  result[0] = 1; // Setting the first digit to 1    return result; // Returning the new array with the carried over 1  }  } |

Output:

A number and symbols on a white background

Description automatically generated

180. Write a Java program to swap two adjacent nodes in a linked list.

*Expected Output:*

Original Linked list:

10->20->30->40->50

After swiping Linked list becomes:

20->10->40->30->50

Code:

|  |
| --- |
| // Importing necessary Java utilities  import java.util.\*;  // Main class Code  public class Code {  // Main method  public static void main(String[] args) {  // Creating a linked list  ListNode l = new ListNode(10);  l.next = new ListNode(20);  l.next.next = new ListNode(30);  l.next.next.next = new ListNode(40);  l.next.next.next.next = new ListNode(50);    // Printing original linked list  System.out.println("\nOriginal Linked list:");  printList(l);    // Swapping pairs of nodes in the linked list  ListNode p = swap\_Pairs(l);    // Printing linked list after swapping pairs  System.out.println("\n\nAfter swapping, Linked list becomes:");  printList(p);  }    // Method to swap pairs of nodes in a linked list  public static ListNode swap\_Pairs(ListNode head) {  ListNode temp = new ListNode(0); // Creating a temporary node  temp.next = head; // Setting temp node's next to the head of the original linked list  head = temp; // Assigning head to temp    // Swapping pairs using iterative approach  while (head.next != null && head.next.next != null) {  ListNode a = head.next;  ListNode b = head.next.next;  head.next = b;  a.next = b.next;  b.next = a;  head = a;  }  return temp.next; // Returning the modified linked list  }  // Method to print the linked list  static void printList(ListNode p) {  while (p != null) {  System.out.print(p.val); // Printing node value  if (p.next != null) {  System.out.print("->"); // Adding "->" if more nodes are present  }  p = p.next; // Moving to the next node  }  }  }  // Definition of ListNode class  class ListNode {  int val;  ListNode next;  ListNode(int x) {  val = x;  }  } |

Output:

A screenshot of a computer

Description automatically generated

181. Write a Java program to find the length of the last word in a given string. The string contains upper/lower-case alphabets and empty space characters like ' '.

Sample Output:

Original String: The length of last word

Length of the last word of the above string: 4

Code:

|  |
| --- |
| // Importing necessary Java utilities  import java.util.\*;  // Main class Code  public class Code {  // Main method  public static void main(String[] args) {  // Initializing a string  String str1 = "The length of last word";  // Printing the original string  System.out.println("Original String: " + str1);  // Printing the length of the last word of the string  System.out.println("Length of the last word of the above string: " + length\_Of\_last\_word(str1));  }  // Method to calculate the length of the last word in a string  public static int length\_Of\_last\_word(String str1) {  int length\_word = 0; // Initializing the variable to store the length of the last word  String[] words = str1.split(" "); // Splitting the string into words based on spaces    // Checking if words exist in the array after splitting  if (words.length > 0) {  // Assigning the length of the last word to the variable  length\_word = words[words.length - 1].length();  } else {  length\_word = 0; // If no words are present, setting the length to 0  }    return length\_word; // Returning the length of the last word  }  } |

Output:

A close up of a text

Description automatically generated

182. Write a Java program to check if two binary trees are identical. Assume that two binary trees have the same structure and every identical position has the same value.

Sample Output:

Comparing TreeNode a and TreeNode b:

false

Comparing TreeNode b and TreeNode c:

true

Code:

|  |
| --- |
| // Importing necessary Java utilities  import java.util.\*;  // Main class Code  public class Code {  // Main method  public static void main(String[] args) {  // Creating TreeNode 'a'  TreeNode a = new TreeNode(1);  a.left = new TreeNode(2);  a.right = new TreeNode(3);  a.left.left = new TreeNode(4);    // Creating TreeNode 'b'  TreeNode b = new TreeNode(1);  b.left = new TreeNode(2);  b.right = new TreeNode(3);  b.left.right = new TreeNode(4);    // Creating TreeNode 'c'  TreeNode c = new TreeNode(1);  c.left = new TreeNode(2);  c.right = new TreeNode(3);  c.left.right = new TreeNode(4);    // Comparing TreeNode 'a' and TreeNode 'b'  System.out.println("\nComparing TreeNode a and TreeNode b:");  System.out.println(is\_Identical\_tree\_node(a, b));    // Comparing TreeNode 'b' and TreeNode 'c'  System.out.println("\nComparing TreeNode b and TreeNode c:");  System.out.println(is\_Identical\_tree\_node(b, c));  }  // Method to check if two TreeNode objects are identical  public static boolean is\_Identical\_tree\_node(TreeNode a, TreeNode b) {  // Write your code here  if (a == null && b == null) return true;  if (a == null || b == null) {  return false;  }  if (a.val != b.val) return false;  return is\_Identical\_tree\_node(a.left, b.left) &&  is\_Identical\_tree\_node(a.right, b.right);  }  }  // Definition of TreeNode class  class TreeNode {  public int val;  public TreeNode left, right;  // Constructor to initialize TreeNode object with a value  public TreeNode(int val) {  this.val = val;  this.left = this.right = null;  }  } |

Output:

A screenshot of a computer

Description automatically generated

183. Write a Java program to accept a positive number and repeatedly add all its digits until the result has only one digit.

*Expected Output:*

Input a positive integer: 25

7

Code:

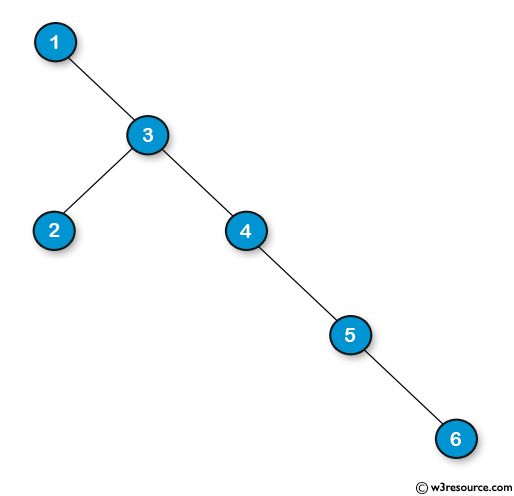
|  |
| --- |
| // Importing necessary Java utilities  import java.util.\*;  // Main class Code  public class Code {  // Main method  public static void main(String[] args) {  // Creating Scanner object for user input  Scanner in = new Scanner(System.in);    // Prompting user to input a positive integer  System.out.print("Input a positive integer: ");    // Reading the input value provided by the user  int n = in.nextInt();    // Checking if the input is a positive integer  if (n > 0)  // Printing the result of add\_digits\_until\_one method if the input is positive  System.out.println(add\_digits\_until\_one(n));  }  // Method to add digits of a number until the result becomes a single digit  public static int add\_digits\_until\_one(int n) {  // Loop to keep adding digits until the number becomes a single digit  while (n > 9) {  int sum\_digits = 0;    // Loop to extract digits and calculate their sum  while (n != 0) {  sum\_digits += n % 10; // Adding the last digit to sum  n /= 10; // Removing the last digit  }  n = sum\_digits; // Assigning the sum to 'n' for next iteration  }  return n; // Returning the single-digit sum  }  } |

Output:

A close-up of a word

Description automatically generated

184. Write a Java program to find the length of the longest consecutive sequence path in a given binary tree.  
Note: The longest consecutive path need to be from parent to child.



