# LAB # 04

**LAB TASKS 1:**

public class ArraySwap {

// Function to swap two arrays of size 4

public static void swapArrays(int[] arr1, int[] arr2) {

// Check if both arrays have size 4

if (arr1.length == 4 && arr2.length == 4) {

// Swap elements by using a temporary array

for (int i = 0; i < arr1.length; i++) {

int temp = arr1[i];

arr1[i] = arr2[i];

arr2[i] = temp;

}

} else {

System.out.println("Both arrays must have a size of 4.");

}

}

public static void main(String[] args) {

// Define two arrays of size 4

int[] array1 = {1, 2, 3, 4};

int[] array2 = {5, 6, 7, 8};

System.out.println("Before swapping:");

System.out.print("Array 1: ");

printArray(array1);

System.out.print("Array 2: ");

printArray(array2);

// Swap the arrays

swapArrays(array1, array2);

System.out.println("\nAfter swapping:");

System.out.print("Array 1: ");

printArray(array1);

System.out.print("Array 2: ");

printArray(array2);

}

// Helper function to print the arrays

public static void printArray(int[] array) {

for (int i = 0; i < array.length; i++) {

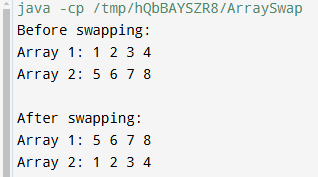
System.out.print(array[i] + " ");

}

System.out.println();

}

**OUTPUT**



**LAB TASKS 2:**

public class ArraySwap {

public static void main(String[] args) {

// Lab file owner

// Create two arrays with 4 elements each

int[] array1 = {765, 34562, 30, 4};

int[] array2 = {2345, 6890, 1237, 9878};

// Merge the two arrays

int[] mergedArray = mergeArrays(array1, array2);

// Print the merged array

System.out.print("Merged Array: ");

for (int i = 0; i < mergedArray.length; i++) {

System.out.print(mergedArray[i] + " ");

}

}

// Simple method to merge two arrays

public static int[] mergeArrays(int[] array1, int[] array2) {

// Create a new array to hold both arrays

int[] merged = new int[array1.length + array2.length];

// Copy array1 elements

System.arraycopy(array1, 0, merged, 0, array1.length);

// Copy array2 elements

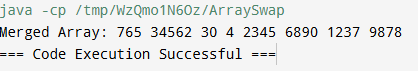
System.arraycopy(array2, 0, merged, array1.length, array2.length);

return merged;

}

}

**OUTPUT**



**LAB TASKS 3:**

import java.util.Scanner;

public class PalindromeChecker {

// Method to check if a string is a palindrome

public static boolean isPalindrome(String str) {

// Remove spaces and convert to lowercase for case-insensitive comparison

str = str.replaceAll("\\s", "").toLowerCase();

int left = 0, right = str.length() - 1;

// Check for palindrome by comparing characters from both ends

while (left < right) {

if (str.charAt(left) != str.charAt(right)) {

return false; // Not a palindrome

}

left++;

right--;

}

return true; // It's a palindrome

}

public static void main(String[] args) {

// Create a scanner object for input

Scanner scanner = new Scanner(System.in);

// Input: Number of strings to check

System.out.println("Enter the number of strings to check:");

int n = scanner.nextInt();

scanner.nextLine(); // Consume the leftover newline character

// Create an array to store the strings

String[] strings = new String[n];

// Input: The strings themselves

System.out.println("Enter the strings:");

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

strings[i] = scanner.nextLine();

}

// Check each string for palindrome property

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

if (isPalindrome(strings[i])) {

System.out.println(strings[i] + " is a palindrome.");

} else {

System.out.println(strings[i] + " is not a palindrome.");

}

}

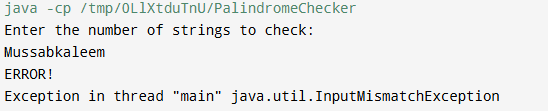
// Close the scanner

scanner.close();

}

}

**OUTPUT**



**LAB TASKS 4:**

public class EvenOddCounter {

public static void main(String[] args) {

int[] array = {1, 2, 3, 4, 5, 6}; // Example array

int[] result = countEvenOdd(array);

System.out.println("Even numbers: " + result[0]);

System.out.println("Odd numbers: " + result[1]);

}

public static int[] countEvenOdd(int[] arr) {

int evenCount = 0;

int oddCount = 0;

for (int num : arr) {

if (num % 2 == 0) {

evenCount++;

} else {

oddCount++;

}

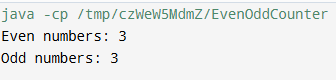
}

return new int[]{evenCount, oddCount};

}

}

**OUTPUT**



**LAB TASKS 5:**

import java.util.\*;

public class MergeAndRemoveDuplicates {

public static void main(String[] args) {

int[] array1 = {1, 2, 3, 4, 5};

int[] array2 = {3, 4, 5, 6, 7};

int[] result = mergeAndRemoveDuplicates(array1, array2);

System.out.println("Merged array without duplicates: " + Arrays.toString(result));

}

public static int[] mergeAndRemoveDuplicates(int[] arr1, int[] arr2) {

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

// Add elements from both arrays to the set

for (int num : arr1) {

set.add(num);

}

for (int num : arr2) {

set.add(num);

}

// Convert the set back to an array

int[] result = new int[set.size()];

int index = 0;

for (int num : set) {

result[index++] = num;

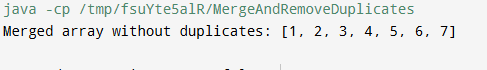
}

return result;

}

}

**OUTPUT**



**HOME TASK 1:**

import java.util.Scanner;

public class ArraySumAndMean {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

double[] numbers = new double[7]; // Allocate memory for 7 real numbers

double sum = 0;

