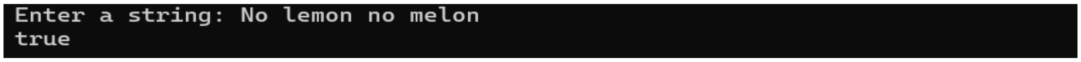
PRACTICE SET – 8 21-11-2024

DSA

1.Valid Palindrome

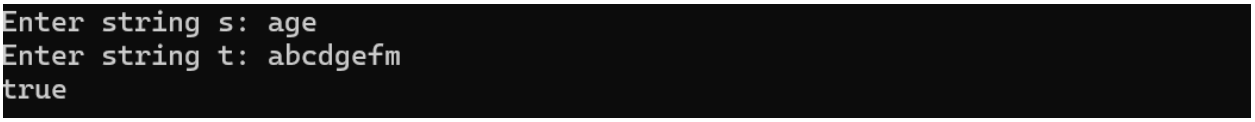
```
import java.util.Scanner;

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 filteredString = filtered.toString();
        String reversedString = filtered.reverse().toString();

        return filteredString.equals(reversedString);
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String s = scanner.nextLine();
        System.out.println(isPalindrome(s));
    }
}
```



```
Enter a string: No lemon no melon
true
```

2.Is Subsequence

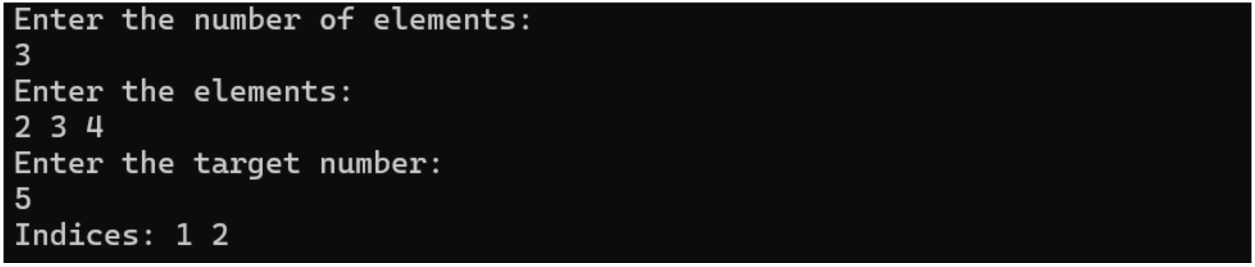
```
import java.util.Scanner;


public class Problem2 {
    public static boolean isSubsequence(String s, String t) {
        int sPointer = 0, tPointer = 0;


        while (sPointer < s.length() && tPointer < t.length()) {
            if (s.charAt(sPointer) == t.charAt(tPointer)) {
                sPointer++;
            }
            tPointer++;
        }


        return sPointer == s.length();
    }


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


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


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


        System.out.println(isSubsequence(s, t));
    }
}
```



```
Enter string s: age
Enter string t: abcdgefm
true
```

3.Two Sum II – Input array is sorted

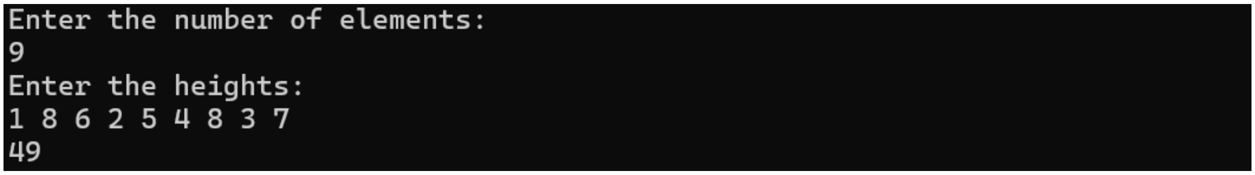
```
import java.util.Scanner;

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[]{-1, -1};
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the number of elements:");
        int n = scanner.nextInt();
        int[] numbers = new int[n];
        System.out.println("Enter the elements:");
        for (int i = 0; i < n; i++) {
            numbers[i] = scanner.nextInt();
        }
        System.out.println("Enter the target number:");
        int target = scanner.nextInt();
        int[] result = twoSum(numbers, target);
        System.out.println("Indices: " + result[0] + " " + result[1]);
    }
}
```



```
Enter the number of elements:
3
Enter the elements:
2 3 4
Enter the target number:
5
Indices: 1 2
```