*Expected Output:*

Length of the longest consecutive sequence path: 4

Code:

|  |
| --- |
| // Importing necessary Java utilities  import java.util.\*;  // TreeNode class definition  class TreeNode {  public int val;  public TreeNode left, right;  // TreeNode class constructor  public TreeNode(int val) {  this.val = val;  this.left = this.right = null;  }  }  // Main class Code  public class Code {  // Main method  public static void main(String[] args) {  // Creating the tree nodes and constructing the binary tree  TreeNode a = new TreeNode(1);  a.right = new TreeNode(3);  a.right.left = new TreeNode(2);  a.right.right = new TreeNode(4);  a.right.right.right = new TreeNode(5);  a.right.right.right.right = new TreeNode(6);  // Printing the length of the longest consecutive sequence path  System.out.println("Length of the longest consecutive sequence path: " + longest\_Consecutive(a));  }  // Method to find the longest consecutive sequence path in a binary tree  public static int longest\_Consecutive(TreeNode root) {  // Base case: if the root is null, return 0  if (root == null) {  return 0;  }  // Compute the result by recursively traversing the tree  int result = diffn(root, 1) + diffn(root, -1);  return Math.max(result, Math.max(longest\_Consecutive(root.left), longest\_Consecutive(root.right)));  }  // Helper method to compute the depth of the consecutive sequence path  private static int diffn(TreeNode tnode, int diff) {  // Base case: if the tree node is null, return 0  if (tnode == null) {  return 0;  }  // Initialize depths for left and right subtrees  int left\_depth = 0, right\_depth = 0;  // Check if there exists a consecutive sequence path in left and right subtrees  if (tnode.left != null && tnode.val - tnode.left.val == diff) {  left\_depth = diffn(tnode.left, diff) + 1;  }  if (tnode.right != null && tnode.val - tnode.right.val == diff) {  right\_depth = diffn(tnode.right, diff) + 1;  }  // Return the maximum depth among left and right consecutive sequence paths  return Math.max(left\_depth, right\_depth);  }  } |

Output:



185. Write a Java program to check if two strings are isomorphic or not.

*Expected Output:*

Is abca and zbxz are Isomorphic? true

Code:

|  |
| --- |
| // Importing necessary Java utilities  import java.util.\*;  // Main class Code  public class Code {  // Main method  public static void main(String[] args) {  // Declaring and initializing two strings  String str1 = "abca";  String str2 = "zbxz";    // Printing if the two strings are isomorphic or not  System.out.println("Is " + str1 + " and " + str2 + " are Isomorphic? " + is\_Isomorphic(str1, str2));  }  // Method to check if two strings are isomorphic  public static boolean is\_Isomorphic(String str1, String str2) {  // Check for invalid inputs or unequal lengths of strings  if (str1 == null || str2 == null || str1.length() != str2.length())  return false;    // Creating a HashMap to store character mappings  Map<Character, Character> map = new HashMap<>();    // Loop through each character in the strings  for (int i = 0; i < str1.length(); i++) {  char char\_str1 = str1.charAt(i), char\_str2 = str2.charAt(i);    // If the mapping for str1 character already exists  if (map.containsKey(char\_str1)) {  // Check if the mapping matches with the corresponding character in str2  if (map.get(char\_str1) != char\_str2)  return false;  } else {  // If no mapping for str1 character exists, check if str2 character is already mapped to another str1 character  if (map.containsValue(char\_str2))  return false;    // Create a new mapping for str1 character to str2 character  map.put(char\_str1, char\_str2);  }  }    // If no discrepancies found, return true (strings are isomorphic)  return true;  }  } |

Output:

A close up of a text

Description automatically generated

186. Write a Java program to check if a number is a strobogrammatic number. The number is represented as a string.

According to Wikipedia "A strobogrammatic number is a number whose numeral is rotationally symmetric, so that it appears the same when rotated 180 degrees. In other words, the numeral looks the same right-side up and upside down (e.g., 69, 96, 1001). A strobogrammatic prime is a strobogrammatic number that is also a prime number, i.e., a number that is only divisible by one and itself (e.g., 11). It is a type of ambigram, words and numbers that retain their meaning when viewed from a different perspective, such as palindromes."  
The first few strobogrammatic numbers are:  
0, 1, 8, 11, 69, 88, 96, 101, 111, 181, 609, 619, 689, 808, 818, 888, 906, 916, 986, 1001, 1111, 1691, 1881, 1961, 6009, 6119, 6699, 6889, 6969, 8008, 8118, 8698, 8888, 8968, 9006, 9116, 9696, 9886, 9966, ...

*Expected Output:*

Is 9006 is Strobogrammatic? true

Code:

|  |
| --- |
| // Importing necessary Java utilities  import java.util.\*;  // Main class  public class Main {  // Main method  public static void main(String[] args) {  // Declaring and initializing a string  String n = "9006";  // Printing if the string is Strobogrammatic or not  System.out.println("Is " + n + " is Strobogrammatic? " + is\_Strobogrammatic(n));  }  // Method to check if the given string is Strobogrammatic  public static boolean is\_Strobogrammatic(String n) {  // Check for null or empty string  if (n == null || n.length() == 0) {  return true;  }  // Create a HashMap to store Strobogrammatic pairs  Map<Character, Character> map = new HashMap<>();  map.put('0', '0');  map.put('1', '1');  map.put('8', '8');  map.put('6', '9');  map.put('9', '6');  // Use two pointers to traverse the string from both ends  int left = 0;  int right = n.length() - 1;  // Continue until the left pointer is less than or equal to the right pointer  while (left <= right) {  // Check if the characters at the current positions are valid Strobogrammatic pairs  if (!map.containsKey(n.charAt(right)) || n.charAt(left) != map.get(n.charAt(right))) {  return false;  }  // Move the pointers towards the center  left++;  right--;  }  // If the loop completes, the string is Strobogrammatic  return true;  }  } |

Output:

A blue text on a white background

Description automatically generated

187. Write a Java program to find the index of the first non-repeating character in a given string.

*Expected Output:*

Index of first non-repeating character in 'google' is: 4

Code:

|  |
| --- |
| // Importing necessary Java utilities  import java.util.\*;  // Main class Code  public class Main {  // Main method  public static void main(String[] args) {  // Declaring and initializing a string  String str1 = "google";  // Printing the index of the first non-repeating character in the given string  System.out.println("Index of first non-repeating character in '" + str1 + "' is: " + first\_unique\_character(str1));  }  // Method to find the index of the first non-repeating character in the given string  public static int first\_unique\_character(String str1) {  // Creating a HashMap to store character frequencies  HashMap<Character, Integer> map = new HashMap<>();  // Iterating through the string to count character occurrences and store in the map  for (int i = 0; i < str1.length(); ++i) {  char chr = str1.charAt(i);  // Incrementing the count if character already exists, else adding the character with count 1  map.put(chr, map.containsKey(chr) ? map.get(chr) + 1 : 1);  }  // Iterating through the string to find the first non-repeating character  for (int i = 0; i < str1.length(); ++i) {  if (map.get(str1.charAt(i)) < 2) {  // Returning the index of the first non-repeating character  return i;  }  }  // If no non-repeating character found, returning -1  return -1;  }  } |

Output:

A close-up of a logo

Description automatically generated

188. Write a Java program to find all the start indices of a given string's anagrams in another given string.

*Expected Output:*

Original String: zyxwyxyxzwxyz

Starting anagram indices of xyz: [0, 6, 10]

Code:

|  |
| --- |
| // Importing necessary Java utilities  import java.util.\*;  // Main class  public class Main {  // Main method  public static void main(String[] args) {  // Declaring and initializing two strings  String str1 = "zyxwyxyxzwxyz";  String str2 = "xyz";  // Printing the original strings  System.out.println("Original String: " + str1);  System.out.println("Starting anagram indices of " + str2 + ": " + find\_Anagrams(str1, str2));  }  // Method to find the starting indices of anagrams of str2 in str1  public static List<Integer> find\_Anagrams(String str1, String str2) {  // Creating a list to store starting indices of anagrams  List<Integer> list = new ArrayList<>();  // Check if str1 is smaller than str2 or str2 is empty  if (str1.length() < str2.length() || str2.length() < 1) {  return list;  }  // If str1 is the same as str2, add 0 as the starting index  if (str1.equals(str2)) {  list.add(0);  return list;  }  // Creating a HashMap to store character frequencies in str2  HashMap<Character, Integer> map = new HashMap<>();  for (char c : str2.toCharArray()) {  if (map.containsKey(c)) {  map.put(c, map.get(c) + 1);  } else {  map.put(c, 1);  }  }  // Variables to track lengths and count of correct characters  int str2\_length = str2.length();  int current\_length = 0;  int correct\_chars = 0;  // Looping through str1 to find anagrams of str2  for (int i = 0; i < str1.length(); ++i) {  current\_length++;  if (map.containsKey(str1.charAt(i))) {  int ctr = map.get(str1.charAt(i));  if (ctr > 0) {  correct\_chars++;  }  map.put(str1.charAt(i), ctr - 1);  }  if (current\_length == str2\_length) {  int begin\_pos = i - str2\_length + 1;  if (correct\_chars == str2\_length) {  list.add(begin\_pos);  }  if (map.containsKey(str1.charAt(begin\_pos))) {  int ctr = map.get(str1.charAt(begin\_pos));  if (ctr >= 0) {  correct\_chars--;  }  map.put(str1.charAt(begin\_pos), ctr + 1);  }  current\_length--;  }  }  return list;  }  } |

Output:

A close-up of a blue and white font

AI-generated content may be incorrect.

