21/11/2024 DSA PRACTICE 8

1.Valid Palindrome

```
import java.util.Scanner;


class ValidPalindrome{
    public boolean isPalindrome(String s) {
        int i = 0, j = s.length() - 1;
        while (i < j) {
            if (!Character.isLetterOrDigit(s.charAt(i))) {
                i++;
            } else if (!Character.isLetterOrDigit(s.charAt(j))) {
                j--;
            } else if (Character.toLowerCase(s.charAt(i)) == Character.toLowerCase(s.charAt(j))) {
                i++;
                j--;
            } else {
                return false;
            }
        }
        return true;
    }


    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
```

```
        System.out.println("Enter a string to check if it's a palindrome:");
        String input = scanner.nextLine();

        ValidPalindrome solution = new ValidPalindrome();
        if (solution.isPalindrome(input)) {
            System.out.println("The string is a palindrome.");
        } else {
            System.out.println("The string is not a palindrome.");
        }

        scanner.close();
    }
}
```

Time Complexity : O(n)

```
C:\Users\POOJA\Documents\SDE\DSA_Practice8>javac ValidPalindrome.java

C:\Users\POOJA\Documents\SDE\DSA_Practice8>java ValidPalindrome
Enter a string to check if it's a palindrome:
race a car
The string is not a palindrome.
```

2. Is Subsequence

```
import java.util.Scanner;

class Subsequence {
    public boolean isSubsequence(String s, String t) {
        int n = s.length();
        int m = t.length();
```

```
        int i = 0, j = 0;

        while (i < n && j < m) {
            if (s.charAt(i) == t.charAt(j)) {
                i++;
            }
            j++;
        }

        return i == n;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter the first string (s):");
        String s = scanner.nextLine();

        System.out.println("Enter the second string (t):");
        String t = scanner.nextLine();

        Subsequence solution = new Subsequence();
        boolean result = solution.isSubsequence(s, t);

        if (result) {
            System.out.println("\"" + s + "\" is a subsequence of \"" + t + "\".");
        } else {
            System.out.println("\"" + s + "\" is not a subsequence of \"" + t + "\".");
```

```
        }


        scanner.close();
    }
}
```

Time Complexity : O(n)

```
C:\Users\POOJA\Documents\SDE\DSA_Practice8>javac Subsequence.java

C:\Users\POOJA\Documents\SDE\DSA_Practice8>java Subsequence
Enter the first string (s):
abc
Enter the second string (t):
ahbgdc
"abc" is a subsequence of "ahbgdc".
```

3.Two Sum 2

```
import java.util.Scanner;

import java.util.Arrays;


class TwoSum2 {
    public int[] twoSum(int[] numbers, int target) {
        int left = 0;
        int right = numbers.length - 1;


        while (left < right) {
            int curr_sum = numbers[left] + numbers[right];
            if (curr_sum == target) {
                return new int[] {left, right};
            } else if (curr_sum < target) {
                left++;
```

```
            } else {
                right--;
            }
        }
        return new int[] {-1, -1}; // Return this if no solution exists
}

public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter the size of the array:");
        int n = scanner.nextInt();

        int[] numbers = new int[n];
        System.out.println("Enter " + n + " sorted integers:");
        for (int i = 0; i < n; i++) {
            numbers[i] = scanner.nextInt();
        }

        System.out.println("Enter the target value:");
        int target = scanner.nextInt();

        TwoSum2 solution = new TwoSum2();
        int[] result = solution.twoSum(numbers, target);

        if (result[0] == -1) {
            System.out.println("No two numbers found with the target sum.");
        } else {
```

```
                System.out.println("Indices of numbers that add up to the target: " +
Arrays.toString(result));
            }

            scanner.close();
        }
}
```

Time Complexity : O(n)

```
C:\Users\POOJA\Documents\SDE\DSA_Practice8>javac TwoSum2.java

C:\Users\POOJA\Documents\SDE\DSA_Practice8>java TwoSum2
Enter the size of the array:
4
Enter 4 sorted integers:
2 7 11 15
Enter the target value:
9
Indices of numbers that add up to the target: [0, 1]
```

