**21-11-2024**

**CODING PRACTICE PROBLEMS**

**1.Valid Palindrome**

import java.util.\*;

public class Problem1 {

public static boolean isPalindrome(String s) {

StringBuilder filtered = new StringBuilder();

for (char c : s.toCharArray()) {

if (Character.isLetterOrDigit(c)) {

filtered.append(Character.toLowerCase(c));

}

}

String cleaned = filtered.toString();

String reversed = filtered.reverse().toString();

return cleaned.equals(reversed);

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter a string to check if it's a palindrome:");

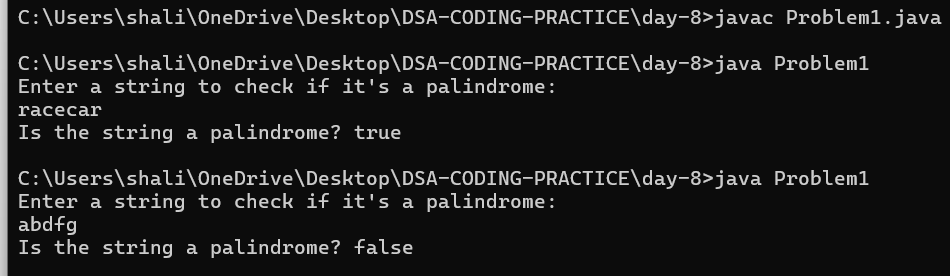
String s = sc.nextLine();

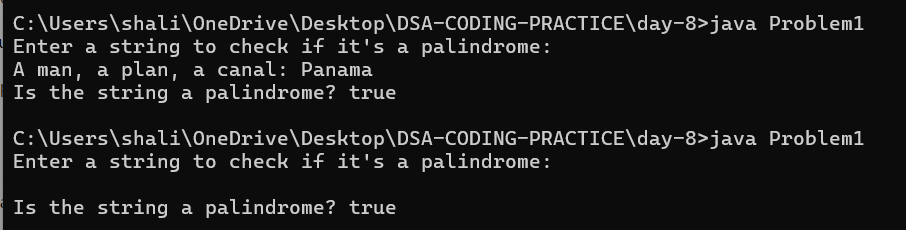
boolean result = isPalindrome(s);

System.out.println("Is the string a palindrome? " + result);

}

}



****

**2.Is Subsequence**

import java.util.\*;

public class Problem2 {

public static boolean isSubsequence(String s, String t) {

int sIndex = 0, tIndex = 0;

while (sIndex < s.length() && tIndex < t.length()) {

if (s.charAt(sIndex) == t.charAt(tIndex)) {

sIndex++;

}

tIndex++;

}

return sIndex == s.length();

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter string s:");

String s = sc.nextLine();

System.out.println("Enter string t:");

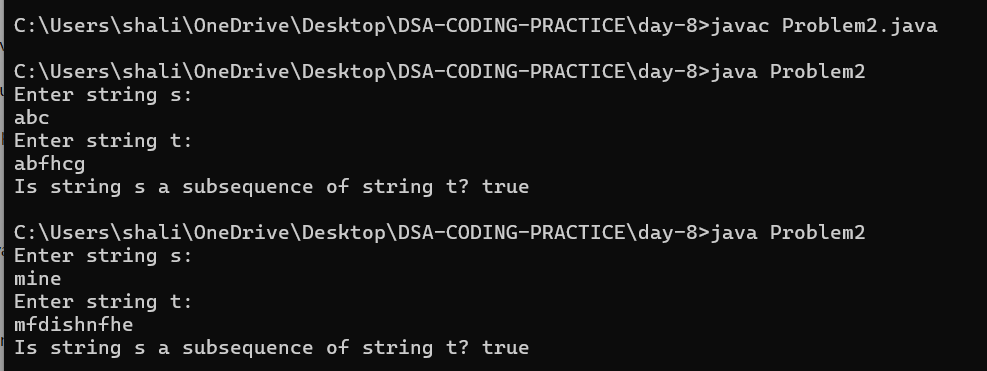
String t = sc.nextLine();

boolean result = isSubsequence(s, t);

System.out.println("Is string s a subsequence of string t? " + result);

}

}



**3.Two Sum II**

import java.util.\*;

public class Problem3 {

public static int[] twoSum(int[] numbers, int target) {

int left = 0, right = numbers.length - 1;

while (left < right) {

int sum = numbers[left] + numbers[right];

if (sum == target) {

return new int[]{left + 1, right + 1};

} else if (sum < target) {

left++;

} else {

right--;

}

}

return new int[0];

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the size of the array:");

int n = sc.nextInt();

int[] numbers = new int[n];

System.out.println("Enter the elements of the array:");

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

numbers[i] = sc.nextInt();