189. Write a Java program to two non-negative integers num1 and num2 represented as strings, return the sum of num1 and num2.

*Expected Output:*

'123' + '456' = 579

Code:

|  |
| --- |
| // Importing necessary Java utilities  import java.util.\*;  // Main class Code  public class Code {  // Main method  public static void main(String[] args) {  // Declaring and initializing two strings representing numbers  String n1 = "123";  String n2 = "456";    // Printing the addition of two strings representing numbers  System.out.println("'" + n1 + "'" + " + " + "'" + n2 + "'" + " = " + addStrings(n1, n2));  }    // Method to add two strings representing numbers  public static String addStrings(String n1, String n2) {  // Convert input strings to integer arrays  int[] x = str\_num(n1);  int[] y = str\_num(n2);    // Initialize an array to store the sum, considering carry  int[] sum = new int[Math.max(x.length, y.length) + 1];  int z = 0;  int index = sum.length - 1;  int i = 0;  int j = 0;    // Iterate through both integer arrays to calculate the sum  while (index >= 0) {  if (i < x.length) {  z += x[i++];  }  if (j < y.length) {  z += y[j++];  }  sum[index--] = z % 10;  z /= 10; // store the carry  }    // Construct the sum string from the array  StringBuilder sb = new StringBuilder(sum.length);  for (i = (sum[0] == 0 ? 1 : 0); i < sum.length; ++i) {  sb.append(sum[i]);  }  return sb.toString();  }  // Helper method to convert a string of digits to an integer array  private static int[] str\_num(String num) {  char[] digits = num.toCharArray();  int[] number = new int[digits.length];  int index = number.length - 1;  for (char digit : digits) {  number[index--] = digit - '0'; // Convert character to integer and store in the array  }  return number;  }  } |

Output:

A white rectangular object with a black border

Description automatically generated

190. Write a Java program to find the missing string from two given strings.

*Expected Output:*

Missing string: [Code]

Code:

|  |
| --- |
| // Importing necessary Java utilities  import java.util.\*;  // Main class Code  public class Code {  // Main method  public static void main(String[] args) {  // Declaring and initializing two strings  String str1 = "Java Programming Exercises, Practice, Code";  String str2 = "Java Programming Exercises, Practice,";    // Printing the missing words in the string  System.out.println("Missing string: " + Arrays.toString(missing\_Words(str1, str2)));  }  // Method to find missing words in the given strings  public static String[] missing\_Words(String t, String s) {  // Splitting the strings into arrays using space as delimiter  String[] s1 = t.split(" ");  String[] s2 = s.split(" ");    // Calculating the number of missing words  int sz = s1.length - s2.length;  String[] missing\_str = new String[sz];  int c = 0;    // Looping through the first array to find missing words  for (int i = 0; i < s1.length; i++) {  int flag = 0;  // Checking if the word is present in the second array  for (int j = 0; j < s2.length; j++) {  if (s1[i].equals(s2[j]))  flag = 1;  }  // If word is not found in the second array, add it to missing string array  if (flag == 0) {  missing\_str[c++] = s1[i];  }  }  return missing\_str; // Return the array containing missing words  }  } |

Output:

A close-up of a sign

AI-generated content may be incorrect.

191. Write a Java program to test whether there are two integers x and y such that x^2 + y^2 is equal to a given positive number.

*Expected Output:*

Input a positive integer: 25

Is 25 sum of two square numbers? true

Code:

|  |
| --- |
| import java.util.\*;  // Main class for the Code  public class Code {  // Main method to execute the Code  public static void main(String[] args) {  // Create a Scanner object for user input  Scanner in = new Scanner(System.in);    // Prompt the user to input a positive integer  System.out.print("Input a positive integer: ");    // Read the user input as an integer  int n = in.nextInt();  // Check if the input is a positive integer  if (n > 0) {  // Display the result of the sum\_of\_square\_numbers function  System.out.print("Is " + n + " sum of two square numbers? " + sum\_of\_square\_numbers(n));  }  }  // Function to check if a number is the sum of two square numbers  public static boolean sum\_of\_square\_numbers(int n) {  // Initialize two pointers, left\_num and right\_num  int left\_num = 0, right\_num = (int) Math.sqrt(n);  // Iterate until the left\_num pointer is less than or equal to the right\_num pointer  while (left\_num <= right\_num) {  // Check if the sum of squares of left\_num and right\_num is equal to n  if (left\_num \* left\_num + right\_num \* right\_num == n) {  return true;  } else if (left\_num \* left\_num + right\_num \* right\_num < n) {  // Increment left\_num if the current sum is less than n  left\_num++;  } else {  // Decrement right\_num if the current sum is greater than n  right\_num--;  }  }  // If no pair of square numbers sum up to n, return false  return false;  }  } |

Output:

A close-up of a number

AI-generated content may be incorrect.

192. Write a Java program to rearrange the alphabets in the order followed by the sum of digits in a given string containing uppercase alphabets and integer digits (from 0 to 9).

*Expected Output:*

ADEHNS23

Code:

|  |
| --- |
| // Import necessary Java utility and language packages  import java.util.\*;  import java.lang.\*;  // Main class for the Code  public class Code {  // Constant representing the maximum number of characters  static final int MAX\_CHAR = 20;  // Main method to execute the Code  public static void main(String args[]) {  // Input string with alphanumeric characters  String str1 = "AND456HSE8";    // Print the result of the arrange\_String\_nums function  System.out.println(arrange\_String\_nums(str1));  }  // Function to arrange uppercase characters and sum of numbers in the given string  static String arrange\_String\_nums(String str1) {  // Array to count the occurrences of each uppercase character  int char\_count[] = new int[MAX\_CHAR];  // Variable to store the sum of numeric characters  int sum\_num = 0;  // Iterate through the characters in the input string  for (int i = 0; i < str1.length(); i++) {  // Check if the character is uppercase and update the char\_count array  if (Character.isUpperCase(str1.charAt(i)))  char\_count[str1.charAt(i) - 'A']++;  else  // Accumulate the numeric characters for sum  sum\_num = sum\_num + (str1.charAt(i) - '0');  }  // Initialize a string to store the rearranged characters  String rarr\_part = "";  // Iterate through the characters using their ASCII values  for (int i = 0; i < MAX\_CHAR; i++) {  // Convert ASCII value to corresponding character  char ch = (char)('A' + i);  // Append the characters to the result string based on their occurrences  while (char\_count[i]-- != 0)  rarr\_part = rarr\_part + ch;  }  // If the sum of numeric characters is greater than 0, append it to the result string  if (sum\_num > 0)  rarr\_part = rarr\_part + sum\_num;  // Return the rearranged string  return rarr\_part;  }  } |

Output:

A close-up of a number

AI-generated content may be incorrect.

193. Write a Java program that accepts an integer and sums the elements from all possible subsets of a set formed by the first n natural numbers.

*Expected Output:*

Input a positive integer: 25

Sum of subsets of n is : 1157627904

Code:

|  |
| --- |
| // Import Scanner class from java.util package for user input  import java.util.Scanner;  // Main class for the Code  public class Code {  // Main method to execute the Code  public static void main(String[] args) {  // Create a Scanner object for user input  Scanner in = new Scanner(System.in);  // Prompt the user to input a positive integer  System.out.print("Input a positive integer: ");  // Read the user input as an integer  int n = in.nextInt();  // Calculate the sum of subsets using a mathematical formula  int result = (n \* (n + 1) / 2) \* (1 << (n - 1));  // Display the result of the sum of subsets  System.out.print("Sum of subsets of n is : " + result);  }  } |

Output:

A close-up of a math

AI-generated content may be incorrect.

194. Write a Java program to determine the all positions of a given number in a given matrix. If the number is not found print ("Number not found!").

*Expected Output:*

(0,2)

(1,0)

(2,1)

Code:

|  |
| --- |
| // Main class for the Code  public class Code {  // Main method to execute the Code  public static void main(String[] args) {  // Initialize the target number  int num = 3;  // Initialize a 2D matrix  int matrix[][] = {  {2, 5, 3},  {3, 2, 1},  {1, 3, 5}  };  // Get the number of rows in the matrix  int r = matrix.length;  // Get the number of columns in the matrix  int c = matrix[0].length - 1;  // Initialize variables for matrix traversal  int m = 0, n = 0;  // Boolean flag to check if the number is found in the matrix  Boolean flag = false;  // Iterate through the rows of the matrix  while (m < r) {  // Iterate through the columns of the matrix  while (n <= c) {  // Check if the current element is equal to the target number  if (matrix[m][n] == num) {  // Display the coordinates of the found number  System.out.print("\n(" + m + "," + n + ")\n");  // Set the flag to true indicating the number is found  flag = true;  }  // Move to the next column  n++;  }  // Move to the next row and reset column index  m++;  n = 0;  }  // Display a message if the number is not found in the matrix  if (flag == false)  System.out.print("Number not found!");  }  } |

Output:

A screenshot of a computer

AI-generated content may be incorrect.