4.Container with water

```
import java.util.Scanner;

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 currentHeight = Math.min(height[left], height[right]);
            int currentArea = width * currentHeight;
            maxArea = Math.max(maxArea, currentArea);

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

        return maxArea;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the number of elements:");
        int n = scanner.nextInt();
        int[] height = new int[n];
        System.out.println("Enter the heights:");
        for (int i = 0; i < n; i++) {
            height[i] = scanner.nextInt();
        }
        int result = maxArea(height);
        System.out.println(result);
    }
}
```

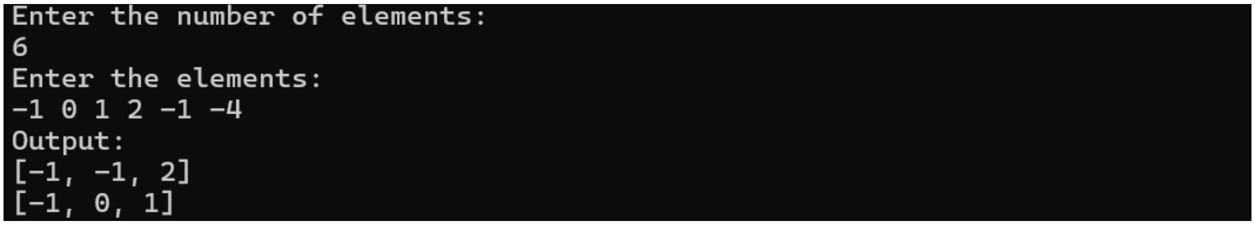


```
Enter the number of elements:
9
Enter the heights:
1 8 6 2 5 4 8 3 7
49
```

5.3sum

```
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Scanner;

public class Problem5 {
```

```
public static List<List<Integer>> threeSum(int[] nums) {
    List<List<Integer>> result = new ArrayList<>();
    Arrays.sort(nums);

    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 scanner = new Scanner(System.in);
        System.out.println("Enter the number of elements:");
        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();
        }
        List<List<Integer>> result = threeSum(nums);
        if (result.isEmpty()) {
            System.out.println("No triplets found.");
        } else {
            System.out.println("Output:");
            for (List<Integer> triplet : result) {
                System.out.println(triplet);
            }
        }
    }
}
```

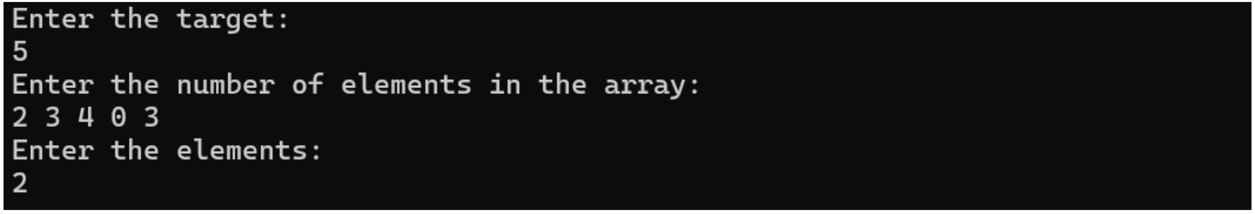


```
Enter the number of elements:
6
Enter the elements:
-1 0 1 2 -1 -4
Output:
[-1, -1, 2]
[-1, 0, 1]
```