4.Container with most water

```
import java.util.Scanner;


class MostWater {
    public int maxArea(int[] height) {
        int n = height.length;
        int left = 0;
        int right = n - 1;
        int max_area = 0;

        while (left < right) {
            int current_area = Math.min(height[left], height[right]) * (right - left);
            max_area = Math.max(max_area, current_area);
```

```
            if (height[left] < height[right]) {
                left++;
            } else {
                right--;
            }
        }
        return max_area;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter the number of elements in the array:");
        int n = scanner.nextInt();

        int[] height = new int[n];
        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < n; i++) {
            height[i] = scanner.nextInt();
        }

        MostWater solution = new MostWater();
        int result = solution.maxArea(height);

        System.out.println("max area : " + result);

        scanner.close();
```

```
    }
}
```

Time Complexity : O(n)

```
C:\Users\POOJA\Documents\SDE\DSA_Practice8>javac MostWater.java

C:\Users\POOJA\Documents\SDE\DSA_Practice8>java MostWater
Enter the number of elements in the array:
9
Enter the elements of the array:
1 8 6 2 5 4 8 3 7
max area : 49
```

5. 3Sum

```
import java.util.ArrayList;

import java.util.Arrays;

import java.util.List;

import java.util.Scanner;


class ThreeSum {

    public 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;

            int 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 scanner = new Scanner(System.in);

        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 elements of the array:");
        for (int i = 0; i < n; i++) {
            nums[i] = scanner.nextInt();
        }

        ThreeSum solution = new ThreeSum();
        List<List<Integer>> triplets = solution.threeSum(nums);

        System.out.println(triplets);

        scanner.close();
    }
}
```

Time Complexity : O(n^2)

```
C:\Users\POOJA\Documents\SDE\DSA_Practice8>javac ThreeSum.java

C:\Users\POOJA\Documents\SDE\DSA_Practice8>java ThreeSum
Enter the number of elements in the array:
6
Enter the elements of the array:
-1 0 1 2 -1 -4
[[-1, -1, 2], [-1, 0, 1]]
```

6. Minimum Subarray Sum

```
import java.util.Scanner;

class MinSubArray {
    public int minSubArrayLen(int target, int[] nums) {
        int n = nums.length;
        int left = 0, sum = 0, minLength = Integer.MAX_VALUE;

        for (int right = 0; right < n; 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 scanner = new Scanner(System.in);

        System.out.print("Enter the target value: ");
        int target = scanner.nextInt();

        System.out.print("Enter the length of the array: ");
        int n = scanner.nextInt();
```

```
        int[] nums = new int[n];
        System.out.print("Enter the elements of the array: ");
        for (int i = 0; i < n; i++) {
            nums[i] = scanner.nextInt();
        }

        MinSubArray solution = new MinSubArray();
        int result = solution.minSubArrayLen(target, nums);
        System.out.println("The minimal length of the subarray is: " + result);

        scanner.close();
    }
}
```

Time Complexity : O(n)

```
C:\Users\POOJA\Documents\SDE\DSA_Practice8>javac MinSubArray.java

C:\Users\POOJA\Documents\SDE\DSA_Practice8>java MinSubArray
Enter the target value: 7
Enter the length of the array: 6
Enter the elements of the array: 2 3 1 2 4 3
The minimal length of the subarray is: 2
```

7. Longest substring without repeating characters

```
import java.util.Scanner;

class LongestSubstring {
    public int lengthOfLongestSubstring(String s) {
        int n = s.length();
        int maxLength = 0;
```

```
        int left = 0;
        java.util.HashSet<Character> set = new java.util.HashSet<>();

        for (int right = 0; right < n; right++) {
            while (set.contains(s.charAt(right))) {
                set.remove(s.charAt(left));
                left++;
            }
            set.add(s.charAt(right));
            maxLength = Math.max(maxLength, right - left + 1);
        }

        return maxLength;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter a string: ");
        String s = scanner.nextLine();

        LongestSubstring solution = new LongestSubstring();
        int result = solution.lengthOfLongestSubstring(s);
        System.out.println("The length of the longest substring : " + result);

        scanner.close();
    }
}
```