195. Write a Java program to check if three given side lengths (integers) can make a triangle or not.

*Expected Output:*

Input side1: 5

Input side2: 6

Input side3: 8

Is the said sides form a triangle: true

Code:

|  |
| --- |
| // Import Scanner class from java.util package for user input  import java.util.\*;  // Main class for the Code  public class Code {  // Main method to execute the Code  public static void main(String[] args) {  // Create a Scanner object for user input  Scanner in = new Scanner(System.in);  // Prompt the user to input the first side of the triangle  System.out.print("Input side1: ");  // Read the user input as an integer  int s1 = in.nextInt();  // Prompt the user to input the second side of the triangle  System.out.print("Input side2: ");  // Read the user input as an integer  int s2 = in.nextInt();  // Prompt the user to input the third side of the triangle  System.out.print("Input side3: ");  // Read the user input as an integer  int s3 = in.nextInt();  // Display the result of the isValidTriangle function  System.out.print("Is the said sides form a triangle: " + isValidTriangle(s1, s2, s3));  }  // Function to check if the given sides form a valid triangle  public static boolean isValidTriangle(int a, int b, int c) {  // Check the triangle inequality theorem to determine validity  return (a + b > c && b + c > a && c + a > b);  }  } |

Output:

A white background with blue text

AI-generated content may be incorrect.

196. rite a Java program to create a spiral array of n \* n sizes from a given integer n.

*Expected Output:*

Input a number: 5

Spiral array becomes:

1 2 3 4 5

16 17 18 19 6

15 24 25 20 7

14 23 22 21 8

13 12 11 10 9

Code:

|  |
| --- |
| // Import Scanner class from java.util package for user input  import java.util.\*;  // Main class for the Code  public class Code {  // Main method to execute the Code  public static void main(String[] args) {  // Create a Scanner object for user input  Scanner in = new Scanner(System.in);    // Prompt the user to input a number  System.out.print("Input a number: ");    // Read the user input as an integer  int n = in.nextInt();    // Generate a spiral array using the spiral\_Array function  int[][] result = spiral\_Array(n);    // Display the generated spiral array  System.out.print("Spiral array becomes:\n");  for(int i = 0; i < result.length; i++) {  for(int j = 0; j < result[i].length; j++) {  System.out.print(result[i][j]);  if(j < result[i].length - 1) System.out.print(" ");  }  System.out.println();  }  }  // Function to generate a spiral array of size n x n  public static int[][] spiral\_Array(int n) {  // Initialize a 2D array to store the spiral array  int[][] temp = new int[n][n];    // Arrays to represent movement in x and y directions  int[] dx = new int[]{0, 1, 0, -1};  int[] dy = new int[]{1, 0, -1, 0};    // Variables for current position (x, y) and direction (d)  int x, y, d;    // Variables for iteration  int i, j, nx, ny;    // Initialize the array with -1 values  for (i = 0; i < n; ++i) {  for (j = 0; j < n; ++j) {  temp[i][j] = -1;  }  }    // Initialize starting position and direction  x = 0;  y = 0;  d = 0;    // Fill the array with spiral order values  for (i = 1; i <= n \* n; ++i) {  temp[x][y] = i;  nx = x + dx[d];  ny = y + dy[d];    // Check boundaries and visited positions  if (nx < 0 || nx >= n || ny < 0 || ny >= n || temp[nx][ny] != -1) {  d = (d + 1) % 4; // Change direction if boundary or visited  nx = x + dx[d];  ny = y + dy[d];  }    // Update current position  x = nx;  y = ny;  }    // Return the generated spiral array  return temp;  }  } |

Output:

A screenshot of a computer

AI-generated content may be incorrect.

197. Write a Java program to test if a given number (positive integer) is a perfect square or not.

*Expected Output:*

Input a positive integer: 6

Is the said number perfect square? false

Code:

|  |
| --- |
| // Import Scanner class from java.util package for user input  import java.util.\*;  // Main class for the Code  public class Code {  // Main method to execute the Code  public static void main(String[] args) {  // Create a Scanner object for user input  Scanner in = new Scanner(System.in);  // Prompt the user to input a positive integer  System.out.print("Input a positive integer: ");  // Read the user input as an integer  int n = in.nextInt();  // Display the result of the is\_Perfect\_Square function  System.out.print("Is the said number perfect square? " + is\_Perfect\_Square(n));  }  // Function to check if a given number is a perfect square  public static boolean is\_Perfect\_Square(int n) {  // Extract the last digit of the number  int x = n % 10;  // Check if the last digit is 2, 3, 7, or 8 (numbers whose squares end with these digits)  if (x == 2 || x == 3 || x == 7 || x == 8) {  return false;  }  // Iterate from 0 to half of the input number plus 1  for (int i = 0; i <= n / 2 + 1; i++) {  // Check if the square of the current iteration is equal to the input number  if ((long) i \* i == n) {  return true;  }  }  // If no perfect square is found, return false  return false;  }  } |

Output:

A close-up of a sign

AI-generated content may be incorrect.

198. Write a Java program to calculate the position of a given prime number.

*Expected Output:*

Input a positive integer: 15

Position of the said Prime number: 6

Code:

|  |
| --- |
| // Import Scanner class from java.util package for user input  import java.util.\*;  // Main class for the Code  public class Code {  // Main method to execute the Code  public static void main(String[] args) {  // Create a Scanner object for user input  Scanner in = new Scanner(System.in);  // Prompt the user to input a prime number  System.out.print("Input a prime number: ");  // Read the user input as an integer  int n = in.nextInt();  // Display the position of the given prime number using the kth\_Prime function  System.out.print("Position of the said Prime number: " + kth\_Prime(n));  }  // Function to find the position of a given prime number in the sequence of primes  public static int kth\_Prime(int n) {  // Array to store prime numbers, initialized with the first prime number (2)  int[] prime\_num = new int[10000];  int num = 3; // Starting from the next number after 2  int i = 0, index = 0; // Variables for iteration and index tracking  prime\_num[0] = 2; // Initialize the first prime number in the array  // Continue finding primes until reaching the input number  while (num <= n) {  // Iterate through the existing primes to check if num is divisible  for (i = 0; i <= index; i++) {  if (num % prime\_num[i] == 0) {  break;  }  }  // If num is not divisible by any existing primes, add it to the array  if (i > index) {  prime\_num[++index] = num;  }  // Move on to the next number  num++;  }  // Return the position of the input prime number in the sequence  return index + 1;  }  } |

Output:

A close up of a text

AI-generated content may be incorrect.

199. Write a Java program to check if a string follows a given pattern.

Example pattern:  
Given pattern = "xyyx", str = "red black black red", return true.  
Given pattern = "xyyx", str = "red black black green", return false.  
Given pattern = "xxxx", str = "red black black red", return false.  
Given pattern = "xxxx", str = "red red red red", return true.

*Expected Output:*

Is the string and pattern matched? false

Code:

|  |
| --- |
| // Import Scanner and Map classes from java.util package for user input and data storage  import java.util.\*;  // Main class for the Code  public class Code {  // Main method to execute the Code  public static void main(String[] args) {  // Sample input strings for testing word pattern matching  String str = "red black black red";  // String str = "red red red red";  String pattern = "xyxx";  // String pattern = "xxxx";    // Display the result of the word\_Pattern\_Match function  System.out.print("Is the string and pattern matched? " + word\_Pattern\_Match(pattern, str));  }  // Function to check if a given string follows a given word pattern  public static boolean word\_Pattern\_Match(String pattern, String str) {  // Convert the pattern string to an array of characters  char[] word\_pattern = pattern.toCharArray();    // Split the input string into an array of words using space as a delimiter  String[] words = str.split(" ");  // Create a HashMap to store the mapping between characters and words  Map map = new HashMap<>();    // Create a HashSet to check for duplicate mappings  Set set = new HashSet<>();  // Iterate through the characters in the pattern  for (int i = 0; i < word\_pattern.length; i++) {  // Check if the character is already mapped  if (map.containsKey(word\_pattern[i])) {  // Check if the mapped word is different from the current word in the array  if (!map.get(word\_pattern[i]).equals(words[i])) {  return false;  }  continue;  }  // Check if the current word is already mapped to another character  if (set.contains(words[i])) {  return false;  }    // Add the mapping between the character and the current word to the HashMap  map.put(word\_pattern[i], words[i]);    // Add the current word to the HashSet to mark it as used  set.add(words[i]);  }  // If all conditions are satisfied, return true  return true;  }  } |

Output:

A close-up of a white background

AI-generated content may be incorrect.