}

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

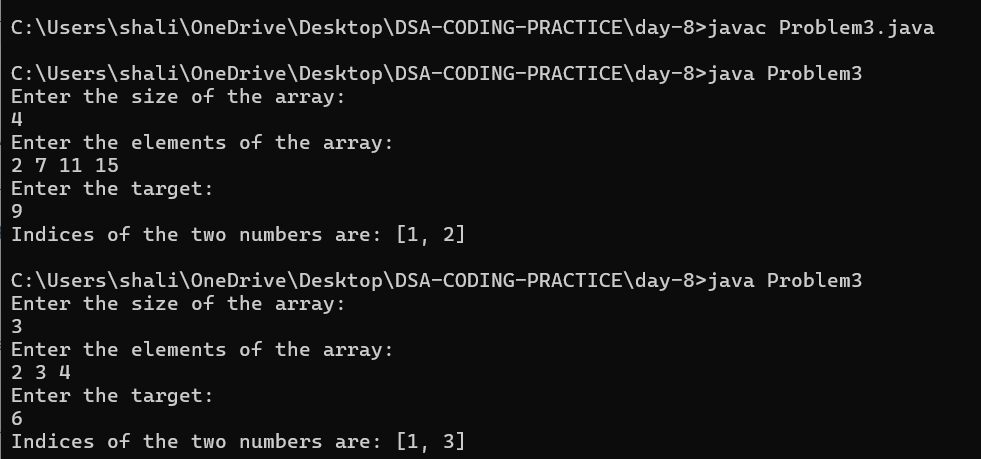
int target = sc.nextInt();

int[] result = twoSum(numbers, target);

System.out.println("Indices of the two numbers are: " + Arrays.toString(result));

}

}



**4. Container with most water**

import java.util.\*;

public class Problem4 {

public static int maxArea(int[] height) {

int left = 0, right = height.length - 1;

int maxArea = 0;

while (left < right) {

int width = right - left;

int minHeight = Math.min(height[left], height[right]);

maxArea = Math.max(maxArea, width \* minHeight);

if (height[left] < height[right]) {

left++;

} else {

right--;

}

}

return maxArea;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the size of the array:");

int n = sc.nextInt();

int[] height = new int[n];

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

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

height[i] = sc.nextInt();

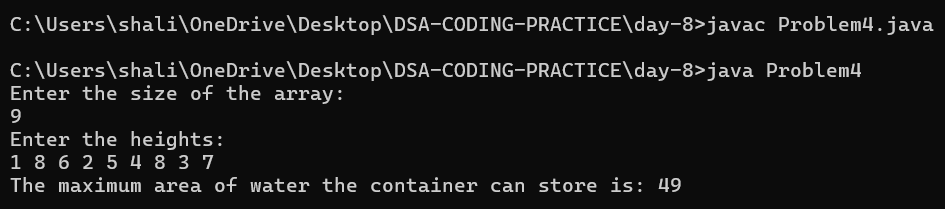
}

int result = maxArea(height);

System.out.println("The maximum area of water the container can store is: " + result);

}

}



**5.3Sum**

import java.util.\*;

public class Problem5 {

public static List<List<Integer>> threeSum(int[] nums) {

Arrays.sort(nums);

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

for (int i = 0; i < nums.length - 2; i++) {

if (i > 0 && nums[i] == nums[i - 1]) continue;

int left = i + 1, right = nums.length - 1;

while (left < right) {

int sum = nums[i] + nums[left] + nums[right];

if (sum == 0) {

result.add(Arrays.asList(nums[i], nums[left], nums[right]));

while (left < right && nums[left] == nums[left + 1]) left++;

while (left < right && nums[right] == nums[right - 1]) right--;

left++;

right--;

} else if (sum < 0) {

left++;

} else {

right--;

}

}

}

return result;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the size of the array:");

int n = sc.nextInt();

int[] nums = new int[n];

System.out.println("Enter the elements of the array:");

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

nums[i] = sc.nextInt();

}

List<List<Integer>> result = threeSum(nums);

System.out.println("The triplets that sum up to zero are:");

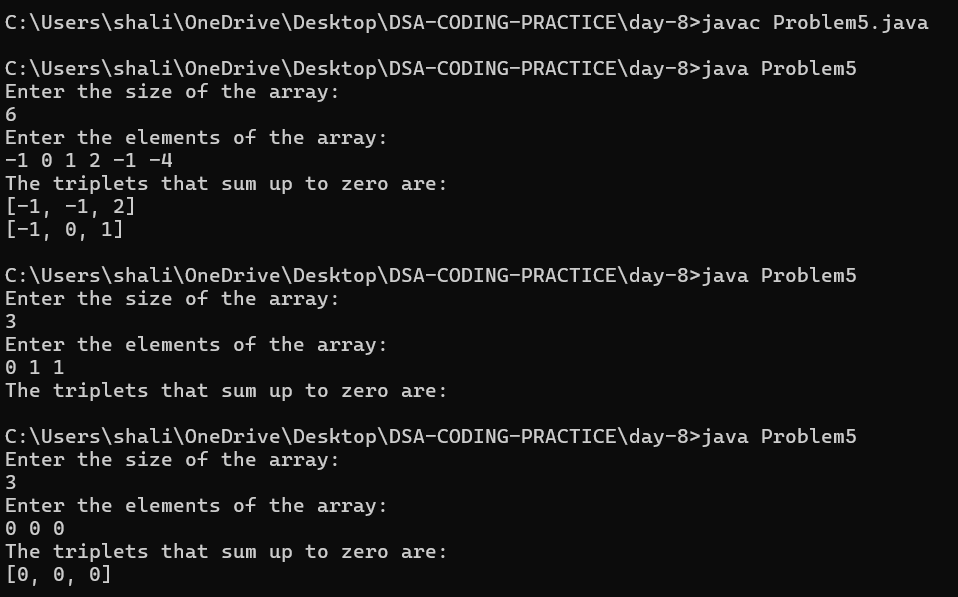
for (List<Integer> triplet : result) {

System.out.println(triplet);