Time Complexity : O(n)

```
C:\Users\POOJA\Documents\SDE\DSA_Practice8>javac LongestSubstring.java

C:\Users\POOJA\Documents\SDE\DSA_Practice8>java LongestSubstring
Enter a string: abcabcbb
The length of the longest substring : 3
```

8. Substring with concatenation of all words

```
import java.util.*;


class SubstringConcatenation {
    public List<Integer> findSubstring(String s, String[] words) {
        List<Integer> result = new ArrayList<>();
        int wordLength = words[0].length();
        int wordCount = words.length;
        int substringLength = wordLength * wordCount;

        Map<String, Integer> wordMap = new HashMap<>();
        for (String word : words) {
            wordMap.put(word, wordMap.getOrDefault(word, 0) + 1);
        }

        for (int i = 0; i <= s.length() - substringLength; i++) {
            Map<String, Integer> seenWords = new HashMap<>();
            int j = 0;
            while (j < wordCount) {
                String word = s.substring(i + j * wordLength, i + (j + 1) * wordLength);
                if (!wordMap.containsKey(word)) {
                    break;
```

```
                }
                seenWords.put(word, seenWords.getOrDefault(word, 0) + 1);
                if (seenWords.get(word) > wordMap.get(word)) {
                    break;
                }
                j++;
            }
            if (j == wordCount) {
                result.add(i);
            }
        }

        return result;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the string: ");
        String s = scanner.nextLine();

        System.out.print("Enter the number of words: ");
        int n = scanner.nextInt();
        scanner.nextLine();    // consume the remaining newline

        String[] words = new String[n];
        System.out.println("Enter the words:");
        for (int i = 0; i < n; i++) {
```

```
            words[i] = scanner.nextLine();
        }

        SubstringConcatenation solution = new SubstringConcatenation();
        List<Integer> result = solution.findSubstring(s, words);
        System.out.println("The starting indices of the concatenated substrings are: " + result);

        scanner.close();
    }
}
```

Time Complexity : O(n×m×k)

```
C:\Users\POOJA\Documents\SDE\DSA_Practice8>javac SubstringConcatenation.java

C:\Users\POOJA\Documents\SDE\DSA_Practice8>java SubstringConcatenation
Enter the string: barfoothefoobarman
Enter the number of words: 2
Enter the words:
foo
bar
The starting indices of the concatenated substrings are: [0, 9]
```

9. Minimum window substring

```
import java.util.*;

class MinWindowSubstr {
    public String minWindow(String s, String t) {
        Map<Character, Integer> targetMap = new HashMap<>();
        for (char c : t.toCharArray()) {
            targetMap.put(c, targetMap.getOrDefault(c, 0) + 1);
        }
```

```
        int left = 0, right = 0, count = 0, minLength = Integer.MAX_VALUE;
        int start = 0;
        Map<Character, Integer> windowMap = new HashMap<>();

        while (right < s.length()) {
            char c = s.charAt(right);
            windowMap.put(c, windowMap.getOrDefault(c, 0) + 1);

            if (targetMap.containsKey(c) && windowMap.get(c) <= targetMap.get(c)) {
                count++;
            }

            while (count == t.length()) {
                if (right - left + 1 < minLength) {
                    minLength = right - left + 1;
                    start = left;
                }

                char leftChar = s.charAt(left);
                windowMap.put(leftChar, windowMap.get(leftChar) - 1);
                if (targetMap.containsKey(leftChar) && windowMap.get(leftChar) <
targetMap.get(leftChar)) {
                    count--;
                }
                left++;
            }
            right++;
        }
```

```
        return minLength == Integer.MAX_VALUE ? "" : s.substring(start, start + minLength);
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the string s: ");
        String s = scanner.nextLine();

        System.out.print("Enter the string t: ");
        String t = scanner.nextLine();

        MinWindowSubstr solution = new MinWindowSubstr();
        String result = solution.minWindow(s, t);
        System.out.println("The minimum window substring is: " + result);

        scanner.close();
    }
}
```