200. Write a Java program to remove duplicate letters and arrange them in lexicographical order from a given string containing only lowercase letters.

Note: In mathematics, the lexicographic or lexicographical order (also known as lexical order, dictionary order, alphabetical order or lexicographic(al) product) is a generalization of the way words are alphabetically ordered based on the alphabetical order of their component letters.

*Expected Output:*

Original string: zxywooxz

After removing duplicate characters: xywoz

Code:

|  |
| --- |
| // Import Scanner class from java.util package for user input  import java.util.\*;  // Main class for the Code  public class Main {  // Main method to execute the Code  public static void main(String[] args) {  // Sample input string for testing duplicate letter removal  String str = "zxywooxz";  // Display the original string  System.out.print("Original string: " + str);  // Display the result after removing duplicate characters and arranging in lexicographical order  System.out.print("\nAfter removing duplicate characters and arranging in lexicographical order: " + removeDuplicateLetters(str));  }  // Function to remove duplicate letters from the given string and arrange in lexicographical order  public static String removeDuplicateLetters(String s) {  // Array to track whether a letter is already in the result  boolean[] inResult = new boolean[26];  // Array to count the occurrences of each lowercase letter  int[] count = new int[26];  // Stack to store the characters  Stack<Character> stack = new Stack<>();  // Count the occurrences of each letter in the input string  for (char c : s.toCharArray()) {  count[c - 'a']++;  }  // Iterate through the characters in the input string  for (char c : s.toCharArray()) {  // Decrement the count of the current character in the occurrences array  count[c - 'a']--;  // If the character is already in the result, skip  if (inResult[c - 'a']) continue;  // Pop characters from the stack while conditions are met  while (!stack.isEmpty() && c < stack.peek() && count[stack.peek() - 'a'] > 0) {  inResult[stack.pop() - 'a'] = false;  }  // Push the current character onto the stack  stack.push(c);  inResult[c - 'a'] = true;  }  // Sort the characters in the stack  Collections.sort(stack);  // Build the result string from the characters in the stack  StringBuilder result = new StringBuilder();  for (char c : stack) {  result.append(c);  }  return result.toString();  }  } |

Output:

A close-up of a white background

AI-generated content may be incorrect.

A close-up of a sign

AI-generated content may be incorrect.

201. Write a Java program to divide a given array of integers into given k non-empty subsets whose sums are all equal. Return true if all sums are equal otherwise return false.

Example:  
nums = {1,3,3,5,6,6}, k = 4;  
4 subsets (5,1), (3, 3), (6), (6) with equal sums.

*Expected Output:*

Original Array: [1, 3, 3, 5, 6, 6]

Target of subsets: 4

After removing duplicate characters: true

Code:

|  |
| --- |
| // Import Arrays and other utility classes from java.util package  import java.util.Arrays;  // Main class for the Code  public class Code {  // Main method to execute the Code  public static void main(String[] args) {  // Sample input array and target value for testing subset partitioning  int[] nums = {1, 3, 3, 5, 6, 6};  int target = 4;  // Display the original array  System.out.print("Original Array: " + Arrays.toString(nums));  // Display the target value for subsets  System.out.print("\nTarget of subsets: " + target);  // Display the result after removing duplicate characters using partition\_k\_subsets function  System.out.print("\nAfter removing duplicate characters: " + partition\_k\_subsets(nums, target));  }  // Function to recursively search for valid subsets with a specific sum  static boolean search\_subset(int used, int n, boolean[] flag, int[] nums, int target) {  // Base case: all elements used, subset found  if (n == 0) {  return true;  }  // Check if the current subset has not been considered before  if (!flag[used]) {  // Mark the current subset as visited  flag[used] = true;  // Calculate the remaining sum needed for the subset  int remain\_num = (n - 1) % target + 1;  // Iterate through the elements in the array  for (int i = 0; i < nums.length; i++) {  // Check if the current element is not used in the subset and its value is less than or equal to the remaining sum  if ((((used >> i) & 1) == 0) && nums[i] <= remain\_num) {  // Recursively search for the subset with the updated parameters  if (search\_subset(used | (1 << i), n - nums[i], flag, nums, target)) {  return true;  }  }  }  }  return false;  }  // Function to partition an array into k subsets with equal sum  public static boolean partition\_k\_subsets(int[] nums, int k) {  // Calculate the total sum of the elements in the array  int sum = Arrays.stream(nums).sum();  // Check if the sum is not divisible by k, return false  if (sum % k > 0) {  return false;  }  // Create a boolean array to track visited subsets  boolean[] flag = new boolean[1 << nums.length];  // Call the recursive search\_subset function to check for valid subsets  return search\_subset(0, sum, flag, nums, sum / k);  }  } |

Output:

A close-up of a text

AI-generated content may be incorrect.

202. Write a Java program to find the total number of continuous subarrays in a given array of integers whose sum equals an integer.

*Expected Output:*

Original Array: [4, 2, 3, 3, 7, 2, 4]

Value of k: 6

Total number of continuous subarrays: 3

Code:

|  |
| --- |
| // Import utility classes from java.util package  import java.util.\*;  // Main class  public class Main {  // Main method to execute the Code  public static void main(String[] args) {  // Sample input array and value of k for counting continuous subarrays  int[] nums = {4, 2, 3, 3, 7, 2, 4};  int k = 6;  // Display the original array  System.out.print("Original Array: " + Arrays.toString(nums));  // Display the value of k  System.out.print("\nValue of k: " + k);  // Display the total number of continuous subarrays whose sum equals k  System.out.print("\nTotal number of continuous subarrays: " + max\_SubArray(nums, k));  }  // Function to find the total number of continuous subarrays whose sum equals k  public static int max\_SubArray(int[] nums, int k) {  int ctr = 0; // Counter for total subarrays found  int sum = 0; // Variable to track current sum  Map<Integer, Integer> map = new HashMap<>(); // HashMap to store prefix sums and their counts  // Initialize the map with a sum of 0 and count 1 (base case)  map.put(0, 1);  // Iterate through the input array  for (int i = 0; i < nums.length; i++) {  sum += nums[i]; // Update the current sum  // Check if there exists a prefix sum at (sum - k), increment counter if found  if (map.containsKey(sum - k)) {  ctr += map.get(sum - k);  }  // Update the count of the current sum in the map  map.put(sum, map.getOrDefault(sum, 0) + 1);  }  // Return the total count of continuous subarrays whose sum equals k  return ctr;  }  } |

Output:

A close-up of a number

AI-generated content may be incorrect.

203. Write a Java program to find the contiguous subarray of given length k which has the maximum average value of a given array of integers. Display the maximum average value.

*Expected Output:*

Original Array: [4, 2, 3, 3, 7, 2, 4]

Value of k: 3

Maximum average value: 4.333333333333333

Code:

|  |
| --- |
| import java.util.\*;  // Main class named "Main"  public class Main {  // Main method, the entry point of the program  public static void main(String[] args) {  // Sample input array and value of k for finding maximum average  int[] nums = {4, 2, 3, 3, 7, 2, 4};  int k = 3;  // Display the original array  System.out.print("Original Array: " + Arrays.toString(nums));  // Display the value of k  System.out.print("\nValue of k: " + k);  // Display the maximum average value  System.out.print("\nMaximum average value: " + find\_max\_average(nums, k));  }  // Function to find the maximum average of subarrays of length k  public static double find\_max\_average(int[] nums, int k) {  int sum = 0;  // Calculate the initial sum of the first k elements  for (int i = 0; i < k; i++) {  sum += nums[i];  }  int max\_val = sum;  // Iterate through the array to find the maximum average  for (int i = k; i < nums.length; i++) {  // Update the sum by removing the leftmost element and adding the current element  sum = sum - nums[i - k] + nums[i];  // Update the maximum value if the current sum is greater  max\_val = Math.max(max\_val, sum);  }  // Return the maximum average value  return (double) max\_val / k;  }  } |

Output:

A screenshot of a computer

AI-generated content may be incorrect.

