Problem Solving – 21/11/2024

1. Valid Palindrome
   Time complexity: O(n)
   Space complexity: O(1)

```java
package program21stNov;
import java.util.Scanner;
public class ValidPalindrome {
    public boolean isPalindrome(String s) {
        int l = 0;
        int r = s.length() - 1;
        while (l < r) {
            while (l < r && !Character.isLetterOrDigit(s.charAt(l))) l++;
            while (l < r && !Character.isLetterOrDigit(s.charAt(r))) r--;
            if (Character.toLowerCase(s.charAt(l)) != Character.toLowerCase(s.charAt(r))) return false;
            l++;
            r--;
        }
        return true;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter a string:");
        String input = scanner.nextLine();
        ValidPalindrome validator = new ValidPalindrome();
        boolean result = validator.isPalindrome(input);
        if (result) {
            System.out.println("The string is a palindrome.");
        } else {
            System.out.println("The string is not a palindrome.");
        }
        scanner.close();
    }
}

```

```
<terminated> ValidPalindrome [Java Application] C:\Program Files\J
Enter a string:
aslj;dfaskjdh
The string is not a palindrome.
```

2. Is Subsequence
   Time complexity: O(n)
   Space complexity: O(1)

```java
package program21stNov;
import java.util.Scanner;
public class IsSubsequence {
    public boolean isSubsequence(String s, String t) {
        int i = 0, j = 0;
        while (i < s.length() && j < t.length()) {
            if (s.charAt(i) == t.charAt(j)) {
                i++;
            }
            j++;
        }
        return i == s.length();
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the first string:");
        String s = scanner.nextLine();
        System.out.println("Enter the second string:");
        String t = scanner.nextLine();
        IsSubsequence checker = new IsSubsequence();
        boolean result = checker.isSubsequence(s, t);
        if (result) {
            System.out.println("Yes");
        } else {
            System.out.println("No");
        }
        scanner.close();
    }
}

```

```
<terminated> IsSubsequence [Java Application] C:\Program Files\
Enter the first string:
abcd
Enter the second string:
akjbhgciuuyhdmncx
Yes
```

3. Two Sum Sorted Array
   Time complexity: O(n)
   Space complexity: O(1)

```java
package program21stNov;
import java.util.Scanner;
public class TwoSumSortedArray {
    public int[] twoSum(int[] numbers, int target) {
        int left = 0;
        int right = numbers.length - 1;
        while (left < right) {
            int total = numbers[left] + numbers[right];
            if (total == target) {
                return new int[]{left + 1, right + 1};
            } else if (total > target) {
                right--;
            } else {
                left++;
            }
        }
        return new int[]{-1, -1};
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the size of the array:");
        int size = scanner.nextInt();
        int[] numbers = new int[size];
        System.out.println("Enter the elements of the sorted array:");
        for (int i = 0; i < size; i++) {
            numbers[i] = scanner.nextInt();
        }
        System.out.println("Enter the target value:");
        int target = scanner.nextInt();
        TwoSumSortedArray solver = new TwoSumSortedArray();
        int[] result = solver.twoSum(numbers, target);
        if (result[0] == -1) {
            System.out.println("No solution found.");
        } else {
            System.out.println(result[0] +" "+ result[1]);
        }
        scanner.close();
    }
}

```

```
<terminated> TwoSumSortedArray [Java Application] C:\Program Files\Java
Enter the size of the array:
10
Enter the elements of the sorted array:
1 2 3 4 5 6 7 8 9 11
Enter the target value:
9
1 8
```

4. Container With Most Water
   Time complexity: O(n)
   Space complexity: O(1)

```java
package program21stNov;
public class ContainerWithMostWater {
    public int maxArea(int[] height) {
        int area = 0, res = 0;
        int l = 0, r = height.length - 1;
        while (l < r) {
            area = (r - l) * Math.min(height[l], height[r]);
            res = Math.max(res, area);

            if (height[l] > height[r]) {
                r--;
            } else {
                l++;
            }
        }
        return res;
    }

    public static void main(String[] args) {
        java.util.Scanner sc = new java.util.Scanner(System.in);
        System.out.println("Enter the number of elements:");
        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();
        }
        ContainerWithMostWater solution = new ContainerWithMostWater();
        int result = solution.maxArea(height);
        System.out.println("The maximum area is: " + result);
    }
}
```

```
<terminated> ContainerWithMostWater [Java Application] C:\Progr
Enter the number of elements:
8
Enter the heights:
1 2 3 4 5 6 7 8
The maximum area is: 16
```