Time Complexity : O(m+n)

```
C:\Users\POOJA\Documents\SDE\DSA_Practice8>java MinWindowSubstr
Enter the string s: ADOBECODEBANC
Enter the string t: ABC
The minimum window substring is: BANC
```

10. Valid Parenthesis

```
import java.util.Stack;

import java.util.Scanner;
```

```
class ValidParenthesis {
    public boolean isValid(String s) {
        Stack<Character> stack = new Stack<>();
        for (char c : s.toCharArray()) {
            if (c == '(' || c == '{' || c == '[') {
                stack.push(c);
            } else {
                if (stack.isEmpty()) return false;
                char top = stack.pop();
                if (c == ')' && top != '(') return false;
                if (c == '}' && top != '{') return false;
                if (c == ']' && top != '[') return false;
            }
        }
        return stack.isEmpty();
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the string s: ");
        String s = scanner.nextLine();

        ValidParenthesis solution = new ValidParenthesis();
        boolean result = solution.isValid(s);
        System.out.println("The string is valid: " + result);
```

```
        scanner.close();
    }
}
```

Time Complexity : O(n)

```
C:\Users\POOJA\Documents\SDE\DSA_Practice8>javac ValidParenthesis.java

C:\Users\POOJA\Documents\SDE\DSA_Practice8>java ValidParenthesis
Enter the string s: ()[]{}
The string is valid: true
```

11. Simplify Path

```
import java.util.Scanner;

import java.util.Stack;


class SimplifyPath{

    public String simplifyPath(String path) {

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

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


        for (String part : parts) {

            if (part.equals("..")) {

                if (!stack.isEmpty()) stack.pop();

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

                stack.push(part);

            }

        }


        StringBuilder result = new StringBuilder();

        for (String dir : stack) {
```

```
                result.append("/").append(dir);
            }

            return result.length() == 0 ? "/" : result.toString();
        }

        public static void main(String[] args) {
            Scanner scanner = new Scanner(System.in);

            System.out.print("Enter the absolute path: ");
            String path = scanner.nextLine();

            SimplifyPath solution = new SimplifyPath();
            String result = solution.simplifyPath(path);
            System.out.println("Simplified path: " + result);

            scanner.close();
        }
}
```

Time Complexity : O(n)

```
C:\Users\POOJA\Documents\SDE\DSA_Practice8>javac SimplifyPath.java

C:\Users\POOJA\Documents\SDE\DSA_Practice8>java SimplifyPath
Enter the absolute path: /home//foo/
Simplified path: /home/foo
```

12. Min Stack

```
import java.util.Scanner;

import java.util.Stack;
```

```
class MinStack {
    private Stack<Integer> stack;
    private Stack<Integer> minStack;

    public MinStack() {
        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.pop().equals(minStack.peek())) {
            minStack.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);
        MinStack minStack = new MinStack();
        while (true) {
            String operation = scanner.next();
            if (operation.equals("push")) {
                int val = scanner.nextInt();
                minStack.push(val);
            } else if (operation.equals("pop")) {
                minStack.pop();
            } else if (operation.equals("top")) {
                System.out.println(minStack.top());
            } else if (operation.equals("getMin")) {
                System.out.println(minStack.getMin());
            } else if (operation.equals("exit")) {
                break;
            }
        }
        scanner.close();
    }
}
```

Time Complexity : O(1)

```
C:\Users\POOJA\Documents\SDE\DSA_Practice8>java MinStack
push -2
push 0
push -3
getmin
top
-3
exit
```

13. Evaluate Reverse Polish Notation

```
import java.util.*;


class EvalRPN {
    public int evalRPN(String[] tokens) {
        Stack<Integer> stack = new Stack<>();
        for (String token : tokens) {
            if (token.equals("+") || token.equals("-") || token.equals("*") || token.equals("/")) {
                int b = stack.pop();
                int a = stack.pop();
                switch (token) {
                    case "+":
                        stack.push(a + b);
                        break;
                    case "-":
                        stack.push(a - b);
                        break;
                    case "*":
                        stack.push(a * b);
                        break;
                    case "/":
                        stack.push(a / b);
```

```
                    break;
                }
            } else {
                stack.push(Integer.parseInt(token));
            }
        }
        return stack.pop();
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        int n = Integer.parseInt(scanner.nextLine());
        String[] tokens = new String[n];
        for (int i = 0; i < n; i++) {
            tokens[i] = scanner.nextLine();
        }
        EvalRPN solution = new EvalRPN();
        System.out.println(solution.evalRPN(tokens));
    }
}
```

Time Complexity : O(n)

```
C:\Users\POOJA\Documents\SDE\DSA_Practice8>javac EvalRPN.j

C:\Users\POOJA\Documents\SDE\DSA_Practice8>java EvalRPN
5
2
1
+
3
*
9
```

14. Basic Calculator

```
import java.util.*;


public class Calculator {
    public int calculate(String s) {
        Stack<Integer> stack = new Stack<>();
        int result = 0, number = 0, sign = 1;

        for (char c : s.toCharArray()) {
            if (Character.isDigit(c)) {
                number = number * 10 + (c - '0');
            } else if (c == '+') {
                result += sign * number;
                number = 0;
                sign = 1;
            } else if (c == '-') {
                result += sign * number;
                number = 0;
                sign = -1;
            } else if (c == '(') {
                stack.push(result);
                stack.push(sign);
                result = 0;
                sign = 1;
            } else if (c == ')') {
                result += sign * number;
                number = 0;
                result *= stack.pop();
```

```
                result += stack.pop();
            }
        }
        return result + (sign * number);
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String s = sc.nextLine();
        Calculator sol = new Calculator();
        System.out.println(sol.calculate(s));
    }
}
```

Time Complexity : O(n)

```
C:\Users\POOJA\Documents\SDE\DSA_Practice8>javac Calculator.java

C:\Users\POOJA\Documents\SDE\DSA_Practice8>java Calculator
123 + 768
891
```

15. Search Insert Position

```
import java.util.Scanner;


class InsertPosition {
    public int searchInsert(int[] nums, int target) {
        int left = 0, right = nums.length;
        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;
            }
        }
        return left;
    }

    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();
        }
        System.out.println("Enter the target value:");
        int target = sc.nextInt();

        InsertPosition solution = new InsertPosition();
        int result = solution.searchInsert(nums, target);
        System.out.println("The target would be inserted at index: " + result);
    }
}
```

Time Complexity : O(log n)

```
C:\Users\POOJA\Documents\SDE\DSA_Practice8>javac InsertPosition.java

C:\Users\POOJA\Documents\SDE\DSA_Practice8>java InsertPosition
Enter the size of the array:
4
Enter the elements of the array:
1 3 5 6
Enter the target value:
2
The target would be inserted at index: 1
```

16. Search a 2D matrix

```
import java.util.Scanner;


class SearchMatrix{
    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);

        System.out.println("Enter the number of rows (m):");
        int m = scanner.nextInt();

        System.out.println("Enter the number of columns (n):");
        int n = scanner.nextInt();

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

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

        SearchMatrix solution = new SearchMatrix();
        boolean result = solution.searchMatrix(matrix, target);
```

```
            if (result) {
                System.out.println("true");
            } else {
                System.out.println("false");
            }
        }
}
```

Time Complexity : O(log(m * n))

```
C:\Users\POOJA\Documents\SDE\DSA_Practice8>java SearchMatrix
Enter the number of rows (m):
3
Enter the number of columns (n):
4
Enter the elements of the matrix row by row:
1 3 5 7
10 11 16 20
23 30 34 60
Enter the target value:
3
true
```

17. Find Peak Element

```
import java.util.Scanner;


class PeakElement {
    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);

        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 elements of the array:");
        for (int i = 0; i < n; i++) {
            nums[i] = scanner.nextInt();
        }

        PeakElement solution = new PeakElement();
        int peakIndex = solution.findPeakElement(nums);

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

Time Complexity : O(log n)

```
C:\Users\POOJA\Documents\SDE\DSA_Practice8>javac PeakElement.java

C:\Users\POOJA\Documents\SDE\DSA_Practice8>java PeakElement
Enter the number of elements in the array:
7
Enter the elements of the array:
1 2 1 3 5 6 4
Peak element index: 5
```

18. Search in Rotated Sorted Array

```
import java.util.Scanner;


class Search {
    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);

        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 elements of the array:");
        for (int i = 0; i < n; i++) {
            nums[i] = scanner.nextInt();
        }

        System.out.println("Enter the target value:");
        int target = scanner.nextInt();

        Search solution = new Search();
```

```
        int result = solution.search(nums, target);

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

Time Complexity : O(log n)

```
C:\Users\POOJA\Documents\SDE\DSA_Practice8>javac Search.java

C:\Users\POOJA\Documents\SDE\DSA_Practice8>java Search
Enter the number of elements in the array:
7
Enter the elements of the array:
4 5 6 7 0 1 2
Enter the target value:
4
Target index: 0
```

19. Find First and Last Position of an Element in a sorted array

```
import java.util.Scanner;

class FirstAndLast{
    public int[] searchRange(int[] nums, int target) {
        int[] result = {-1, -1};
        result[0] = findLeft(nums, target);
        result[1] = findRight(nums, target);
        return result;
    }

    private int findLeft(int[] nums, int target) {
        int left = 0, right = nums.length - 1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
```

```
            if (nums[mid] >= target) {
                right = mid - 1;
            } else {
                left = mid + 1;
            }
        }
        if (left < nums.length && nums[left] == target) {
            return left;
        }
        return -1;
    }

    private int findRight(int[] nums, int target) {
        int left = 0, right = nums.length - 1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (nums[mid] <= target) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }
        if (right >= 0 && nums[right] == target) {
            return right;
        }
        return -1;
    }
```

```
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        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 elements of the array:");
        for (int i = 0; i < n; i++) {
            nums[i] = scanner.nextInt();
        }

        System.out.println("Enter the target value:");
        int target = scanner.nextInt();

        FirstAndLast solution = new FirstAndLast();
        int[] result = solution.searchRange(nums, target);

        System.out.println("Result: [" + result[0] + ", " + result[1] + "]");
    }
}
```

Time Complexity : O(log n)

```
C:\Users\POOJA\Documents\SDE\DSA_Practice8>javac FirstAndLast.java

C:\Users\POOJA\Documents\SDE\DSA_Practice8>java FirstAndLast
Enter the number of elements in the array:
6
Enter the elements of the array:
5 7 7 8 8 10
Enter the target value:
8
Result: [3, 4]
```

20. Find Minimum in Rotated Sorted Array

```
import java.util.Scanner;


class MinElement{
    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);
```

```
        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 elements of the rotated array:");
        for (int i = 0; i < n; i++) {
            nums[i] = scanner.nextInt();
        }

        MinElement solution = new MinElement();
        int result = solution.findMin(nums);

        System.out.println("Minimum element: " + result);
    }
}
```

Time Complexity : O(log n)

```
C:\Users\POOJA\Documents\SDE\DSA_Practice8>javac MinElement.java

C:\Users\POOJA\Documents\SDE\DSA_Practice8>java MinElement
Enter the number of elements in the array:
5
Enter the elements of the rotated array:
3 4 5 1 2
Minimum element: 1
```