}

}

}



**6.Minimum size subarray sum**

import java.util.\*;

public class Problem6 {

public static int minSubArrayLen(int target, int[] nums) {

int left = 0, sum = 0, minLength = Integer.MAX\_VALUE;

for (int right = 0; right < nums.length; right++) {

sum += nums[right];

while (sum >= target) {

minLength = Math.min(minLength, right - left + 1);

sum -= nums[left];

left++;

}

}

return minLength == Integer.MAX\_VALUE ? 0 : minLength;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

int target = sc.nextInt();

System.out.println("Enter the size of the array:");

int n = sc.nextInt();

int[] nums = new int[n];

System.out.println("Enter the elements of the array:");

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

nums[i] = sc.nextInt();

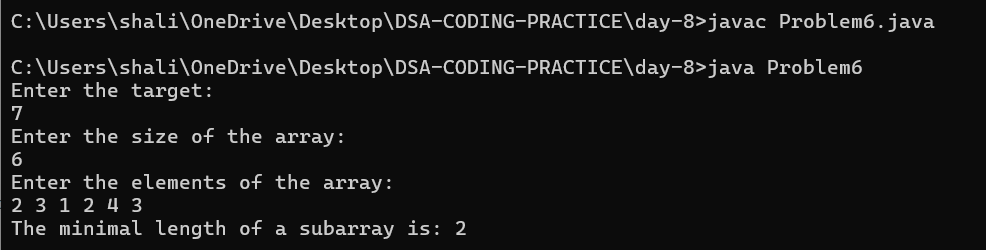
}

int result = minSubArrayLen(target, nums);

System.out.println("The minimal length of a subarray is: " + result);

}

}



**7.Longest Substring without repeating characters**

import java.util.\*;

public class Problem7 {

public static int lengthOfLongestSubstring(String s) {

Set<Character> seen = new HashSet<>();

int left = 0, maxLength = 0;

for (int right = 0; right < s.length(); right++) {

while (seen.contains(s.charAt(right))) {

seen.remove(s.charAt(left));

left++;

}

seen.add(s.charAt(right));

maxLength = Math.max(maxLength, right - left + 1);

}

return maxLength;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

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

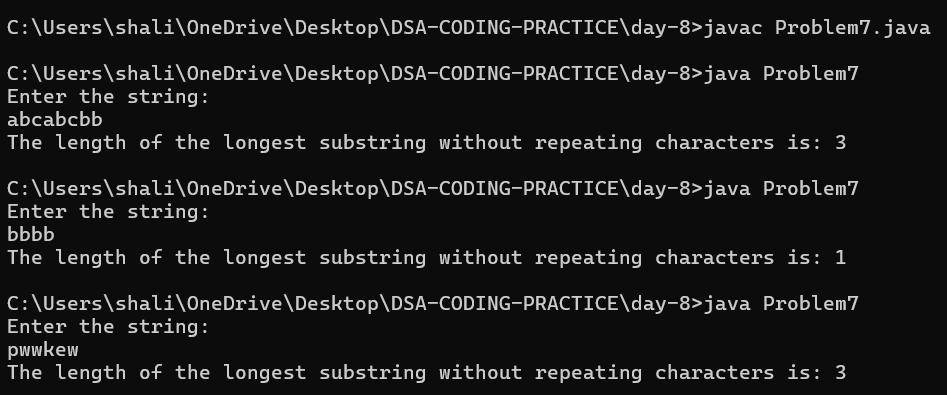
String s = sc.nextLine();

int result = lengthOfLongestSubstring(s);

System.out.println("The length of the longest substring without repeating characters is: " + result);

}

}



**8.Substring with concatenation of all words**

import java.util.\*;

public class Problem8 {

public static List<Integer> findSubstring(String s, String[] words) {

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

if (s == null || s.length() == 0 || words == null || words.length == 0) {

return result;

}

int wordLength = words[0].length();

int totalWords = words.length;

int substringLength = wordLength \* totalWords;

Map<String, Integer> wordCount = new HashMap<>();

for (String word : words) {

wordCount.put(word, wordCount.getOrDefault(word, 0) + 1);

}

for (int i = 0; i <= s.length() - substringLength; i++) {

Map<String, Integer> seenWords = new HashMap<>();

int j = 0;

while (j < totalWords) {

int wordIndex = i + j \* wordLength;

String currentWord = s.substring(wordIndex, wordIndex + wordLength);

if (!wordCount.containsKey(currentWord)) {

break;

}

seenWords.put(currentWord, seenWords.getOrDefault(currentWord, 0) + 1);

if (seenWords.get(currentWord) > wordCount.get(currentWord)) {

break;

}

j++;

}

if (j == totalWords) {

result.add(i);

}

}

return result;

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the string s:");

String s = sc.nextLine();

System.out.println("Enter the number of words:");

int n = sc.nextInt();

sc.nextLine(); // Consume the newline

String[] words = new String[n];