5. Three Sum
   Time complexity: $O(n^2)$
   Space complexity: O(n)

```java
package program21stNov;
import java.util.*;
public class ThreeSum {
    public List<List<Integer>> threeSum(int[] nums) {
        List<List<Integer>> res = new ArrayList<>();
        Arrays.sort(nums);
        for (int i = 0; i < nums.length; i++) {
            if (i > 0 && nums[i] == nums[i - 1]) {
                continue;
            }
            int j = i + 1;
            int k = nums.length - 1;
            while (j < k) {
                int total = nums[i] + nums[j] + nums[k];
                if (total > 0) {
                    k--;
                } else if (total < 0) {
                    j++;
                } else {
                    res.add(Arrays.asList(nums[i], nums[j], nums[k]));
                    j++;
                    while (j < k && nums[j] == nums[j - 1]) {
                        j++;
                    }}}}
        return res;
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.println("Enter the number of elements:");
        int n = sc.nextInt();
        int[] nums = new int[n];
        System.out.println("Enter the numbers:");
        for (int i = 0; i < n; i++) {
            nums[i] = sc.nextInt();
        }
        ThreeSum solution = new ThreeSum();
        List<List<Integer>> result = solution.threeSum(nums);
        System.out.println("The triplets are:");
        for (List<Integer> triplet : result) {
            System.out.println(triplet);
        }
        sc.close();
    }
}
```

```
<terminated> ThreeSum [Java Application] C:\Program Files\Ja
Enter the number of elements:
12
Enter the numbers:
-1 -2 -3 -4 -5 5 4 3 2 1 5 6
The triplets are:
[-5, -1, 6]
[-5, 1, 4]
[-5, 2, 3]
[-4, -2, 6]
[-4, -1, 5]
[-4, 1, 3]
[-3, -2, 5]
[-3, -1, 4]
[-3, 1, 2]
[-2, -1, 3]
```

6. Minimum Size Subarray Sum
   Time complexity: O(n)
   Space complexity: O(1)

```java
package program21stNov;
import java.util.Scanner;
public class MinimumSizeSubarraySum {
    public static int minSubArrayLen(int target, int[] nums) {
        int minLen = Integer.MAX_VALUE;
        int left = 0;
        int curSum = 0;
        for (int right = 0; right < nums.length; right++) {
            curSum += nums[right];
            while (curSum >= target) {
                if (right - left + 1 < minLen) {
                    minLen = right - left + 1;
                }
                curSum -= nums[left];
                left++;
            }
        }
        return minLen != Integer.MAX_VALUE ? minLen : 0;
    }
    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.println("Enter the elements of the array: ");
        for (int i = 0; i < n; i++) {
            nums[i] = scanner.nextInt();
        }
        int result = minSubArrayLen(target, nums);
        System.out.println(result);
    }
}
```

```
<terminated> MaximumSizeSubarraySum [Java Application]
Enter the target value: 7
Enter the length of the array: 6
Enter the elements of the array:
2 3 1 2 4 3
2
```

7. Longest Substring Without Repeating Characters
   Time complexity: O(n)
   Space complexity: O(min(n, m))

```java
package program21stNov;
import java.util.*;
public class LongestSubstringWithoutRepeatingCharacters {
    public static int lengthOfLongestSubstring(String s) {
        Set<Character> st = new HashSet<>();
        int left = 0, result = 0;
        for (int right = 0; right < s.length(); right++) {
            while (st.contains(s.charAt(right))) {
                st.remove(s.charAt(left));
                left++;
            }
            st.add(s.charAt(right));
            result = Math.max(result, right - left + 1);
        }
        return result;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the string: ");
        String s = scanner.nextLine();
        int result = lengthOfLongestSubstring(s);
        System.out.println("The length is:" + result);
    }
}

```

```
<terminated> LongestSubstringWithoutRepeatingCharacters [Java
Enter the string: bavcsdbavcsdabvc
The length is:6
```