204. Write a Java program to compute xn % y where x, y and n are all 32-bit integers.

*Expected Output:*

Input x : 25

Input n : 35

Input y : 45

x^n % y = 5.0

Code:

|  |
| --- |
| // Import Scanner class from java.util package for user input  import java.util.\*;  // Main class for the Code  public class Main {  // Main method to execute the Code  public static void main(String[] args) {  // Create a Scanner object for user input  Scanner in = new Scanner(System.in);  // Prompt the user to input x  System.out.print("Input x : ");  // Read the user input as an integer  int x = in.nextInt();  // Prompt the user to input n  System.out.print("Input n : ");  // Read the user input as an integer  int n = in.nextInt();  // Prompt the user to input y  System.out.print("Input y : ");  // Read the user input as an integer  int y = in.nextInt();  // Calculate the result of x raised to the power of n  double result = Math.pow(x, n);  // Calculate the remainder when result is divided by y  double result1 = result % y;  // Display the result of (x^n % y)  System.out.println("x^n % y = " + result1);  }  } |

Output:

A screenshot of a computer

AI-generated content may be incorrect.

205. Write a Java program to check whether an integer is a power of 2 or not using O(1) time.

Note: O(1) means that it takes a constant time, like 12 nanoseconds, or two minutes no matter the amount of data in the set.  
O(n) means it takes an amount of time linear with the size of the set, so a set twice the size will take twice the time. You probably don't want to put a million objects into one of these.

*Expected Output:*

Input a number : 25

false

Code:

|  |
| --- |
| import java.util.\*;  public class Main {  public static void main(String[] args) {  // Initialize a boolean variable  boolean b = true;  // Create a Scanner object for user input  Scanner in = new Scanner(System.in);  // Prompt the user to input a number  System.out.print("Input a number: ");  int num = in.nextInt();  // Start a block of code  {  // Continue looping until num becomes 1  while (num != 1) {  // Check if num is odd  if (num % 2 != 0) {  // Toggle the boolean variable  b = !b;  // Print the current value of the boolean variable and exit the program  System.out.print(b);  System.exit(0);  }  // Divide num by 2  num = num / 2;  }  // Print the final value of the boolean variable  System.out.print(b);  }  }  } |

Output:

A close-up of a number

AI-generated content may be incorrect.

206. From Wikipedia,  
A cyclic redundancy check (CRC) is an error-detecting code commonly used in digital networks and storage devices to detect accidental changes to raw data. Blocks of data entering these systems get a short check value attached, based on the remainder of a polynomial division of their contents. On retrieval, the calculation is repeated and, in the event the check values do not match, corrective action can be taken against data corruption. CRCs can be used for error correction.

Example:

Write a Java program to generate a CRC32 checksum of a given string or byte array.

Input:

Input a string: The quick brown fox

crc32 checksum of the string: b74574de

Code:

|  |
| --- |
| import java.util.\*;  public class Main {  public static void main(String[] args) {  // Initialize a boolean variable  boolean b = true;  // Create a Scanner object for user input  Scanner in = new Scanner(System.in);  // Prompt the user to input a number  System.out.print("Input a number: ");  int num = in.nextInt();  // Start a block of code  {  // Continue looping until num becomes 1  while (num != 1) {  // Check if num is odd  if (num % 2 != 0) {  // Toggle the boolean variable  b = !b;  // Print the current value of the boolean variable and exit the program  System.out.print(b);  System.exit(0);  }  // Divide num by 2  num = num / 2;  }  // Print the final value of the boolean variable  System.out.print(b);  }  }  } |

Output:

A close up of text

AI-generated content may be incorrect.

207. Write a Java program to merge two sorted (ascending) linked lists in ascending order.

*Expected Output:*

How many elements do you want to add in 1st linked list?: 3

Input numbers of 1st linked list in ascending order: 1 2 3

How many elements do you want to add in 2nd linked list?: 3

Input numbers of 2nd linked list in ascending order: 4 5 6

Merged list: 1 2 3 4 5 6

Code:

|  |
| --- |
| // Importing required classes from the java.util package  import java.util.Scanner;  import java.util.BitSet;  // Defining a class named "Code"  public class Code {  // Method to convert a byte array to CRC32 checksum  public static int convert\_crc32(byte[] data) {  // Creating a BitSet to represent the bits of the input byte array  BitSet bitSet = BitSet.valueOf(data);  // Initializing CRC32 to 0xFFFFFFFF  int crc32 = 0xFFFFFFFF;  // Looping through each bit in the BitSet  for (int i = 0; i < data.length \* 8; i++) {  // Checking if the MSB of CRC32 and the current bit in BitSet are different  if (((crc32 >>> 31) & 1) != (bitSet.get(i) ? 1 : 0))  // If different, performing XOR with the polynomial 0x04C11DB7  crc32 = (crc32 << 1) ^ 0x04C11DB7;  else  // If same, shifting CRC32 to the left  crc32 = (crc32 << 1);  }  // Reversing the bits of CRC32  crc32 = Integer.reverse(crc32);  // Returning the final CRC32 checksum by performing XOR with 0xFFFFFFFF  return crc32 ^ 0xFFFFFFFF;  }    // Main method, the entry point of the program  public static void main(String[] args) {  // Creating a Scanner object for user input  Scanner scanner = new Scanner(System.in);  // Prompting the user to input a string  System.out.print("Input a string: ");  // Reading the input string from the user  String str1 = scanner.nextLine();  // Calling the convert\_crc32 method and printing the CRC32 checksum in hexadecimal format  System.out.println("crc32 checksum of the string: " + Integer.toHexString(convert\_crc32(str1.getBytes())));  }  } |

Output:

A screenshot of a computer screen

AI-generated content may be incorrect.

208. Write a Java program to create a basic string compression method using repeated character counts.

Input string: aaaabbbbcccccddddeeee

*Expected Output:*

Enter a string (you can include space as well)

aaaabbbbcccccddddeeee

The compressed string along with the counts of repeated characters is:

a4b4c5d4e4

Code:

|  |
| --- |
| import java.util.Scanner;  public class StringCompression {  public static void main(String[] args) {  // Create an instance of the StringCompression class  StringCompression str = new StringCompression();    String s1, s2;  Scanner in = new Scanner(System.in);    // Prompt the user to enter a string (including spaces)  System.out.println("Enter a string (you can include space as well)");  s1 = in.nextLine();    // Trim all the spaces from the string using replaceAll method  s2 = s1.replaceAll("\\s", "");    // Call the Compression method to compress the string  str.Compression(s2);  }    // Create a Java Method Compression to compress the string  public static String Compression(String s) {  int count = 1;  StringBuilder sb = new StringBuilder();  // Below for loop counts all characters of the string apart from the last one  // The last character won't get appended by the StringBuilder here as it  // does not enter the for loop once the length completes the count  for (int i = 1; i < s.length() - 1; i++) {  if (s.charAt(i) == s.charAt(i - 1)) {  count++;  } else {  sb.append(s.charAt(i - 1));  sb.append(count);  count = 1;  }  }    // Count the last character of the string  if (s.length() > 1) {  // Compare the last two characters of the string  if (s.charAt(s.length() - 1) == s.charAt(s.length() - 2)) {  count++;  } else {  sb.append(s.charAt(s.length() - 2));  sb.append(count);  count = 1;  }  sb.append(s.charAt(s.length() - 1));  sb.append(count);  }    // Convert the StringBuilder to a string  s = sb.toString();    // Print the compressed string along with the counts of repeated characters  System.out.println("The compressed string along with the counts of repeated characters is:" + "\n" + s);    // Return the compressed string  return s;  }  } |

Output:

A computer screen shot of a computer

AI-generated content may be incorrect.

A computer screen shot of a computer screen

AI-generated content may be incorrect.

209. Write a Java program to find all unique combinations from a collection of candidate numbers. The sum of the numbers will equal a given target number.