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

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

words[i] = sc.nextLine();

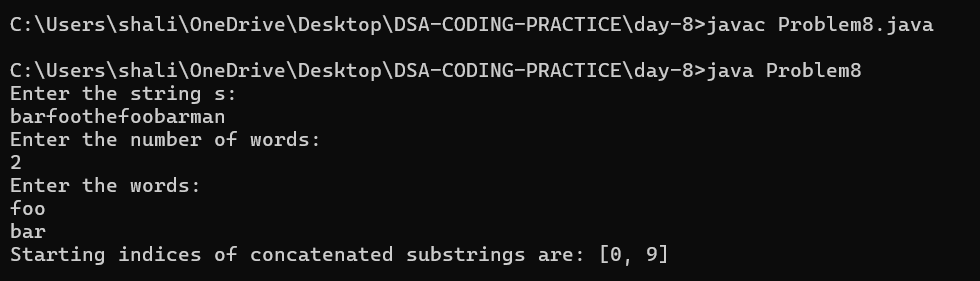
}

List<Integer> result = findSubstring(s, words);

System.out.println("Starting indices of concatenated substrings are: " + result);

}

}



**9.Minimum window substring**

import java.util.\*;

public class Problem9 {

public static String minWindow(String s, String t) {

if (s == null || t == null || s.length() == 0 || t.length() == 0) {

return "";

}

Map<Character, Integer> tFrequency = new HashMap<>();

for (char c : t.toCharArray()) {

tFrequency.put(c, tFrequency.getOrDefault(c, 0) + 1);

}

int left = 0, right = 0, matched = 0, minLength = Integer.MAX\_VALUE;

int start = 0;

Map<Character, Integer> windowFrequency = new HashMap<>();

while (right < s.length()) {

char rightChar = s.charAt(right);

windowFrequency.put(rightChar, windowFrequency.getOrDefault(rightChar, 0) + 1);

if (tFrequency.containsKey(rightChar) && windowFrequency.get(rightChar) <= tFrequency.get(rightChar)) {

matched++;

}

while (matched == t.length()) {

if (right - left + 1 < minLength) {

minLength = right - left + 1;

start = left;

}

char leftChar = s.charAt(left);

windowFrequency.put(leftChar, windowFrequency.get(leftChar) - 1);

if (tFrequency.containsKey(leftChar) && windowFrequency.get(leftChar) < tFrequency.get(leftChar)) {

matched--;

}

left++;

}

right++;

}

return minLength == Integer.MAX\_VALUE ? "" : s.substring(start, start + minLength);

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the string s:");

String s = sc.nextLine();

System.out.println("Enter the string t:");

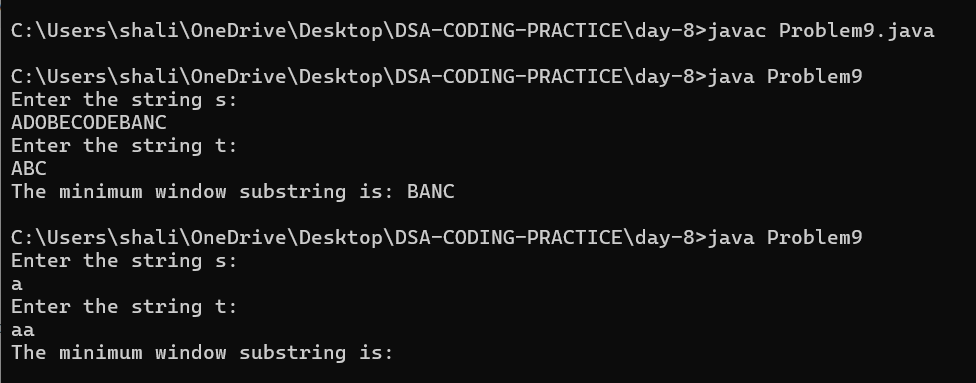
String t = sc.nextLine();

String result = minWindow(s, t);

System.out.println("The minimum window substring is: " + result);

}

}



**10.Valid Parantheses**

import java.util.\*;

public class Problem10 {

public static boolean isValid(String s) {

Stack<Character> stack = new Stack<>();

for (char ch : s.toCharArray()) {

if (ch == '(' || ch == '{' || ch == '[') {

stack.push(ch);

} else {

if (stack.isEmpty()) {

return false;

}

char top = stack.pop();

if ((ch == ')' && top != '(') ||

(ch == '}' && top != '{') ||

(ch == ']' && top != '[')) {

return false;

}

}

}

return stack.isEmpty();

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the string s:");

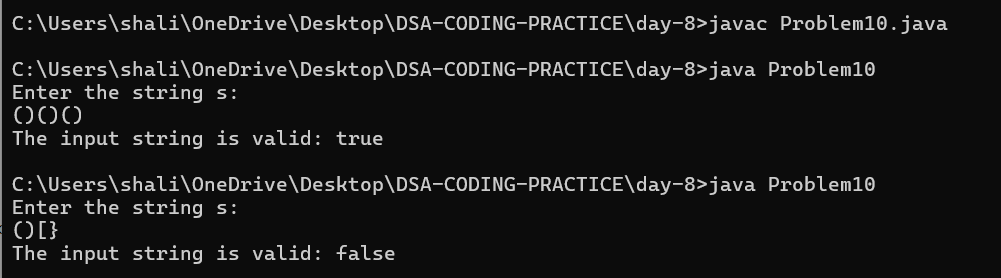
String s = sc.nextLine();

boolean result = isValid(s);