6.Minimum Size Subarray Sum

```
import java.util.Scanner;
```

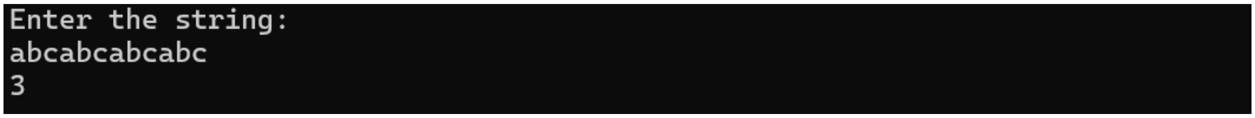
```
public class Problem6 {
    public static int minSubArrayLen(int target, int[] nums) {
        int n = nums.length;
        int left = 0, sum = 0, minLen = Integer.MAX_VALUE;

        for (int right = 0; right < n; right++) {
            sum += nums[right];

            while (sum >= target) {
                minLen = Math.min(minLen, right - left + 1);
                sum -= nums[left];
                left++;
            }
        }

        return minLen == Integer.MAX_VALUE ? 0 : minLen;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Enter the target:");
        int target = scanner.nextInt();
        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();
        }
        int result = minSubArrayLen(target, nums);
        System.out.println(result);
    }
}
```



```
Enter the target:
5
Enter the number of elements in the array:
2 3 4 0 3
Enter the elements:
2
```

7.Longest Substring Without repeating characters

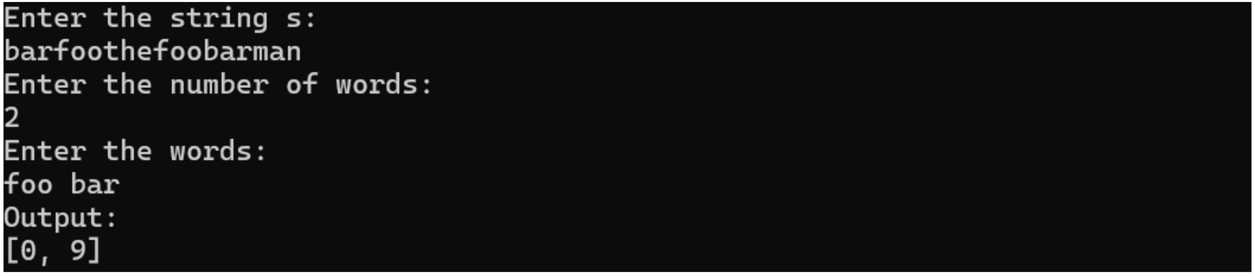
```
import java.util.HashSet;
import java.util.Scanner;


public class Problem7 {
    public static int lengthOfLongestSubstring(String s) {
        int n = s.length();
        HashSet<Character> set = new HashSet<>();
        int left = 0, maxLength = 0;

        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.println("Enter the string:");
        String s = scanner.nextLine();
        int result = lengthOfLongestSubstring(s);
        System.out.println(result);
    }
}
```



```
Enter the string:
abcabcabcabc
3
```

8.Substring with concatenation of all words

```
import java.util.*;

public class Problem8 {
    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();

        System.out.println("Enter the words:");
        String[] words = new String[n];
        for (int i = 0; i < n; i++) {
            words[i] = sc.next();
        }
        List<Integer> result = findSubstring(s, words);

        System.out.println("Output:");
        if (result.isEmpty()) {
            System.out.println("[]");
        } else {
            System.out.println(result);
        }
    }

    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 wordCount = words.length;
        int substringLength = wordLength * wordCount;

        Map<String, Integer> wordFrequencyMap = new HashMap<>();
        for (String word : words) {
            wordFrequencyMap.put(word, wordFrequencyMap.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) {
                int wordStart = i + j * wordLength;
                String word = s.substring(wordStart, wordStart + wordLength);

                if (!wordFrequencyMap.containsKey(word)) {
                    break;
                }

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

                if (seenWords.get(word) > wordFrequencyMap.get(word)) {
                    break;
                }

                j++;
            }

            if (j == wordCount) {
                result.add(i);
            }
        }

        return result;
    }
}
```



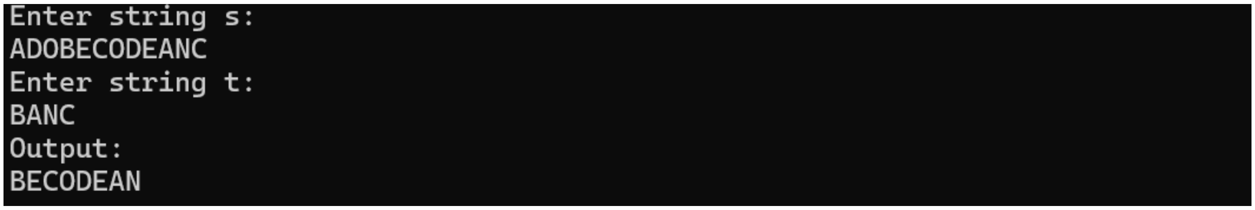
```
Enter the string s:
barfoothefoobarman
Enter the number of words:
2
Enter the words:
foo bar
Output:
[0, 9]
```

9.Minimum Window Substring

```
import java.util.*;

public class Problem9 {
    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();

        String result = minWindow(s, t);

        System.out.println("Output:");
        System.out.println(result);
    }

    public static String minWindow(String s, String t) {
        if (s == null || t == null || s.length() < t.length()) {
            return "";
        }

        Map<Character, Integer> tMap = new HashMap<>();
        for (char c : t.toCharArray()) {
            tMap.put(c, tMap.getOrDefault(c, 0) + 1);
        }

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

```
int left = 0, right = 0;
int required = tMap.size();
int formed = 0;

int[] ans = {-1, 0, 0}; // length, left, right

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

  if (tMap.containsKey(c) && windowMap.get(c).intValue() == tMap.get(c).intValue()) {
    formed++;
  }

  while (left <= right && formed == required) {
    c = s.charAt(left);

    if (ans[0] == -1 || right - left + 1 < ans[0]) {
      ans[0] = right - left + 1;
      ans[1] = left;
      ans[2] = right;
    }

    windowMap.put(c, windowMap.get(c) - 1);
    if (tMap.containsKey(c) && windowMap.get(c).intValue() < tMap.get(c).intValue()) {
      formed--;
    }

    left++;
  }
```

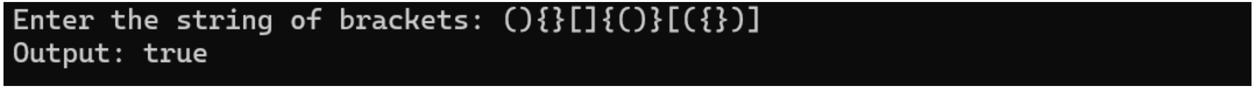
```
            right++;
        }

        return ans[0] == -1 ? "" : s.substring(ans[1], ans[2] + 1);
    }
}
```



```
Enter string s:
ADOBECODEANC
Enter string t:
BANC
Output:
BECODEAN
```

10.Valid Parantheses

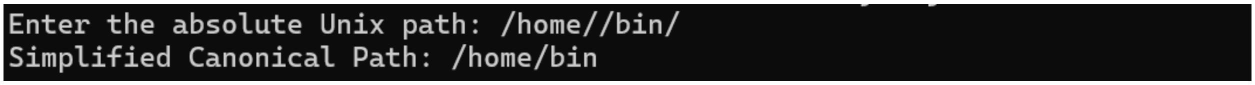
```
import java.util.Stack;
import java.util.Scanner;

public class Problem10 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

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

        if (isValid(s)) {
            System.out.println("Output: true");
        } else {
            System.out.println("Output: false");
        }
    }

    public static boolean isValid(String s) {
        Stack<Character> stack = new Stack<>();

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

            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();
    }
}
```



```
Enter the string of brackets: (){}[]{()}[({})]
Output: true
```

11.Simplify path

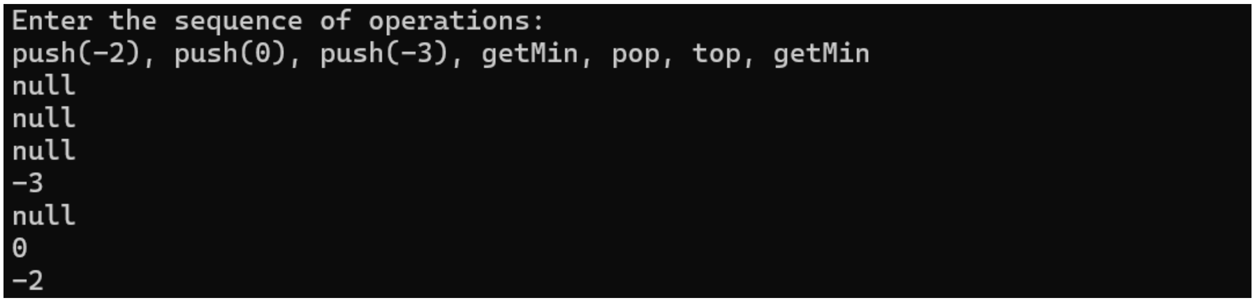
```
import java.util.Stack;
import java.util.Scanner;

public class Problem11 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter the absolute Unix path: ");
        String path = scanner.nextLine();