Input number of elements of the array:  
3  
Input number format: 2 3 4 5:

*Expected Output:*

Enter elements:

6 7 8

Enter target sum:

21

A Code set is:

{ 6 7 8 }

Code:

|  |
| --- |
| import java.util.\*;  class Main {    // Method to insert values into a Map with key as a generic type and value as a List of generic type  private static <K, V> void insert(Map<K, List<V>> hashMap, K key, V value) {  // If the key is not present in the map, create a new entry with an empty ArrayList  if (!hashMap.containsKey(key)) {  hashMap.put(key, new ArrayList<>());  }  // Add the value to the list corresponding to the key  hashMap.get(key).add(value);  }  // Method to print subsets of an array from index i to j  public static void Subsets(int[] A, int i, int j) {  System.out.print("{ ");  for (int k = i; k <= j; k++) {  System.out.print(A[k] + " ");  }  System.out.println("}");  }  // Method to find subsets with a given sum in the array  public static void Subsets(int[] A, int sum) {  // Create a HashMap to store the cumulative sum and corresponding indices  Map<Integer, List<Integer>> hashMap = new HashMap<>();  // Insert an initial entry with key 0 and value -1 (sum\_so\_far - sum = 0 - sum)  insert(hashMap, 0, -1);  int sum\_so\_far = 0;  for (int index = 0; index < A.length; index++) {  // Update the cumulative sum  sum\_so\_far += A[index];  // If the HashMap contains the key (cumulative sum - sum), print subsets  if (hashMap.containsKey(sum\_so\_far - sum)) {  List<Integer> list = hashMap.get(sum\_so\_far - sum);  for (Integer value : list) {  Subsets(A, value + 1, index);  }  }  // Insert the current cumulative sum and index into the HashMap  insert(hashMap, sum\_so\_far, index);  }  }  public static void main(String[] args) {  // Scanner for user input  Scanner s = new Scanner(System.in);  // Prompt for the number of elements in the array  System.out.println("Input number of elements of the array: ");  int n = s.nextInt();  // Prompt for entering array elements in number format  System.out.println("Input number format: 2 3 4 5: ");  int arr[] = new int[n];  // Prompt for entering array elements  System.out.println("Enter elements:");  for (int i = 0; i < n; i++)  arr[i] = s.nextInt();  // Prompt for entering the target sum  System.out.println("Enter target sum:");  int sum = s.nextInt();  // Create a copy of the original array  int A[] = Arrays.copyOf(arr, arr.length);  // Print the Code set (subsets with the given sum)  System.out.println("A Code set is:");  Subsets(A, sum);  // Exit the program  System.exit(0);  }  } |

Output:

A white background with blue text

AI-generated content may be incorrect.

210. Write a Java program to match any single character (use ?) or any sequence of characters (use \*) including empty. The matching should cover the entire input string.

*Expected Output:*

Enter a string

bb

Enter a pattern

b\*

Yes

Code:

|  |
| --- |
| import java.util.\*;  public class PatternMatching {    // Method for wildcard pattern matching  static boolean pattern\_match(String string, String pattern) {  // i measures the length of the string  int i = 0;  // j measures the length of the pattern  int j = 0;  int star\_index = -1;  int i\_index = -1;  while (i < string.length()) {  // If '?' matches the ith character of the string or if the jth character of the  // pattern matches the ith character of the string. e.g. (a & ?), (ab & ab)  if (j < pattern.length() && (pattern.charAt(j) == '?' || pattern.charAt(j) == string.charAt(i))) {  ++i;  ++j;  }  // Counts '\*' characters of the pattern when the count of the string is not  // completed yet. e.g. (a & \*\*\*), (abb & ab\*\*\*\*)  else if (j < pattern.length() && pattern.charAt(j) == '\*') {  star\_index = j;  i\_index = i;  j++;  }  // Counts the characters of the string which are left out once a '\*' of the pattern  // gets counted e.g. (xayb & \*a\*b), (a & \*\*\*), (abcd & ab\*), (aa & ?\*\*)  else if (star\_index != -1) {  j = star\_index + 1;  i = i\_index + 1;  i\_index++;  }  // If the characters of the string and pattern don't match  // e.g. (xy & ab), (abxy & ab)  else {  return false;  }  }  // Counts the '\*' characters of the pattern when the characters before the '\*' characters  // of the pattern completely match the string and both are of the same length  // (apart from the '\*' characters of the pattern)  // e.g. (ab and ab\*\*), (aa and ??\*\*)  while (j < pattern.length() && pattern.charAt(j) == '\*') {  ++j;  }  return j == pattern.length();  }  public static void main(String args[]) {  String str, pat;  Scanner in = new Scanner(System.in);  System.out.println("Enter a string");  str = in.nextLine();  System.out.println("Enter a pattern");  pat = in.nextLine();  if (pattern\_match(str, pat))  System.out.println("Yes");  else  System.out.println("No");  }  } |

Output:

A screenshot of a computer

AI-generated content may be incorrect.

211. Write a Java program to find the heights of the top three buildings in descending order from eight given buildings.

Input:

0 ≤ height of building (integer) ≤ 10,000

*Expected Output:*

Input the heights of eight buildings:

25 19 23 45 18 23 24 19

Heights of the top three buildings:

45

25

24

Code:

|  |
| --- |
| import java.util.\*;  public class Main {  public static void main(String[] args) {  // Creating a Scanner object for user input  Scanner sc = new Scanner(System.in);  // Array to store the heights of eight buildings  int[] t = new int[8];  // Prompting the user to input the heights of eight buildings  System.out.println("Input the heights of eight buildings:");  for (int i = 0; i < 8; i++) {  t[i] = sc.nextInt();  }  // Sorting the array of building heights in ascending order  Arrays.sort(t);  // Displaying the heights of the top three buildings in descending order  System.out.println("\nHeights of the top three buildings:");  for (int i = 7; i >= 5; i--) {  System.out.println(t[i]);  }  }  } |

Output:

A screenshot of a computer

AI-generated content may be incorrect.

212. Write a Java program to compute the digit number of the sum of two given integers.

Input:

Each test case consists of two non-negative integers a and b which are separated by a space in a line. 0 ≤ a, b ≤ 1,000,000

*Expected Output:*

Input two integers(a b):

13 25

Digit number of sum of said two integers:

2

Code:

|  |
| --- |
| import java.util.\*;  public class Main {  public static void main(String[] args) {  // Prompting the user to input two integers (a and b)  System.out.println("Input two integers(a b):");  // Creating a Scanner object for user input  Scanner stdIn = new Scanner(System.in);  // Reading the values of integers a and b from user input  int a = stdIn.nextInt();  int b = stdIn.nextInt();  // Calculating the sum of integers a and b  int sum = a + b;  // Initializing a variable to count the number of digits in the sum  int count = 0;  // Counting the number of digits in the sum using a while loop  while (sum != 0) {  sum /= 10;  ++count;  }  // Displaying the digit number of the sum of the two integers  System.out.println("Digit number of sum of said two integers:");  System.out.println(count);  }  } |

Output:

A screenshot of a computer

AI-generated content may be incorrect.

213. Write a Java program to check whether three given lengths (integers) of three sides form a right triangle. Print "Yes" if the given sides form a right triangle otherwise print "No".

Input:

Each test case consists of two non-negative integers a and b which are separated by a space in a line. 0 ≤ a, b ≤ 1,000,000

*Expected Output:*

Input three integers(sides of a triangle)

6 9 12

If the given sides form a right triangle?

No

Code:

|  |
| --- |
| import java.util.Arrays;  import java.util.Comparator;  import java.util.Scanner;  class Main {  // Creating a Scanner object for user input  Scanner sc = new Scanner(System.in);  // Method to execute the main functionality  public void run() {  // Prompting the user to input three integers (sides of a triangle)  System.out.println("Input three integers(sides of a triangle)");  // Reading three integers and storing them in an array  int[] int\_num = new int[]{  sc.nextInt(), sc.nextInt(), sc.nextInt()  };  // Sorting the array of integers in ascending order  Arrays.sort(int\_num);  // Checking if the given sides form a right triangle  System.out.println("If the given sides form a right triangle?");  ln((int\_num[2] \* int\_num[2] == int\_num[0] \* int\_num[0] + int\_num[1] \* int\_num[1]) ? "Yes" : "No");  }  // Main method to create an instance of the class and run the program  public static void main(String[] args) {  new Main().run();  }  // Method for printing without a newline  public static void pr(Object o) {  System.out.print(o);  }  // Method for printing with a newline  public static void ln(Object o) {  System.out.println(o);  }  // Method for printing an empty line  public static void ln() {  System.out.println();  }  } |

Output:

A white background with blue text

AI-generated content may be incorrect.

214. Write a Java program which solve the equation:  
ax+by=c  
dx+ey=f  
Print the values of x, y where a, b, c, d, e and f are given.