8. Substring With Concatenation of All Words
   Time complexity: O(n * m * w)
   Space complexity: O(n + m)

```java
package program21stNov;
import java.util.*;
public class SubstringwithConcatenationofAllWords {
    public static List<Integer> findSubstring(String s, String[] words) {
        List<Integer> ans = new ArrayList<>();
        int n = s.length();
        int m = words.length;
        int w = words[0].length();
        HashMap<String, Integer> map = new HashMap<>();
        for (String x : words)
            map.put(x, map.getOrDefault(x, 0) + 1);
        for (int i = 0; i < w; i++) {
            HashMap<String, Integer> temp = new HashMap<>();
            int count = 0;
            for (int j = i, k = i; j + w <= n; j = j + w) {
                String word = s.substring(j, j + w);
                temp.put(word, temp.getOrDefault(word, 0) + 1);
                count++;

                if (count == m) {
                    if (map.equals(temp)) {
                        ans.add(k);
                    }
                    String remove = s.substring(k, k + w);
                    temp.computeIfPresent(remove, (a, b) -> (b > 1) ? b - 1 : null);
                    count--;
                    k = k + w;
                }}}
        return ans;}
    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();
        String[] words = new String[n];
        System.out.println("Enter the words: ");
        for (int i = 0; i < n; i++) {
            words[i] = scanner.nextLine();
        }
        List<Integer> result = findSubstring(s, words);
        System.out.println("The starting indices: " + result);
    }
```

```
<terminated> SubstringwithConcatenationofAllWords [Java Appl
Enter the string: barfoothefoobarman
Enter the number of words: 2
Enter the words:
foo
bar
The starting indices: [0, 9]
```

9. Minimum Window Substring
   Time complexity: O(n)
   Space complexity: O(m)

```java
package program21stNov;
import java.util.*;
public class MinimumWindowSubstring {
    public static String minWindow(String s, String t) {
        if (s.length() < t.length()) {
            return "";        }
        Map<Character, Integer> needstr = new HashMap<>();
        for (char ch : t.toCharArray()) {
            needstr.put(ch, needstr.getOrDefault(ch, 0) + 1);        }
        int needcnt = t.length();
        int start = 0;
        int[] res = new int[]{0, Integer.MAX_VALUE};
        for (int end = 0; end < s.length(); end++) {
            char ch = s.charAt(end);
            if (needstr.containsKey(ch)) {
                if (needstr.get(ch) > 0) {
                    needcnt--;
                }
                needstr.put(ch, needstr.get(ch) - 1);
            }
            while (needcnt == 0) {
                char temp = s.charAt(start);
                if (needstr.containsKey(temp) && needstr.get(temp) == 0) {
                    break;
                }
                if (needstr.containsKey(temp)) {
                    needstr.put(temp, needstr.get(temp) + 1);
                }
                start++;
            }
            if (needcnt == 0 && end - start < res[1] - res[0]) {
                res[0] = start;
                res[1] = end;}}
        return res[1] == Integer.MAX_VALUE ? "" : s.substring(res[0], res[1] + 1);
    }
    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();
        String result = minWindow(s, t);
        System.out.println("Minimum window substring: " + result);
    }
```

```
<terminated> MinimumWindowSubstring [Java Application] C:
Enter the string s: ADOBECODEBANC
Enter the string t: ABC
Minimum window substring: BANC
```

10. Valid Parentheses
    Time complexity: O(n)
    Space complexity: O(n)

```java
package program21stNov;
import java.util.*;
public class ValidParantheses {
    public static boolean isValid(String s) {
        Stack<Character> stack = new Stack<Character>();
        for (char c : s.toCharArray()) {
            if (c == '(' || c == '[' || c == '{') {
                stack.push(c);
            } else {
                if (stack.isEmpty()) {
                    return false;
                }
                char top = stack.peek();
                if ((c == ')' && top == '(') || (c == ']' && top == '[') || (c == '}' && top == '{')) {
                    stack.pop();
                } else {
                    return false;
                }
            }
        }
        return stack.isEmpty();
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a string of parentheses: ");
        String s = scanner.nextLine();
        boolean result = isValid(s);
        System.out.println(result);
    }
}

```

```
<terminated> ValidParantheses [Java Application] C:\Program Files\J
Enter a string of parentheses: ({[]})
true
```

11. Simplify Path
    Time complexity: O(n)
    Space complexity: O(n)

```java
package program21stNov;
import java.util.*;
public class SimplifyPath {
    public static String simplifyPath(String path) {
        Stack<String> stack = new Stack<>();
        String[] directories = path.split("/");
        for (String dir : directories) {
            if (dir.equals(".") || dir.isEmpty()) {
                continue;
            } else if (dir.equals("..")) {
                if (!stack.isEmpty()) {
                    stack.pop();
                }
            } else {
                stack.push(dir);
            }
        }
        return "/" + String.join("/", stack);
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the path: ");
        String path = scanner.nextLine();
        String simplifiedPath = simplifyPath(path);
        System.out.println("Simplified path: " + simplifiedPath);
    }
}

```

```
<terminated> SimplifyPath [Java Application] C:\Program Files\Java\bin\javaw.exe (21-N
Enter the path: /home/user/Documents/../Pictures
Simplified path: /home/user/Pictures
```

12. MinStack

Time complexity: O(n)

Space complexity: O(n)

```java
package program21stNov;
import java.util.*;
public class MinStack {
    private List<int[]> st;
    public MinStack() {
        st = new ArrayList<>();
    }
    public void push(int val) {
        int[] top = st.isEmpty() ? new int[]{val, val} : st.get(st.size() - 1);
        int min_val = top[1];
        if (min_val > val) {
            min_val = val;
        }
        st.add(new int[]{val, min_val});
    }
    public void pop() {
        st.remove(st.size() - 1);
    }
    public int top() {
        return st.isEmpty() ? -1 : st.get(st.size() - 1)[0];
    }
    public int getMin() {
        return st.isEmpty() ? -1 : st.get(st.size() - 1)[1];
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        MinStack stack = new MinStack();
        while (true) {
            System.out.println("Choose an option:");
            System.out.println("1. Push");
            System.out.println("2. Pop");
            System.out.println("3. Get Top");
            System.out.println("4. Get Min");
            System.out.println("5. Exit");
            int choice = scanner.nextInt();
            switch (choice) {
                case 1:
                    System.out.print("Enter value to push: ");
                    int val = scanner.nextInt();
                    stack.push(val);
                    break;
                case 2:
                    stack.pop();
                    break;
```

```
                    System.out.print("Enter value to push: ");
                    int val = scanner.nextInt();
                    stack.push(val);
                    break;
                case 2:
                    stack.pop();
                    break;
                case 3:
                    System.out.println("Top: " + stack.top());
                    break;
                case 4:
                    System.out.println("Min: " + stack.getMin());
                    break;
                case 5:
                    scanner.close();
                    return;
                default:
                    System.out.println("Invalid choice.");
            }
        }
    }
}

```

```
<terminated> MinStack [Java Application] C:\Program Files\Java
Choose an option:
1. Push
2. Pop
3. Get Top
4. Get Min
5. Exit
1
Enter value to push: 5
Choose an option:
1. Push
2. Pop
3. Get Top
4. Get Min
5. Exit
1
Enter value to push: 5
Choose an option:
1. Push
2. Pop
3. Get Top
4. Get Min
5. Exit
2
Choose an option:
1. Push
2. Pop
3. Get Top
4. Get Min
5. Exit
4
Min: 5
Choose an option:
1. Push
2. Pop
3. Get Top
4. Get Min
5. Exit
5
```

13. Evaluate Reversed Polish Notation

Time complexity: O(n)

Space complexity: O(n)

```java
package program21stNov;
import java.util.*;
public class EvaluateReversePolishNotation {
    public int evalRPN(String[] tokens) {
        Stack<Integer> stack = new Stack<>();
        for (String c : tokens) {
            if (c.equals("+")) {
                stack.push(stack.pop() + stack.pop());
            } else if (c.equals("-")) {
                int second = stack.pop();
                int first = stack.pop();
                stack.push(first - second);
            } else if (c.equals("*")) {
                stack.push(stack.pop() * stack.pop());
            } else if (c.equals("/")) {
                int second = stack.pop();
                int first = stack.pop();
                stack.push(first / second);
            } else {
                stack.push(Integer.parseInt(c));
            }
        }
        return stack.peek();
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        EvaluateReversePolishNotation obj = new EvaluateReversePolishNotation();
        System.out.println("Enter the number of tokens:");
        int n = scanner.nextInt();
        scanner.nextLine();
        String[] tokens = new String[n];
        System.out.println("Enter the tokens:");
        for (int i = 0; i < n; i++) {
            tokens[i] = scanner.nextLine();
        }
        int result = obj.evalRPN(tokens);
        System.out.println("Result of the expression: " + result);
        scanner.close();
    }
}
```

```
<terminated> EvaluateReversePolishNotation [Java Applica
Enter the number of tokens:
5
Enter the tokens:
2
1
+
3
*
Result of the expression: 9
```

14. Basic Calculator

Time complexity: O(n)

Space complexity: O(n)

```java
package program21stNov;
import java.util.*;
public class BasicCalculator {
    public int calculate(String s) {
        int number = 0;
        int signValue = 1;
        int result = 0;
        Stack<Integer> operationsStack = new Stack<>();
        for (int i = 0; i < s.length(); i++) {
            char c = s.charAt(i);
            if (Character.isDigit(c)) {
                number = number * 10 + (c - '0');
            } else if (c == '+' || c == '-') {
                result += number * signValue;
                signValue = (c == '-') ? -1 : 1;
                number = 0;
            } else if (c == '(') {
                operationsStack.push(result);
                operationsStack.push(signValue);
                result = 0;
                signValue = 1;
            } else if (c == ')') {
                result += signValue * number;
                result *= operationsStack.pop();
                result += operationsStack.pop();
                number = 0;
            }
        }
        return result + number * signValue;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        BasicCalculator calculator = new BasicCalculator();
        System.out.println("Enter the mathematical expression:");
        String input = scanner.nextLine();
        int result = calculator.calculate(input);
        System.out.println("The result: " + result);
        scanner.close();
    }
}

```

```
<terminated> BasicCalculator [Java Application] C:\Program File
Enter the mathematical expression:
1+2+3+(5+11)
The result: 22
```

15. Search Insert Position

Time complexity: O(log n)

Space complexity: O(1)

```java
package program21stNov;
import java.util.Scanner;
public class SearchInsertPosition {
    public static int searchInsert(int[] nums, int target) {
        int start = 0;
        int end = nums.length - 1;
        while (start <= end) {
            int mid = (start + end) / 2;
            if (nums[mid] == target) {
                return mid;
            } else if (nums[mid] > target) {
                end = mid - 1;
            } else {
                start = mid + 1;
            }
        }
        return end + 1;
    }
    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[] nums = new int[n];
        System.out.println("Enter sorted integers:");
        for (int i = 0; i < n; i++) {
            nums[i] = scanner.nextInt();
        }
        System.out.println("Enter the target value:");
        int target = scanner.nextInt();
        int result = searchInsert(nums, target);
        System.out.println("The target should be at " + result);
        scanner.close();
    }
}

```

```
<terminated> SearchInsertPosition [Java Application] C:\Program Fi
Enter the size of the array:
10
Enter sorted integers:
2 4 6 8 10 12 14 16 18 20
Enter the target value:
11
The target should be at 5
```

16. Search 2D Matrix

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

Space complexity: O(1)

```java
package program21stNov;
import java.util.Scanner;
public class Search2DMatrix {
    public boolean searchMatrix(int[][] matrix, int target) {
        int rows = matrix.length;
        int cols = matrix[0].length;
        int low = 0, high = (rows * cols) - 1;

        while (low <= high) {
            int mid = (low + high) / 2;
            int row = mid / cols;
            int col = mid % cols;

            if (matrix[row][col] == target) {
                return true;
            } else if (matrix[row][col] < target) {
                low = mid + 1;
            } else {
                high = mid - 1;
            }
        }
        return false;
    }
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the number of rows:");
        int rows = scanner.nextInt();
        System.out.println("Enter the number of columns:");
        int cols = scanner.nextInt();
        int[][] matrix = new int[rows][cols];
        System.out.println("Enter the elements of the matrix row by row:");
        for (int i = 0; i < rows; i++) {
            for (int j = 0; j < cols; j++) {
                matrix[i][j] = scanner.nextInt();
            }        }
        System.out.println("Enter the target value:");
        int target = scanner.nextInt();
        Search2DMatrix obj = new Search2DMatrix();
        boolean result = obj.searchMatrix(matrix, target);
        if (result) {
            System.out.println("Target is found ");
        } else {
            System.out.println("Target is not found ");
        }
```

```
<terminated> Search2DMatrix [Java Application] C:\Program Files\Java\bin\jav
Enter the number of rows:
4
Enter the number of columns:
4
Enter the elements of the matrix row by row:
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
Enter the target value:
10
Target is found
```

17. Find Peak Element

Time complexity: O(log n)

Space complexity: O(1)

```java
package program21stNov;
import java.util.Scanner;
public class FindPeakElement {
    public int findPeakElement(int[] nums) {
        int left = 0;
        int right = nums.length - 1;

        while (left < right) {
            int mid = (left + right) / 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();
        }
        FindPeakElement obj = new FindPeakElement();
        int peakIndex = obj.findPeakElement(nums);
        System.out.println("The peak element is at index: " + peakIndex);
        scanner.close();
    }
}

```

```
<terminated> FindPeakElement [Java Application] C:\Program Files\Java\b
Enter the number of elements in the array:
5
Enter the elements of the array:
1 2 3 1 5
The peak element is at index: 2
```

18. Search in Rotated Sorted Array

Time complexity: O(log n)

Space complexity: O(1)

```java
package program21stNov;
import java.util.Scanner;
public class SearchinRotatedSortedArray {
    public int search(int[] nums, int target) {
        int l = 0, r = nums.length - 1;
        while (l <= r) {
            int mid = (l + r) / 2;
            if (nums[mid] == target) {
                return mid;
            }
            if (nums[l] <= nums[mid]) {
                if (nums[l] <= target && target <= nums[mid]) {
                    r = mid - 1;
                } else {
                    l = mid + 1;
                }
            } else {
                if (nums[mid] <= target && target <= nums[r]) {
                    l = mid + 1;
                } else {
                    r = 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:");
        for (int i = 0; i < n; i++) {
            nums[i] = scanner.nextInt();}
        System.out.println("Enter the target element to search:");
        int target = scanner.nextInt();
        SearchinRotatedSortedArray obj = new SearchinRotatedSortedArray();
        int result = obj.search(nums, target);
        if (result != -1) {
            System.out.println(result);
        } else {
            System.out.println(-1);
        }

```

```
<terminated> SearchinRotatedSortedArray [Java Application] C:\Program Files\Java
Enter the number of elements in the array:
8
Enter the elements:
5 6 7 8 1 2 3 4
Enter the target element to search:
7
2
```

19. First and Last Position of Elements

Time complexity: O(log n)

Space complexity: O(1)

```java
package program21stNov;
import java.util.Scanner;
public class FirstAndLastPositionOfAnElement {
    public int[] searchRange(int[] nums, int target) {
        int[] result = {-1, -1};
        int left = binarySearch(nums, target, true);
        int right = binarySearch(nums, target, false);
        result[0] = left;
        result[1] = right;
        return result;
    }
    private int binarySearch(int[] nums, int target, boolean isSearchingLeft) {
        int left = 0;
        int right = nums.length - 1;
        int idx = -1;
        while (left <= right) {
            int mid = left + (right - left) / 2;
            if (nums[mid] > target) {
                right = mid - 1;
            } else if (nums[mid] < target) {
                left = mid + 1;
            } else {
                idx = mid;
                if (isSearchingLeft) {
                    right = mid - 1;
                } else {
                    left = mid + 1;
                }}}
        return idx;
    }
    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 sorted array:");
        for (int i = 0; i < n; i++) {
            nums[i] = scanner.nextInt();}
        System.out.println("Enter the target element:");
        int target = scanner.nextInt();
        FirstAndLastPositionOfAnElement obj = new FirstAndLastPositionOfAnElement();
        int[] result = obj.searchRange(nums, target);
        System.out.println("First position: " + result[0]);
        System.out.println("Last position: " + result[1]);
```

```
<terminated> FirstAndLastPositionOfAnElement [Java Application] C:\Program
Enter the number of elements in the array:
10
Enter the elements of the sorted array:
1 1 2 2 3 3 4 5 5 6
Enter the target element:
3
First position: 4
Last position: 5
```

20. Minimum Element in Rotated Sorted Array

Time complexity: O(log n)

Space complexity: O(1)

```java
package program21stNov;
import java.util.Scanner;
public class MinimumElementInRotatedSortedArray {
    public int findMin(int[] nums) {
        int left = 0;
        int right = nums.length - 1;
        while (left < right) {
            int mid = (left + right) / 2;
            if (nums[mid] <= nums[right]) {
                right = mid;
            } else {
                left = mid + 1;
            }
        }
        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 sorted array:");
        for (int i = 0; i < n; i++) {
            nums[i] = scanner.nextInt();
        }
        MinimumElementInRotatedSortedArray obj = new MinimumElementInRotatedSortedArray();
        int minElement = obj.findMin(nums);
        System.out.println("The minimum element " + minElement);
        scanner.close();
    }
}

```

```
<terminated> MinimumElementInRotatedSortedArray [Java Application] C:\Program Files\J
Enter the number of elements in the array:
10
Enter the elements of the rotated sorted array:
6 7 8 9 10 1 2 3 4 5
The minimum element 1
```