        System.out.println("Simplified Canonical Path: " + simplifyPath(path));
    }

    public static String simplifyPath(String path) {
        Stack<String> stack = new Stack<>();

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

        for (String component : components) {
            if (component.equals("..")) {
                if (!stack.isEmpty()) {
                    stack.pop();
                }
            } else if (component.equals(".") || component.isEmpty()) {
                continue;
            } else {
                stack.push(component);
            }
        }

        StringBuilder simplifiedPath = new StringBuilder();

        if (stack.isEmpty()) {
            simplifiedPath.append("/");
        } else {
            while (!stack.isEmpty()) {
                simplifiedPath.insert(0, "/" + stack.pop());
            }
        }

        return simplifiedPath.toString();
    }
}
```



```
Enter the absolute Unix path: /home//bin/
Simplified Canonical Path: /home/bin
```

12.Min stack

```
import java.util.Stack;
import java.util.Scanner;

public class Problem12 {
    static 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);
            } else {
                minStack.push(minStack.peek());
            }
    }

    public void pop() {
        stack.pop();
        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();

    System.out.println("Enter the sequence of operations:");
    String[] operations = scanner.nextLine().split(",");

    for (String operation : operations) {
        operation = operation.trim();
        if (operation.startsWith("push")) {
            int value = Integer.parseInt(operation.substring(5, operation.length() - 1).trim());
            minStack.push(value);
            System.out.println("null");
            } else if (operation.equals("pop")) {
                minStack.pop();
                System.out.println("null");
            } else if (operation.equals("top")) {
                System.out.println(minStack.top());
            } else if (operation.equals("getMin")) {
                System.out.println(minStack.getMin());
            }
        }

        scanner.close();
    }
}
```

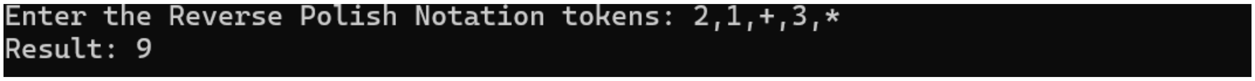


```
Enter the sequence of operations:
push(-2), push(0), push(-3), getMin, pop, top, getMin
null
null
null
-3
null
0
-2
```

13.Evaluate Reverse Polish Notation

```
import java.util.Stack;
import java.util.Scanner;

public class Problem13 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the Reverse Polish Notation tokens: ");
        String input = scanner.nextLine();
    String[] tokens = input.split(",");

    System.out.println("Result: " + evalRPN(tokens));
}

public static 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);
        } else {
            stack.push(Integer.parseInt(token));
        }
    }
```

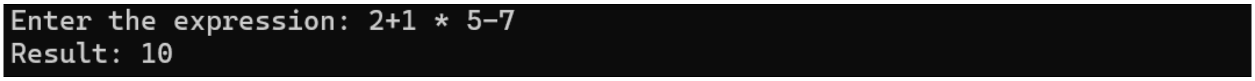
```
        return stack.pop();
    }
}
```



```
Enter the Reverse Polish Notation tokens: 2,1,+,3,*
Result: 9
```

14.Basic Calculator

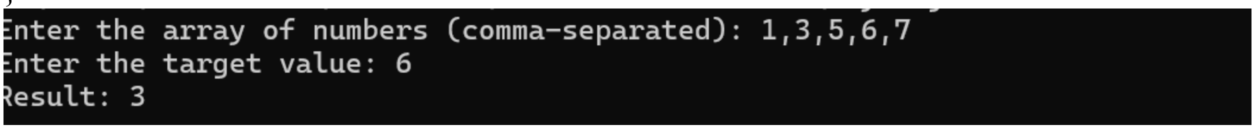
```
import java.util.Stack;
import java.util.Scanner;


public class Problem14 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);