System.out.println("Enter 7 real numbers:");

// Reading elements and calculating sum

for (int i = 0; i < numbers.length; i++) {

System.out.print("Number " + (i + 1) + ": ");

numbers[i] = scanner.nextDouble();

sum += numbers[i];

}

double mean = sum / numbers.length; // Calculate mean

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

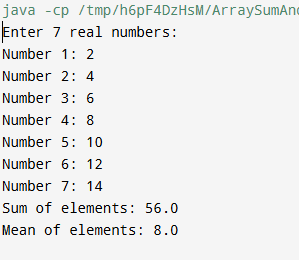
System.out.println("Mean of elements: " + mean);

scanner.close();

}

}

**OUTPUT**



**HOME TASK 2:**

import java.util.Scanner;

import java.util.Arrays;

public class ArraySumAndMean {

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

double[] numbers = new double[7];

double sum = 0;

System.out.println("Enter 7 real numbers:");

for (int i = 0; i < numbers.length; i++) {

System.out.print("Number " + (i + 1) + ": ");

numbers[i] = scanner.nextDouble();

sum += numbers[i];

}

double mean = sum / numbers.length;

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

System.out.println("Mean of elements: " + mean);

System.out.print("Enter a key to split the array: ");

double key = scanner.nextDouble();

double[][] splitArrays = splitArrayAtKey(numbers, key);

if (splitArrays != null) {

System.out.println("First part of the split array: " + Arrays.toString(splitArrays[0]));

System.out.println("Second part of the split array: " + Arrays.toString(splitArrays[1]));

} else {

System.out.println("Key not found in the array. No split performed.");

}

scanner.close();

}

public static double[][] splitArrayAtKey(double[] arr, double key) {

// Search for the key in the array

int index = -1;

for (int i = 0; i < arr.length; i++) {

if (arr[i] == key) {

index = i;

break;

}

}

// If key is not found, return null

if (index == -1) {

return null;

}

// Split the array into two parts at the found index

double[] firstPart = Arrays.copyOfRange(arr, 0, index + 1);

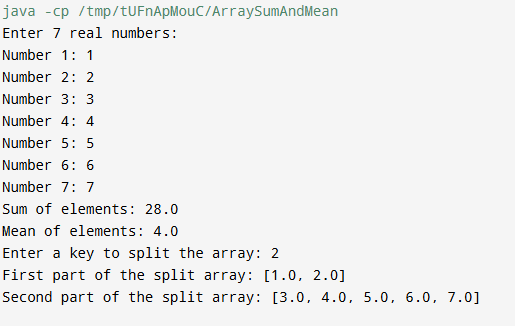
double[] secondPart = Arrays.copyOfRange(arr, index + 1, arr.length);

return new double[][] { firstPart, secondPart };

}

}

**OUTPUT**



**HOME TASK 3:**

import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

public class CombinationSum {

public static void main(String[] args) {

int[] numbers = {10, 1, 2, 7, 6, 5};

int target = 8;

List<List<Integer>> result = findCombinations(numbers, target);

System.out.println("Unique combinations that add up to " + target + ": " + result);

}

public static List<List<Integer>> findCombinations(int[] arr, int target) {

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

Arrays.sort(arr); // Sort to help with skipping duplicates

backtrack(arr, target, 0, new ArrayList<>(), result);

return result;

}

private static void backtrack(int[] arr, int target, int start, List<Integer> currentCombination, List<List<Integer>> result) {

// If the target is reached, add a copy of the current combination to the result

if (target == 0) {

result.add(new ArrayList<>(currentCombination));

return;

}

for (int i = start; i < arr.length; i++) {

// If the current number exceeds the target, break (since the array is sorted)

if (arr[i] > target) break;

// Skip duplicates

if (i > start && arr[i] == arr[i - 1]) continue;

// Choose the current number

currentCombination.add(arr[i]);

// Recurse with the updated target and next index

backtrack(arr, target - arr[i], i + 1, currentCombination, result);

// Undo the choice (backtrack)

currentCombination.remove(currentCombination.size() - 1);

}

}

}

**OUTPUT**



**HOME TASK 4:**

public class MissingNumberFinder {

public static void main(String[] args) {

int[] numbers = {3, 0, 1}; // Example array

int missingNumber = findMissingNumber(numbers);

System.out.println("The missing number is: " + missingNumber);

}

public static int findMissingNumber(int[] arr) {

int n = arr.length;

int expectedSum = n \* (n + 1) / 2;

int actualSum = 0;

// Calculate the actual sum of the array elements

for (int num : arr) {

actualSum += num;

}

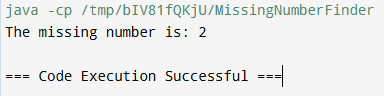
// The difference between expected and actual sum is the missing number

return expectedSum - actualSum;

}

}

**OUTPUT**



**HOME TASK 5:**

import java.util.Arrays;

public class ZigzagSort {

public static void main(String[] args) {

int[] arr = {4, 3, 7, 8, 6, 2, 1}; // Example array

zigzagSort(arr);

System.out.println("Array in zigzag pattern: " + Arrays.toString(arr));

}

public static void zigzagSort(int[] arr) {

for (int i = 0; i < arr.length - 1; i++) {

if (i % 2 == 0) {

// If the index is even, ensure arr[i] < arr[i+1]

if (arr[i] > arr[i + 1]) {

swap(arr, i, i + 1);

}

} else {

// If the index is odd, ensure arr[i] > arr[i+1]

if (arr[i] < arr[i + 1]) {

swap(arr, i, i + 1);

}

}

}

}

private static void swap(int[] arr, int i, int j) {

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

**OUTPUT**