System.out.println("The input string is valid: " + result);

}

}



**11.Simplify Path**

import java.util.\*;

public class Problem11 {

public static String simplifyPath(String path) {

Stack<String> stack = new Stack<>();

String[] dirs = path.split("/");

for (String dir : dirs) {

if (dir.equals("..")) {

if (!stack.isEmpty()) {

stack.pop();

}

} else if (!dir.equals("") && !dir.equals(".")) {

stack.push(dir);

}

}

StringBuilder result = new StringBuilder();

if (stack.isEmpty()) {

result.append("/");

} else {

while (!stack.isEmpty()) {

result.insert(0, "/" + stack.pop());

}

}

return result.toString();

}

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.println("Enter the absolute path:");

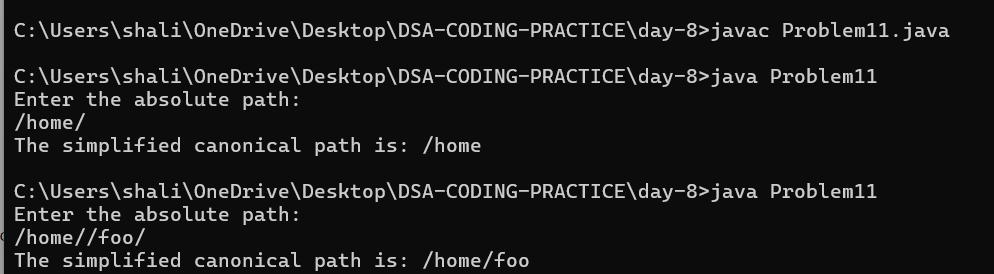
String path = sc.nextLine();

String result = simplifyPath(path);

System.out.println("The simplified canonical path is: " + result);

}

}



**12.Min Stack**

import java.util.\*;

class Problem12 {

private Stack<Integer> stack;

private Stack<Integer> minStack;

public Problem12() {

stack = new Stack<>();

minStack = new Stack<>();

}

public void push(int val) {

stack.push(val);

if (minStack.isEmpty() || val <= minStack.peek()) {

minStack.push(val);

}

}

public void pop() {

if (stack.peek().equals(minStack.peek())) {

minStack.pop();

}

stack.pop();

}

public int top() {

return stack.peek();

}

public int getMin() {

return minStack.peek();

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

Problem12 minStack = new Problem12();

while (true) {

System.out.println("Enter operation (push, pop, top, getMin, or exit): ");

String operation = scanner.nextLine().trim().toLowerCase();

if (operation.equals("exit")) {

break;

}

switch (operation) {

case "push":

System.out.println("Enter value to push: ");

int pushVal = scanner.nextInt();

minStack.push(pushVal);

System.out.println("Pushed: " + pushVal);

scanner.nextLine(); // Consume the newline

break;

case "pop":

minStack.pop();

System.out.println("Popped top element.");

break;

case "top":

System.out.println("Top element: " + minStack.top());

break;

case "getmin":

System.out.println("Minimum element: " + minStack.getMin());

break;

default:

System.out.println("Invalid operation. Try again.");

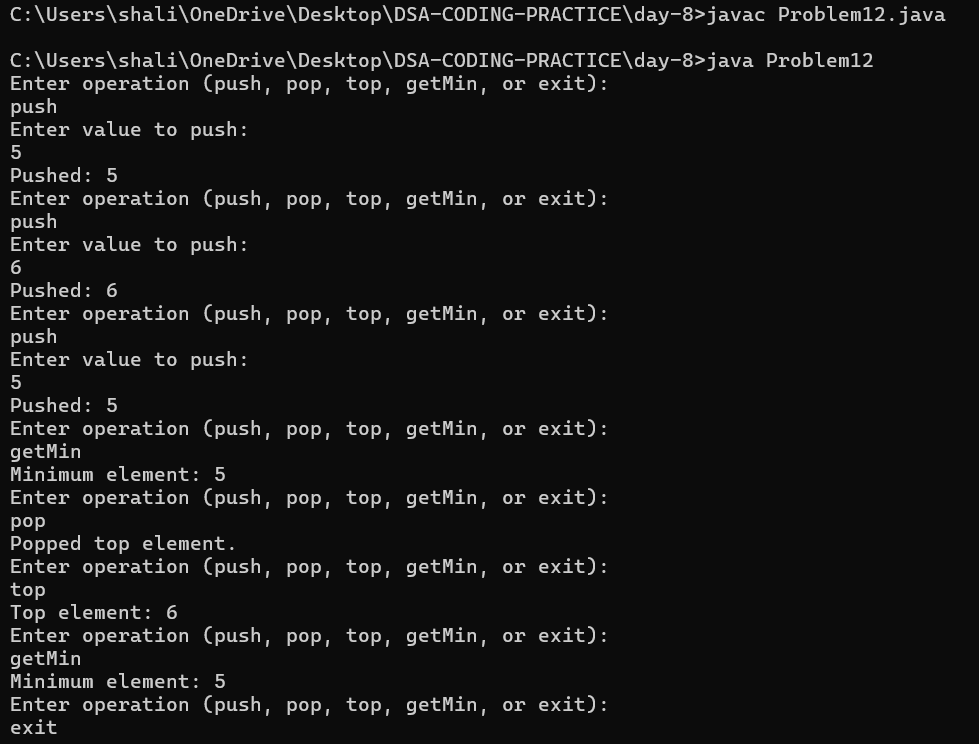
}

}

scanner.close();

}

}



**13.Evaluate reverse Polish notations**

import java.util.\*;

class Problem13 {

public int evalRPN(String[] tokens) {

Stack<Integer> stack = new Stack<>();

for (String token : tokens) {

if (token.equals("+")) {

int b = stack.pop();

int a = stack.pop();

stack.push(a + b);

} else if (token.equals("-")) {

int b = stack.pop();

int a = stack.pop();

stack.push(a - b);

} else if (token.equals("\*")) {

int b = stack.pop();

int a = stack.pop();

stack.push(a \* b);

} else if (token.equals("/")) {

int b = stack.pop();

int a = stack.pop();

stack.push(a / b); // Integer division truncates toward zero

} else {

stack.push(Integer.parseInt(token));

}

}

return stack.pop();

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

Problem13 obj = new Problem13();

System.out.println("Enter the tokens (space-separated): ");

String input = scanner.nextLine();

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

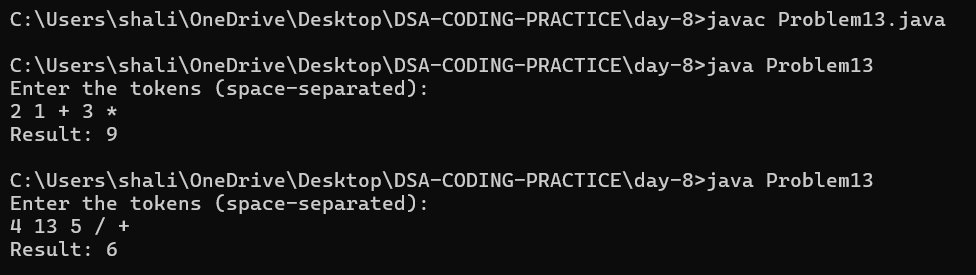
int result = obj.evalRPN(tokens);

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

scanner.close();

}

}



**14.Basic calculator**

import java.util.\*;

class Problem14 {

public int calculate(String s) {

Stack<Integer> stack = new Stack<>();

int currentNumber = 0;

int result = 0;

int sign = 1; // 1 for positive, -1 for negative

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

char ch = s.charAt(i);

if (Character.isDigit(ch)) {

currentNumber = currentNumber \* 10 + (ch - '0');

}

if (ch == '+' || ch == '-' || ch == '(' || ch == ')' || i == s.length() - 1) {

if (ch == '(') {

stack.push(result);

stack.push(sign);

result = 0;

sign = 1;

} else if (ch == ')') {

result += sign \* currentNumber;

currentNumber = 0;

result \*= stack.pop();

result += stack.pop();

} else if (ch == '+' || ch == '-') {

result += sign \* currentNumber;

currentNumber = 0;

sign = (ch == '+') ? 1 : -1;

}

}

}

result += sign \* currentNumber;

return result;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

Problem14 obj = new Problem14();

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

String input = scanner.nextLine();

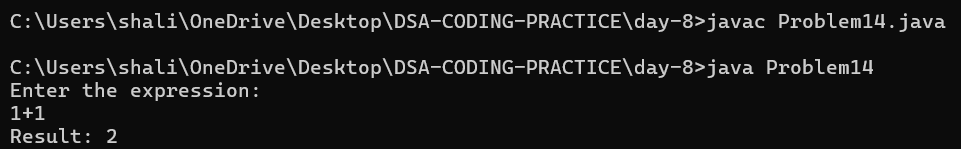
int result = obj.calculate(input);

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

scanner.close();

}

}



**15.Search insert position**

import java.util.\*;

class Problem15 {

public int searchInsert(int[] nums, int target) {

int left = 0, right = nums.length - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

if (nums[mid] == target) {

return mid;

} else if (nums[mid] < target) {

left = mid + 1;

} else {

right = mid - 1;

}

}

return left; // The position where the target would be inserted

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

Problem15 obj = new Problem15();

System.out.println("Enter the sorted array (space-separated): ");

String[] input = scanner.nextLine().split(" ");

int[] nums = new int[input.length];

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

nums[i] = Integer.parseInt(input[i]);

}

System.out.println("Enter the target value: ");

int target = scanner.nextInt();

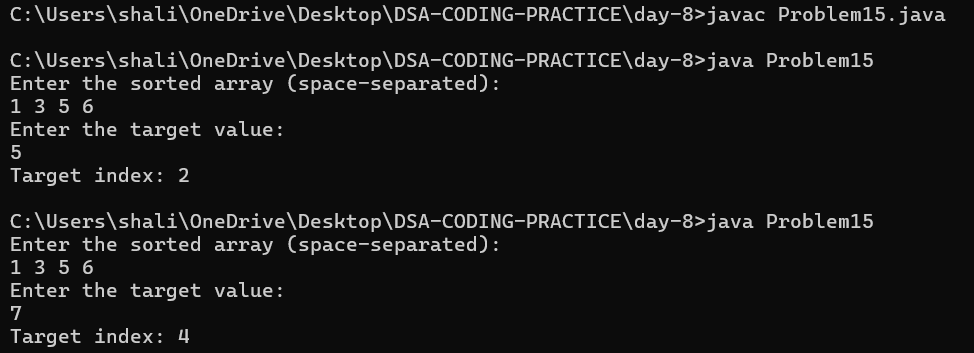
int result = obj.searchInsert(nums, target);

System.out.println("Target index: " + result);

scanner.close();

}

}



**16.Search a 2D Matrix**

import java.util.\*;

class Problem16 {

public boolean searchMatrix(int[][] matrix, int target) {

int m = matrix.length;

int n = matrix[0].length;

int left = 0, right = m \* n - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

int midValue = matrix[mid / n][mid % n];

if (midValue == target) {

return true;

} else if (midValue < target) {

left = mid + 1;

} else {

right = mid - 1;

}

}

return false;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

Problem16 obj = new Problem16();

System.out.println("Enter the number of rows in the matrix: ");

int m = scanner.nextInt();

System.out.println("Enter the number of columns in the matrix: ");

int n = scanner.nextInt();

int[][] matrix = new int[m][n];

System.out.println("Enter the matrix values row by row: ");

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

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

matrix[i][j] = scanner.nextInt();

}

}

System.out.println("Enter the target value: ");

int target = scanner.nextInt();

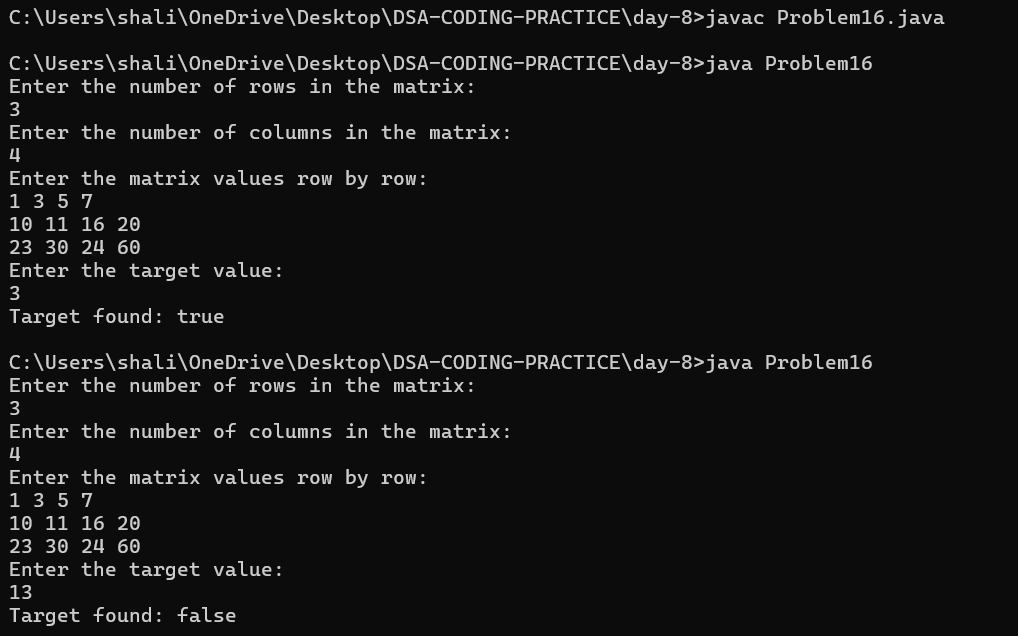
boolean result = obj.searchMatrix(matrix, target);

System.out.println("Target found: " + result);

scanner.close();

}

}



**17.Find Peak Element**

import java.util.\*;

class Problem17 {

public int findPeakElement(int[] nums) {

int left = 0, right = nums.length - 1;

while (left < right) {

int mid = left + (right - left) / 2;

if (nums[mid] > nums[mid + 1]) {

right = mid;

} else {

left = mid + 1;

}

}

return left;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

Problem17 obj = new Problem17();

System.out.println("Enter the number of elements in the array: ");

int n = scanner.nextInt();

int[] nums = new int[n];

System.out.println("Enter the array elements: ");

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

nums[i] = scanner.nextInt();

}

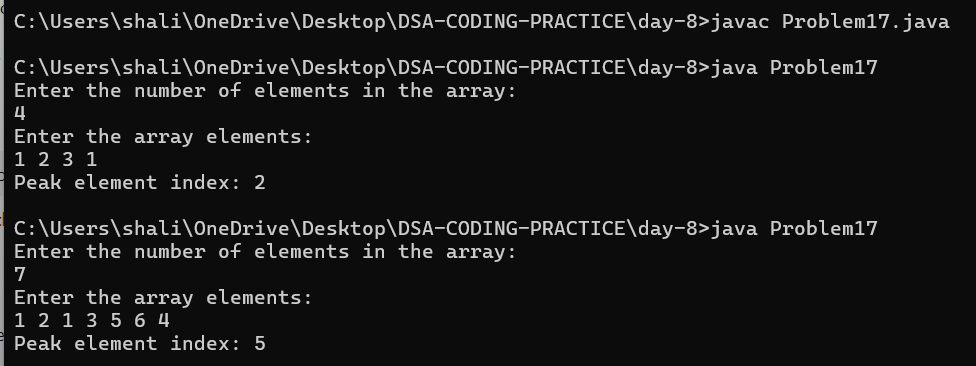
int peakIndex = obj.findPeakElement(nums);

System.out.println("Peak element index: " + peakIndex);

scanner.close();

}

}



**18.Search in a rotated Sorted Array**

import java.util.\*;

class Problem18 {

public int search(int[] nums, int target) {

int left = 0, right = nums.length - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

if (nums[mid] == target) {

return mid;

}

if (nums[left] <= nums[mid]) {

if (nums[left] <= target && target < nums[mid]) {

right = mid - 1;

} else {

left = mid + 1;

}

} else {

if (nums[mid] < target && target <= nums[right]) {

left = mid + 1;

} else {

right = mid - 1;

}

}

}

return -1;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

Problem18 obj = new Problem18();

System.out.println("Enter the number of elements in the array: ");

int n = scanner.nextInt();

int[] nums = new int[n];

System.out.println("Enter the array elements: ");

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

nums[i] = scanner.nextInt();

}

System.out.println("Enter the target value: ");

int target = scanner.nextInt();

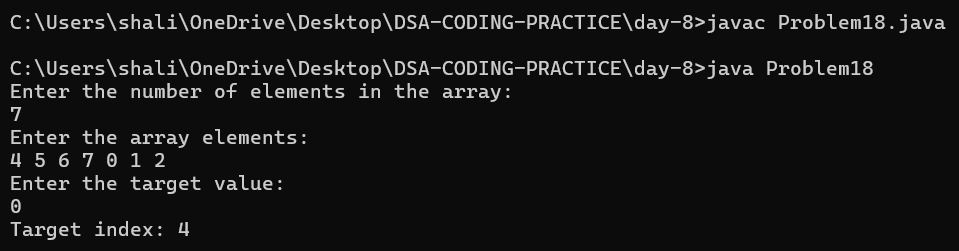
int result = obj.search(nums, target);

System.out.println("Target index: " + result);

scanner.close();

}

}



**19.Find first and last position of element in Sorted Array**

import java.util.\*;

class Problem19 {

public int[] searchRange(int[] nums, int target) {

int[] result = {-1, -1};

result[0] = binarySearch(nums, target, true);

if (result[0] == -1) {

return result; // Target not found

}

result[1] = binarySearch(nums, target, false);

return result;

}

private int binarySearch(int[] nums, int target, boolean left) {

int leftIndex = 0, rightIndex = nums.length - 1, result = -1;

while (leftIndex <= rightIndex) {

int mid = leftIndex + (rightIndex - leftIndex) / 2;

if (nums[mid] == target) {

result = mid;

if (left) {

rightIndex = mid - 1;

} else {

leftIndex = mid + 1;

}

} else if (nums[mid] < target) {

leftIndex = mid + 1;

} else {

rightIndex = mid - 1;

}

}

return result;

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

Problem19 obj = new Problem19();

System.out.println("Enter the number of elements in the array: ");

int n = scanner.nextInt();

int[] nums = new int[n];

System.out.println("Enter the array elements: ");

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

nums[i] = scanner.nextInt();

}

System.out.println("Enter the target value: ");

int target = scanner.nextInt();

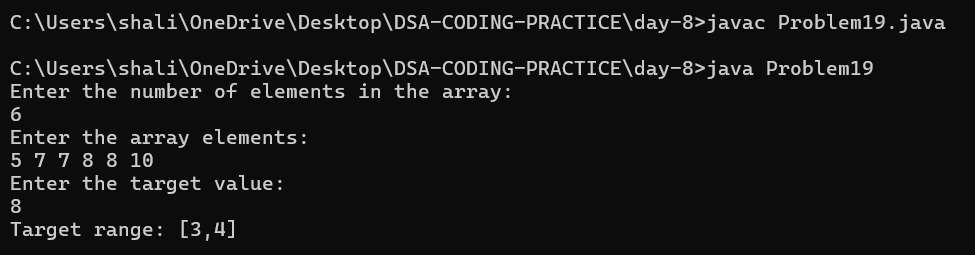
int[] result = obj.searchRange(nums, target);

System.out.println("Target range: [" + result[0] + "," + result[1] + "]");

scanner.close();

}

}



**20.Find Minimum in Rotated Sorted array**

import java.util.\*;

class Problem20 {

public int findMin(int[] nums) {

int left = 0, right = nums.length - 1;

while (left < right) {

int mid = left + (right - left) / 2;

if (nums[mid] > nums[right]) {

left = mid + 1;

} else {

right = mid;

}

}

return nums[left];

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

Problem20 obj = new Problem20();

System.out.println("Enter the number of elements in the array: ");

int n = scanner.nextInt();

int[] nums = new int[n];

System.out.println("Enter the array elements: ");

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

nums[i] = scanner.nextInt();

}

int minElement = obj.findMin(nums);

System.out.println("The minimum element in the array is: " + minElement);

scanner.close();

}

}