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


        System.out.println("Result: " + calculate(s));
    }


    public static int calculate(String s) {
        Stack<Integer> stack = new Stack<>();
        int result = 0;
        int sign = 1;
        int num = 0;


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


            if (Character.isDigit(ch)) {
                num = num * 10 + (ch - '0');
            }

            if (ch == '+' || ch == '-' || i == s.length() - 1 || ch == '(' || ch == ')') {
                if (ch == '(') {
                    stack.push(result);
                    stack.push(sign);
                    result = 0;
                    sign = 1;
                } else if (ch == ')') {
                    result += sign * num;
                    num = 0;
                    result *= stack.pop();
                    result += stack.pop();
                } else {
                    result += sign * num;
                    num = 0;
                    sign = (ch == '-') ? -1 : 1;
                }
            }
        }

        if (num != 0) {
            result += sign * num;
        }

        return result;
    }
}
```



```
Enter the expression: 2+1 * 5-7
Result: 10
```

15.Search Insert Position

```
import java.util.Scanner;


public class Problem15 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);


        System.out.print("Enter the array of numbers (comma-separated): ");
        String input = scanner.nextLine();
        String[] inputArr = input.split(",");
        int[] nums = new int[inputArr.length];


        for (int i = 0; i < inputArr.length; i++) {
            nums[i] = Integer.parseInt(inputArr[i].trim());
        }


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


        System.out.println("Result: " + searchInsert(nums, target));
    }


    public static 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;
    }
}
```



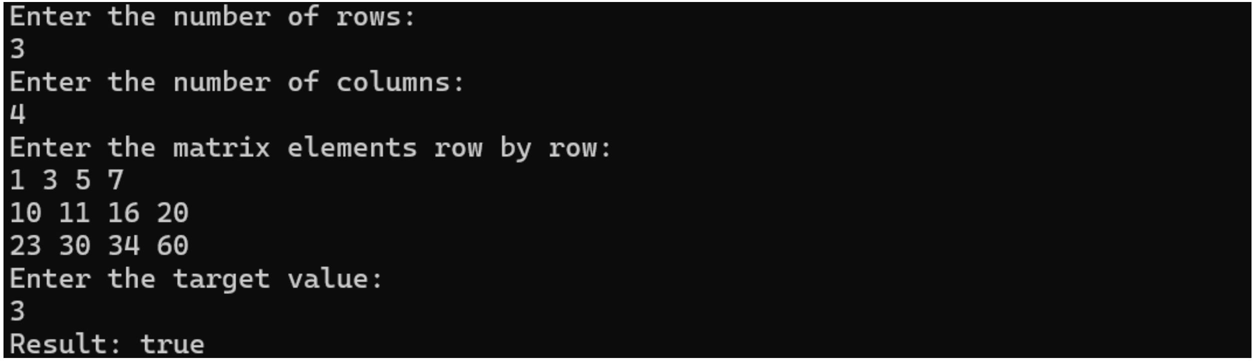
```
Enter the array of numbers (comma-separated): 1,3,5,6,7
Enter the target value: 6
Result: 3
```

16.Search 2D Matrix

```
import java.util.Scanner;


public class Problem16{
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);


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


        int[][] matrix = new int[m][n];
        System.out.println("Enter the matrix elements 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();

    System.out.println("Result: " + searchMatrix(matrix, target));
}

public static 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;
        }
    }
```

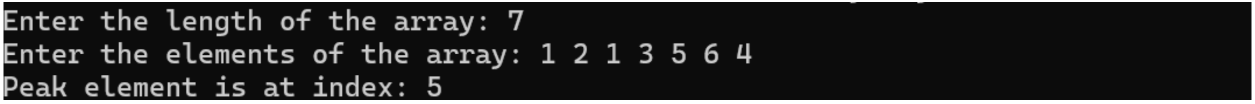
```
        return false;
    }
}
```



```
Enter the number of rows:
3
Enter the number of columns:
4
Enter the matrix elements row by row:
1 3 5 7
10 11 16 20
23 30 34 60
Enter the target value:
3
Result: true
```

17.Find Peak Element

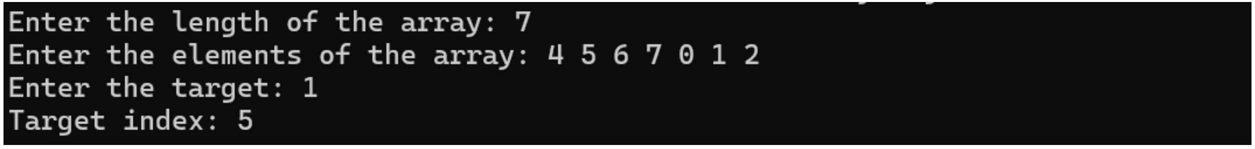
```
import java.util.Scanner;


public class Problem17 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);


        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();
        }


        int peakIndex = findPeakElement(nums);


        System.out.println("Peak element is at index: " + peakIndex);
    }
    public static int findPeakElement(int[] nums) {
        int left = 0, 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;
    }
}
```



```
Enter the length of the array: 7
Enter the elements of the array: 1 2 1 3 5 6 4
Peak element is at index: 5
```

18.Search In Rotated sorted array

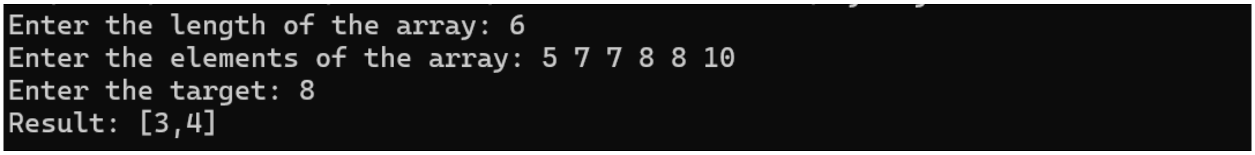
```
import java.util.Scanner;

public class Problem18 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        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();
        }

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

        int targetIndex = search(nums, target);

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

    public static 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 (target >= nums[left] && target < nums[mid]) {
                    right = mid - 1;
                } else {
                    left = mid + 1;
                }
            } else {
                if (target > nums[mid] && target <= nums[right]) {
                    left = mid + 1;
                } else {
                    right = mid - 1;
                }
            }
        }

        return -1;
    }
}
```



```
Enter the length of the array: 7
Enter the elements of the array: 4 5 6 7 0 1 2
Enter the target: 1
Target index: 5
```

19.Find first and last position of Element in Sorted array

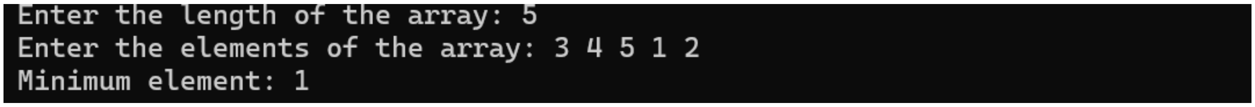
```
import java.util.Scanner;

public class Problem19 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        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();
        }

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

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

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

    public static int[] searchRange(int[] nums, int target) {
        int[] result = {-1, -1};

        // Find the starting position
        result[0] = binarySearch(nums, target, true);

        // Find the ending position
        result[1] = binarySearch(nums, target, false);

        return result;
    }

    private static int binarySearch(int[] nums, int target, boolean findStart) {
        int left = 0, right = nums.length - 1;
        int result = -1;
        while (left <= right) {
            int mid = left + (right - left) / 2;

            if (nums[mid] == target) {
                result = mid;
                if (findStart) {
                    right = mid - 1;
                } else {
                    left = mid + 1;
                }
            } else if (nums[mid] < target) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }

        return result;
    }
}
```



```
Enter the length of the array: 6
Enter the elements of the array: 5 7 7 8 8 10
Enter the target: 8
Result: [3,4]
```

20.Find Minimum in Rotated Sorted Array

```
import java.util.Scanner;

public class Problem20 {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        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();
        }
        int minElement = findMin(nums);
        System.out.println("Minimum element: " + minElement);
    }

    public static 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];
}
}
```



```
Enter the length of the array: 5
Enter the elements of the array: 3 4 5 1 2
Minimum element: 1
```