Input:

a,b,c,d,e,f separated by a single space.  
(-1,000 ≤ a,b,c,d,e,f ≤ 1,000)

*Expected Output:*

Input the value of a, b, c, d, e, f:

5 6 8 9 7 4

-1.684 2.737

Code:

|  |
| --- |
| import java.math.BigDecimal;  import java.util.\*;  public class Main {  public static void main(String[] args) {  // Creating a Scanner object for user input  Scanner sc = new Scanner(System.in);  // Creating ArrayDeque to store Double values for x and y  ArrayDeque<Double>x = new ArrayDeque<>();  ArrayDeque<Double> y = new ArrayDeque<>();  // Prompting the user to input the values of a, b, c, d, e, f  System.out.println("Input the value of a, b, c, d, e, f:");  // Reading values for coefficients a, b, c, d, e, f  int a = sc.nextInt();  int b = sc.nextInt();  int c = sc.nextInt();  int d = sc.nextInt();  int e = sc.nextInt();  int f = sc.nextInt();  // Calculating values for variables s and t  double t = (double) (d \* c - a \* f) / (d \* b - a \* e);  double s = (double) (c - t \* b) / a;  // Pushing the calculated values of x and y into the respective Deques  x.push(s);  y.push(t);  // Getting the size of the Deques  int num = x.size();  // Iterating through the Deques to print the results with rounded values  for (int i = 0; i < num; i++) {  BigDecimal bdx = new BigDecimal(x.pollLast());  BigDecimal bdy = new BigDecimal(y.pollLast());  BigDecimal ansx = bdx.setScale(4, BigDecimal.ROUND\_HALF\_UP);  BigDecimal ansy = bdy.setScale(4, BigDecimal.ROUND\_HALF\_UP);  // Printing the rounded values of x and y  System.out.printf("%.3f", ansx.doubleValue());  System.out.print(" ");  System.out.printf("%.3f", ansy.doubleValue());  System.out.println();  }  }  } |

Output:

A close up of a text

AI-generated content may be incorrect.

215. Write a Java program to compute the debt amount in n months. Monthly, the loan adds 4% interest to the $100,000 borrowed and rounds it to the nearest 1,000.

Input:

An integer n (0 ≤ n ≤ 100)

*Expected Output:*

Input number of months:

6

Amount of debt:

129000

Code:

|  |
| --- |
| import java.util.Scanner;  public class Main {  public static void main(String[] args) {  // Creating a Scanner object for user input  Scanner s = new Scanner(System.in);  // Prompting the user to input the number of months  System.out.println("Input number of months:");  // Reading the number of months from the user  int n = s.nextInt();  // Initializing the principal amount (initial debt) to 100,000  double c = 100000;  // Looping through each month to calculate the debt amount  for (int i = 0; i < n; i++) {  // Calculating the new debt amount after adding 4% interest  c += c \* 0.04;  // Checking if the debt amount is not a multiple of 1000  if (c % 1000 != 0) {  // Reducing the debt amount to the nearest multiple of 1000  c -= c % 1000;  // Adding 1000 to the debt amount  c += 1000;  }  }  // Printing the final debt amount without decimal places  System.out.println("\nAmount of debt: ");  System.out.printf("%.0f\n", c);  }  } |

Output:

A white background with blue text

AI-generated content may be incorrect.

216. Write a Java program which reads an integer n and finds the number of combinations of a,b,c and d (0 ≤ a,b,c,d ≤ 9) where (a + b + c + d) equals n.

Input:

a,b,c,d,e,f separated by a single space.  
(-1,000 ≤ a,b,c,d,e,f ≤ 1,000)

*Expected Output:*

Input the number(n):

5

Number of combinations of a, b, c and d :

56

Code:

|  |
| --- |
| import java.util.Scanner;  public class Main {  public static void main(String[] args) {  // Prompting the user to input the number (n)  System.out.println("Input the number(n):");  // Creating a Scanner object for user input  Scanner s = new Scanner(System.in);  // Reading the input number (n) from the user  int c = s.nextInt();  // Calling the check method to find the number of combinations  int ans = check(c);  // Displaying the number of combinations of a, b, c, and d  System.out.println("Number of combinations of a, b, c, and d :");  System.out.println(ans);  }  // Method to check the number of combinations  static int check(int c) {  // Initializing a counter for combinations  int ctr = 0;  // Nested loops to iterate through all possible combinations of a, b, c, and d  for (int i = 0; i < 10; i++) {  for (int j = 0; j < 10; j++) {  for (int k = 0; k < 10; k++) {  for (int l = 0; l < 10; l++) {  // Checking if the sum of a, b, c, and d equals the input number (n)  if (i + j + k + l == c) {  // Incrementing the counter for valid combinations  ctr++;  }  }  }  }  }  // Returning the total number of combinations  return ctr;  }  } |

Output:

A close-up of a computer screen

AI-generated content may be incorrect.

217. Write a Java program to print the number of prime numbers less than or equal to a given integer.

Input:

n (1 ≤ n ≤ 999,999)

*Expected Output:*

Input the number(n):

1235

Number of prime numbers which are less than or equal to n.:

202

Code:

|  |
| --- |
| import java.util.Scanner;  public class Main {  public static void main(String[] args) {  // Prompting the user to input the number (n)  System.out.println("Input the number(n):");  // Creating a Scanner object for user input  Scanner s = new Scanner(System.in);  // Reading the input number (n) from the user  int c = s.nextInt();  // Calling the check method to find the number of prime numbers  int ans = check(c);  // Displaying the number of prime numbers which are less than or equal to n  System.out.println("Number of prime numbers which are less than or equal to n:");  System.out.println(ans);  }  // Method to check the number of prime numbers  static int check(int c) {  // Creating a boolean array to mark numbers as prime or not  boolean[] prime = new boolean[c + 1];  // Initializing a counter for prime numbers  int count = 0;  // Loop to mark non-prime numbers in the array  for (int i = 2; i <= Math.sqrt(c); i++) {  for (int j = i + i; j <= c; j += i) {  prime[j] = true;  }  }  // Counting the number of prime numbers  for (int i = 2; i <= c; i++) {  if (!prime[i]) {  count++;  }  }  // Returning the total number of prime numbers  return count;  }  } |

Output:

A white background with blue text

AI-generated content may be incorrect.

218. Write a Java program to compute the radius and central coordinates (x, y) of a circle constructed from three given points on the plane surface.

Input:

x1, y1, x2, y2, x3, y3 separated by a single space.

*Expected Output:*

Input x1, y1, x2, y2, x3, y3 separated by a single space:

5 6 4 8 7 9

Radius and the central coordinate:

1.821 (5.786 7.643)

Code:

|  |
| --- |
| // Importing necessary classes for input/output operations  import java.io.IOException;  import java.io.InputStreamReader;  import java.io.BufferedReader;  // Main class named "Main"  class Main {  // Main method with IOException in case of input error  public static void main(String[] args) throws IOException {  // Creating BufferedReader for efficient reading of input  BufferedReader br = new BufferedReader(new InputStreamReader(System.in));  // Prompting the user to input coordinates x1, y1, x2, y2, x3, y3 separated by a single space  System.out.println("Input x1, y1, x2, y2, x3, y3 separated by a single space:");  // Reading the input line and splitting it into an array of strings  String[] input = br.readLine().split(" ");  // Parsing the input strings into double values  double x1 = Double.parseDouble(input[0]);  double y1 = Double.parseDouble(input[1]);  double x2 = Double.parseDouble(input[2]);  double y2 = Double.parseDouble(input[3]);  double x3 = Double.parseDouble(input[4]);  double y3 = Double.parseDouble(input[5]);  // Calculating intermediate values for further computations  double A = x2 - x1;  double B = y2 - y1;  double p = (y2 \* y2 - y1 \* y1 + x2 \* x2 - x1 \* x1) / 2;  double C = x3 - x1;  double D = y3 - y1;  double q = (y3 \* y3 - y1 \* y1 + x3 \* x3 - x1 \* x1) / 2;  // Calculating the coordinates of the center (X, Y) and the radius (r) of the circle  double X = (D \* p - B \* q) / (A \* D - B \* C);  double Y = (A \* q - C \* p) / (A \* D - B \* C);  double r = Math.sqrt(Math.pow(X - x1, 2.0) + Math.pow(Y - y1, 2.0));  // Displaying the radius and the central coordinate of the circle  System.out.println("\nRadius and the central coordinate:");  System.out.printf("%.3f (%.3f %.3f)", r, X, Y);  }  } |

Output: