PRACTISE-SET-10

B Abhimanyu (22CB006)

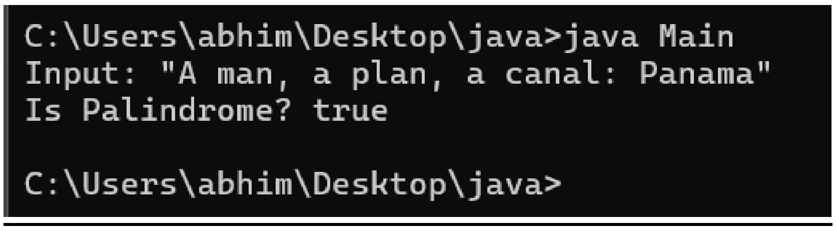
TWO POINTERS

1) valid palindrome

CODE

```java
public class Main {
   public boolean isPalindrome(String s) {
      if (s.length() <= 1) return true;
      StringBuilder n = new StringBuilder();
      for (int i = 0; i < s.length(); i++) {
         if (Character.isLetterOrDigit(s.charAt(i))) {
            n.append(s.charAt(i));
         }
      }
      String filteredString = n.toString().toLowerCase();
      int start = 0, end = filteredString.length() - 1;
      while (start <= end) {
         if (filteredString.charAt(start) != filteredString.charAt(end)) return false;
         start++;
         end--;
      }
      return true;
   }
   public static void main(String[] args) {
      Main solution = new Main();
      String input = "A man, a plan, a canal: Panama";
      boolean result = solution.isPalindrome(input);
      System.out.println("Input: \"" + input + "\"");
      System.out.println("Is Palindrome? " + result);
   }
}
```

OUTPUT



```
C:\Users\abhim\Desktop\java>java Main
Input: "A man, a plan, a canal: Panama"
Is Palindrome? true

C:\Users\abhim\Desktop\java>
```

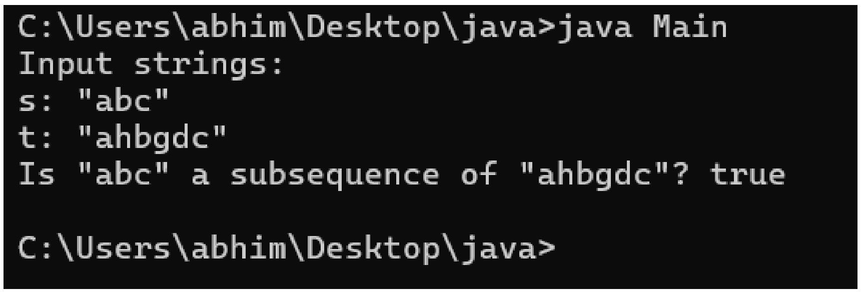
Time Complexity: O(n)
Space Complexity:O(n)

2) is subsequence

CODE

```
public class Main {
  public boolean isSubsequence(String s, String t) {
    if (s.length() == 0) return true;
    int pt = 0, end = s.length();
    for (int i = 0; i < t.length(); i++) {
      if (t.charAt(i) == s.charAt(pt)) pt++;
      if (pt == end) return true;
    }
    return false;
  }
  public static void main(String[] args) {
    Main solution = new Main();
    String s = "abc";
    String t = "ahbgdc";
    boolean result = solution.isSubsequence(s, t);
    System.out.println("Input strings:");
    System.out.println("s: \"" + s + "\"");
    System.out.println("t: \"" + t + "\"");
    System.out.println("Is \"" + s + "\" a subsequence of \"" + t + "\"? " + result);
  }
}
```

OUTPUT



```
C:\Users\abhim\Desktop\java>java Main
Input strings:
s: "abc"
t: "ahbgdc"
Is "abc" a subsequence of "ahbgdc"? true

C:\Users\abhim\Desktop\java>
```

Time Complexity:O(n)

Space Complexity:O(1)

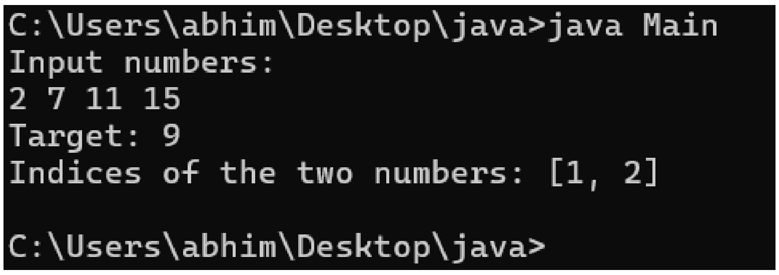
3) Two Sum II – Input array is Sorted

CODE

```
public class Main {
  public int[] twoSum(int[] numbers, int target) {
    int start = 0, end = numbers.length - 1;
    while (start < end) {
      int temp = numbers[start] + numbers[end];
      if (temp == target) {
        return new int[]{start + 1, end + 1};
      } else if (temp > target) {
        end--;
```

```
            } else {
                start++;
            }
        }
        return new int[]{-1, -1};
    }
    public static void main(String[] args) {
        Main main = new Main();
        int[] numbers = {2, 7, 11, 15};
        int target = 9;
        int[] result = main.twoSum(numbers, target);
        System.out.println("Input numbers: ");
        for (int num : numbers) {
            System.out.print(num + " ");
        }
        System.out.println("\nTarget: " + target);
        System.out.println("Indices of the two numbers: [" + result[0] + ", " + result[1] + "]");
    }
}
```

OUTPUT



```
C:\Users\abhim\Desktop\java>java Main
Input numbers:
2 7 11 15
Target: 9
Indices of the two numbers: [1, 2]

C:\Users\abhim\Desktop\java>
```

Time Complexity:O(n)

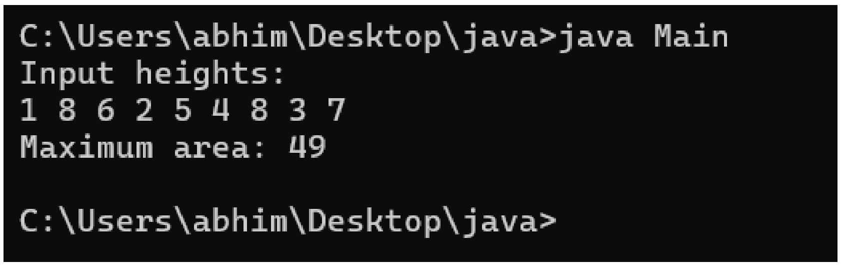
Space Complexity:O(1)

4) container with most water

CODE

```
public class Main {
    public int maxArea(int[] height) {
        int maxarea = 0, start = 0, end = height.length - 1;
        while (start < end) {
            int area = (end - start) * Math.min(height[start], height[end]);
            maxarea = Math.max(maxarea, area);
            if (height[start] == height[end]) {
                start++;
                end--;
            } else if (height[start] > height[end]) {
                end--;
            } else {
                start++;
            }
        }
        return maxarea;
    }
    public static void main(String[] args) {
        Main main = new Main();
        int[] height = {1, 8, 6, 2, 5, 4, 8, 3, 7};
        int result = main.maxArea(height);
        System.out.println("Input heights: ");
        for (int h : height) {
            System.out.print(h + " ");
        }
        System.out.println("\nMaximum area: " + result);
    }
}
```

OUTPUT



```
C:\Users\abhim\Desktop\java>java Main
Input heights:
1 8 6 2 5 4 8 3 7
Maximum area: 49

C:\Users\abhim\Desktop\java>
```

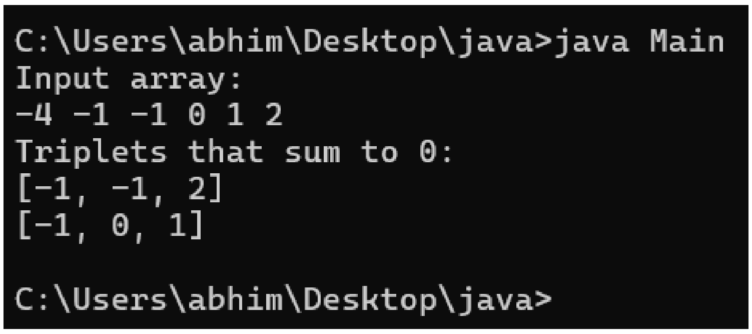
Time Complexity:O(n)
Space Complexity:O(1)

5) 3sum

CODE

```
import java.util.*;
public class Main {
    public List<List<Integer>> threeSum(int[] nums) {
        Arrays.sort(nums);
        List<List<Integer>> ans = new ArrayList<>();
        int n = nums.length;
        for (int i = 0; i < n - 2; i++) {
            if (i > 0 && nums[i] == nums[i - 1]) continue;
            int start = i + 1, end = n - 1;
            while (start < end) {
                int temp = nums[i] + nums[start] + nums[end];
                if (temp == 0) {
                    ans.add(Arrays.asList(nums[i], nums[start], nums[end]));
                    while (start < end && nums[start] == nums[start + 1]) start++;
                    while (start < end && nums[end] == nums[end - 1]) end--;
                    start++;
                    end--;
                } else if (temp > 0) {
                    end--;
                } else {
                    start++;
                }
            }
        }
        return ans;
    }
    public static void main(String[] args) {
        Main main = new Main();
        int[] nums = {-1, 0, 1, 2, -1, -4};
        List<List<Integer>> result = main.threeSum(nums);
        System.out.println("Input array: ");
        for (int num : nums) {
            System.out.print(num + " ");
        }
        System.out.println("\nTriplets that sum to 0:");
        for (List<Integer> triplet : result) {
            System.out.println(triplet);
        }
    }
}
```

OUTPUT



```
C:\Users\abhim\Desktop\java>java Main
Input array:
-4 -1 -1 0 1 2
Triplets that sum to 0:
[-1, -1, 2]
[-1, 0, 1]

C:\Users\abhim\Desktop\java>
```

Time Complexity:O(n^2)

Space Complexity:O(n)

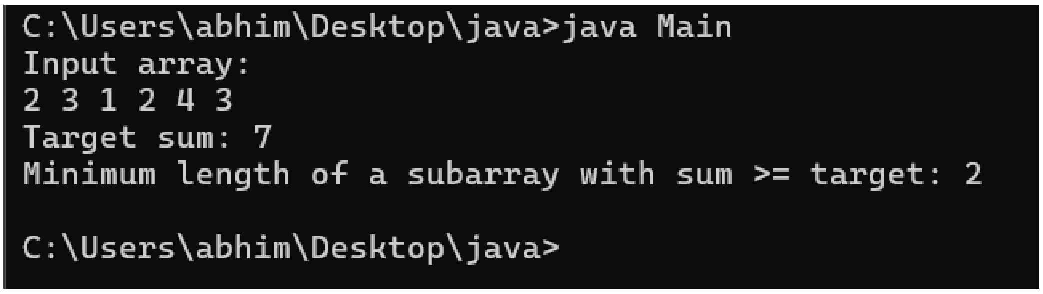
SLIDING WINDOW

6) Minimum size subarray sum

CODE

```
public class Main {
    public int minSubArrayLen(int target, int[] nums) {
        int left = 0, ans = Integer.MAX_VALUE, sum = 0;
        for (int start = 0; start < nums.length; start++) {
            sum += nums[start];
            while (left <= start && sum >= target) {
                ans = Math.min(start - left + 1, ans);
                sum -= nums[left];
                left++;
            }
        }
        if (ans == Integer.MAX_VALUE) return 0;
        return ans;
    }
    public static void main(String[] args) {
        Main main = new Main();
        int target = 7;
        int[] nums = {2, 3, 1, 2, 4, 3};
        int result = main.minSubArrayLen(target, nums);
        System.out.println("Input array: ");
        for (int num : nums) {
            System.out.print(num + " ");
        }
        System.out.println("\nTarget sum: " + target);
        System.out.println("Minimum length of a subarray with sum >= target: " + result);
    }
}
```

<u>OUTPUT</u>



```
C:\Users\abhim\Desktop\java>java Main
Input array:
2 3 1 2 4 3
Target sum: 7
Minimum length of a subarray with sum >= target: 2

C:\Users\abhim\Desktop\java>
```

Time Complexity:O(n)

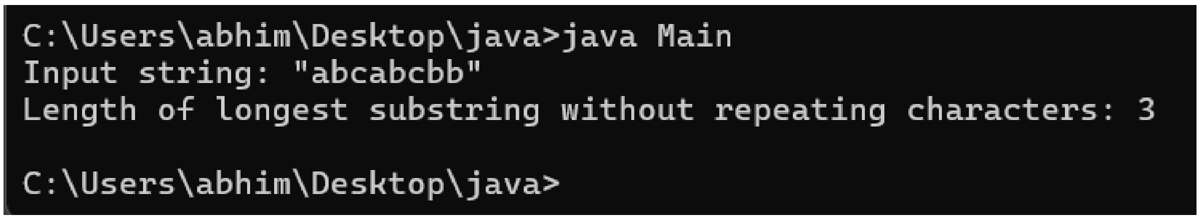
Space Complexity:O(1)

7) longest substring without repeating characters

<u>CODE</u>

```
import java.util.*;
public class Main {
    public int lengthOfLongestSubstring(String s) {
        Map<Character, Integer> map = new HashMap<>();
        int left = 0, len = 0;
        for (int right = 0; right < s.length(); right++) {
            char i = s.charAt(right);
            map.put(i, map.getOrDefault(i, 0) + 1);
            while (map.get(i) > 1) {
                char leftChar = s.charAt(left);
                map.put(leftChar, map.get(leftChar) - 1);
                left++;
            }
            len = Math.max(len, right - left + 1);
        }
        return len;
    }
    public static void main(String[] args) {
        Main main = new Main();
        String s = "abcabcbb";
        int result = main.lengthOfLongestSubstring(s);
        System.out.println("Input string: \"" + s + "\"");
        System.out.println("Length of longest substring without repeating characters: " + result);
    }
}
```

OUTPUT



```
C:\Users\abhim\Desktop\java>java Main
Input string: "abcabcbb"
Length of longest substring without repeating characters: 3

C:\Users\abhim\Desktop\java>
```

Time Complexity:O(n)

Space Complexity:O(n)

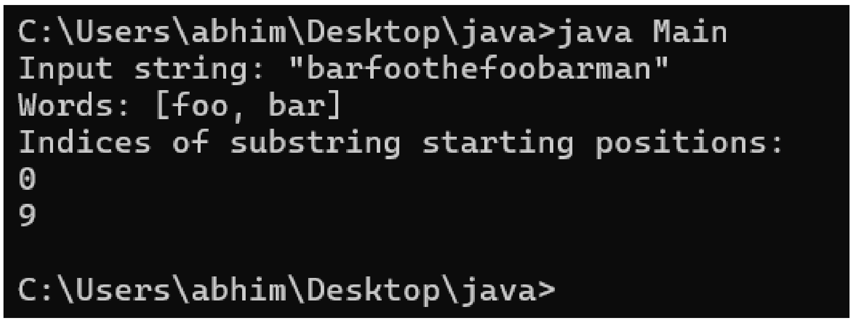
8) substring with concatenation of all words

CODE

```
import java.util.*;
public class Main {
    public List<Integer> findSubstring(String s, String[] words) {
        List<Integer> result = new ArrayList<>();
        if (s == null || words == null || words.length == 0) {
            return result;
        }
        Map<String, Integer> wordCount = new HashMap<>();
        for (String word : words) {
            wordCount.put(word, wordCount.getOrDefault(word, 0) + 1);
        }

        int wordLen = words[0].length();
        int totalLen = words.length * wordLen;
        for (int i = 0; i < wordLen; i++) {
            int left = i;
            int count = 0;
            Map<String, Integer> seen = new HashMap<>();
            for (int right = i; right + wordLen <= s.length(); right += wordLen) {
                String word = s.substring(right, right + wordLen);
                if (wordCount.containsKey(word)) {
                    seen.put(word, seen.getOrDefault(word, 0) + 1);
                    count++;
                    while (seen.get(word) > wordCount.get(word)) {
                        String leftWord = s.substring(left, left + wordLen);
                        seen.put(leftWord, seen.get(leftWord) - 1);
                        count--;
                        left += wordLen;
                    }
                    if (count == words.length) {
                        result.add(left);
                    }
                } else {
                    seen.clear();
                    count = 0;
                    left = right + wordLen;
                }
            }
        }
        return result;
    }
    public static void main(String[] args) {
        Main main = new Main();
        String s = "barfoothefoobarman";
        String[] words = {"foo", "bar"};
        List<Integer> result = main.findSubstring(s, words);
        System.out.println("Input string: \"" + s + "\"");
        System.out.println("Words: " + Arrays.toString(words));
        System.out.println("Indices of substring starting positions:");
        for (int index : result) {
            System.out.println(index);
        }
    }
}
```

OUTPUT



```
C:\Users\abhim\Desktop\java>java Main
Input string: "barfoothefoobarman"
Words: [foo, bar]
Indices of substring starting positions:
0
9

C:\Users\abhim\Desktop\java>
```

Time Complexity:O(n*m)

Space Complexity:O(n+m)

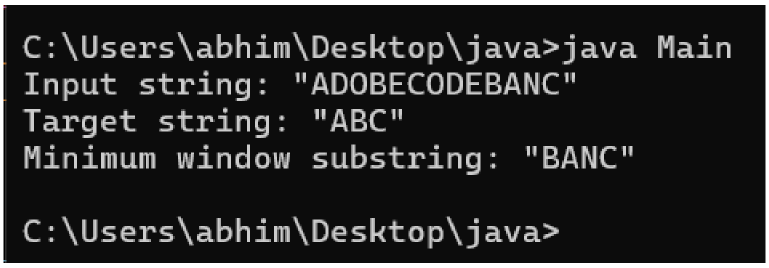
9) minimum window substring

CODE

```
import java.util.*;
public class Main {
  public boolean check(Map<Character, Integer> a, Map<Character, Integer> b) {
    for (char i : a.keySet()) {
      if (!b.containsKey(i) || a.get(i) > b.get(i)) {
        return false;
      }
    }
```

```
        return true;
    }
    public String minWindow(String s, String t) {
        if (s == null || t == null || s.length() < t.length()) return "";
        Map<Character, Integer> tmap = new HashMap<>();
        int minlen = Integer.MAX_VALUE, left = 0;
        for (char i : t.toCharArray()) {
            tmap.put(i, tmap.getOrDefault(i, 0) + 1);
        }
        int end = -1, minStart = 0;
        Map<Character, Integer> smap = new HashMap<>();
        for (int right = 0; right < s.length(); right++) {
            smap.put(s.charAt(right), smap.getOrDefault(s.charAt(right), 0) + 1);
            while (check(tmap, smap) && left <= right) {
                if (right - left + 1 < minlen) {
                    minlen = right - left + 1;
                    minStart = left;
                    end = right;
                }
                smap.put(s.charAt(left), smap.get(s.charAt(left)) - 1);
                left++;
            }
        }
        return minlen == Integer.MAX_VALUE ? "" : s.substring(minStart, end + 1);
    }
    public static void main(String[] args) {
        Main main = new Main();
        String s = "ADOBECODEBANC";
        String t = "ABC";
        String result = main.minWindow(s, t);
        System.out.println("Input string: \"" + s + "\"");
        System.out.println("Target string: \"" + t + "\"");
        System.out.println("Minimum window substring: \"" + result + "\"");
    }
}
```

OUTPUT



```
C:\Users\abhim\Desktop\java>java Main
Input string: "ADOBECODEBANC"
Target string: "ABC"
Minimum window substring: "BANC"

C:\Users\abhim\Desktop\java>
```

Time Complexity:O(n+m)

Space Complexity: O(n+m)

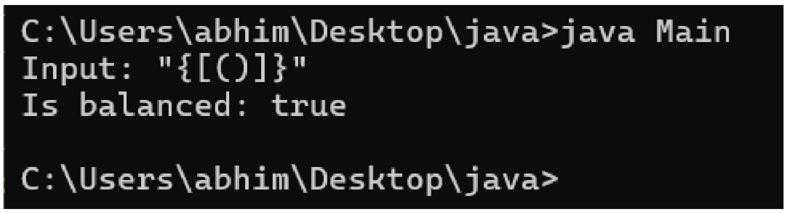
STACK

10) valid paranthesis

CODE

```
import java.util.*;
public class Main {
   static boolean isParenthesisBalanced(String s) {
      if (s.charAt(0) == '}' || s.charAt(0) == ']' || s.charAt(0) == ')') {
         return false;
      }
      Stack<Character> st = new Stack<>();
      for (int i = 0; i < s.length(); i++) {
         if (s.charAt(i) == '[' || s.charAt(i) == '{' || s.charAt(i) == '(') {
            st.push(s.charAt(i));
         } else {
            if (st.isEmpty()) {
               return false;
            }
            char topele = st.peek();
            if (topele == '{' && s.charAt(i) == '}') {
               st.pop();
            } else if (topele == '[' && s.charAt(i) == ']') {
               st.pop();
            } else if (topele == '(' && s.charAt(i) == ')') {
               st.pop();
            } else {
               return false;
            }
         }
      }
      return st.isEmpty();
   }
   public static void main(String[] args) {
      String s = "{[()]}";
      boolean result = isParenthesisBalanced(s);
      System.out.println("Input: \"" + s + "\"");
      System.out.println("Is balanced: " + result);
   }
}
```

OUTPUT



```
C:\Users\abhim\Desktop\java>java Main
Input: "{[()]}"
Is balanced: true

C:\Users\abhim\Desktop\java>
```

Time Complexity:O(n)

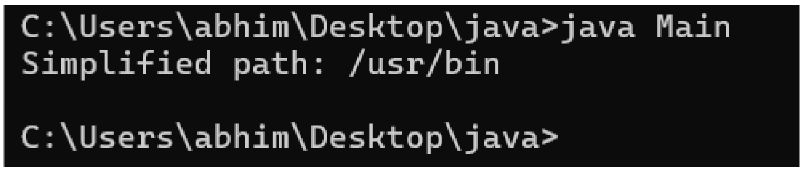
Space Complexity:O(n)

11) simplify path

CODE

```
import java.util.*;
public class Main {
    public String simplifyPath(String path) {
        Stack<String> st = new Stack<>();
        String[] lst = path.split("/");
        for (String i : lst) {
            if (i.equals("") || i.equals(".") || i.equals(" ")) continue;
            if (i.equals("..")) {
                if (!st.isEmpty()) st.pop();
            } else {
                st.push(i);
            }
        }
        if (st.isEmpty()) return "/";
        StringBuilder ans = new StringBuilder();
        while (!st.isEmpty()) {
            String i = st.remove(0);
            ans.append("/").append(i);
        }
        return ans.toString();
    }
    public static void main(String[] args) {
        Main main = new Main();
        String path = "/home/../usr//bin/";
        String result = main.simplifyPath(path);
        System.out.println("Simplified path: " + result);
    }
}
```

OUTPUT



```
C:\Users\abhim\Desktop\java>java Main
Simplified path: /usr/bin

C:\Users\abhim\Desktop\java>
```

Time Complexity:O(n)
Space Complexity:O(n)

12) minstack

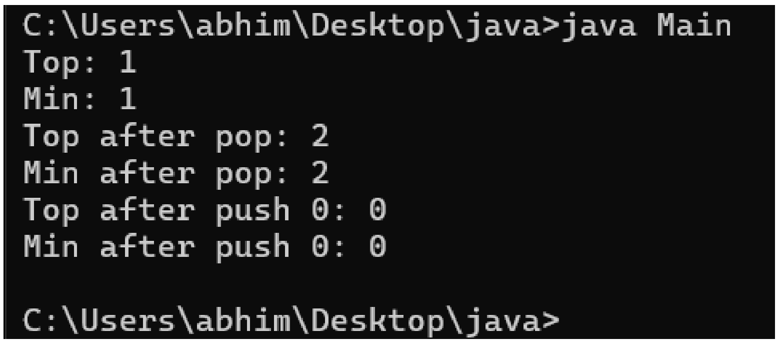
CODE

```
import java.util.*;
```

```
public class Main {
    static 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) {
        MinStack minStack = new MinStack();
        minStack.push(3);
        minStack.push(4);
        minStack.push(2);
        minStack.push(1);
        System.out.println("Top: " + minStack.top());
        System.out.println("Min: " + minStack.getMin());
        minStack.pop();
        System.out.println("Top after pop: " + minStack.top());
        System.out.println("Min after pop: " + minStack.getMin());
        minStack.push(0);
        System.out.println("Top after push 0: " + minStack.top());
        System.out.println("Min after push 0: " + minStack.getMin());
    }
}
```

<u>OUTPUT</u>



```
C:\Users\abhim\Desktop\java>java Main
Top: 1
Min: 1
Top after pop: 2
Min after pop: 2
Top after push 0: 0
Min after push 0: 0

C:\Users\abhim\Desktop\java>
```

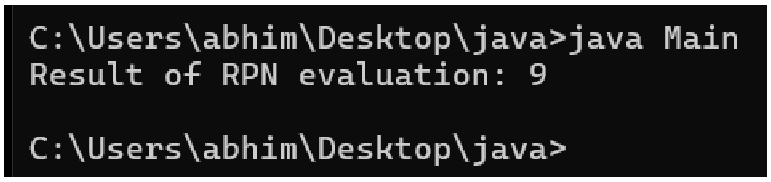
Time Complexity:O(1)
Space Complexity:O(n)

13) Evaluate reverse polish notation

CODE

```
import java.util.*;
public class Main {
   public boolean isNumeric(String i) {
      try {
         Integer.parseInt(i);
         return true;
      } catch (NumberFormatException e) {
         return false;
      }
   }
   public int evalRPN(String[] tokens) {
      Stack<Integer> st = new Stack<>();
      for (String i : tokens) {
         if (isNumeric(i)) {
            st.push(Integer.parseInt(i));
         } else {
            int sec = st.pop();
            int first = st.pop();
            switch (i) {
               case "+":
                  st.push(first + sec);
                  break;
               case "-":
                  st.push(first - sec);
                  break;
               case "*":
                  st.push(first * sec);
                  break;
               case "/":
                  st.push(first / sec);
                  break;
            }
         }
      }
      return st.pop();
   }
   public static void main(String[] args) {
      Main main = new Main();
      String[] tokens = {"2", "1", "+", "3", "*"};
      int result = main.evalRPN(tokens);
      System.out.println("Result of RPN evaluation: " + result);
   }
}
```

OUTPUT



```
C:\Users\abhim\Desktop\java>java Main
Result of RPN evaluation: 9

C:\Users\abhim\Desktop\java>
```

Time Complexity:O(n)

Space Complexity:O(n)

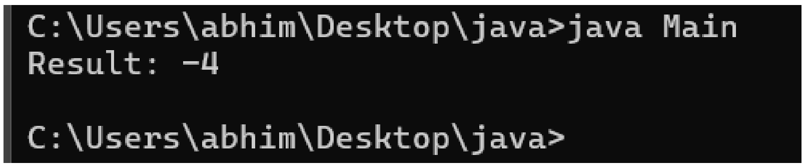
14) basic calculator

CODE

```
import java.util.*;
public class Main {
   public int calculate(String s) {
      int result = 0;
      int currnum = 0;
      int sign = 1;
      Stack<Integer> st = new Stack<>();
      for (char i : s.toCharArray()) {
         if (Character.isDigit(i)) {
            currnum = currnum * 10 + (i - '0');
         } else if (i == '+') {
            result += sign * currnum;
            sign = 1;
            currnum = 0;
         } else if (i == '-') {
            result += sign * currnum;
            sign = -1;
            currnum = 0;
         } else if (i == '(') {
            st.push(result);
            st.push(sign);
            result = 0;
            sign = 1;
         } else if (i == ')') {
            result += sign * currnum;
            result *= st.pop();
            result += st.pop();
            currnum = 0;
         }
      }
      result += sign * currnum;
      return result;
   }
   public static void main(String[] args) {
      Main main = new Main();
      String expression = "1 + (2 - (3 + 4))";
      int result = main.calculate(expression);
```

```
        System.out.println("Result: " + result);  // Should print -4
    }
}
```

OUTPUT



```
C:\Users\abhim\Desktop\java>java Main
Result: -4

C:\Users\abhim\Desktop\java>
```

Time Complexity:O(n)

Space Complexity:O(n)

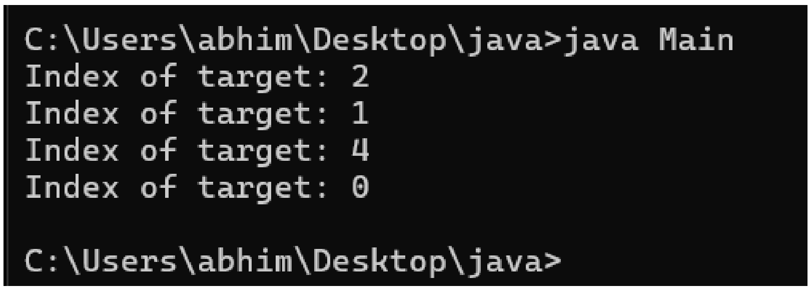
BINARY SEARCH

15) search insert position

CODE

```
public class Main {
    public int searchInsert(int[] nums, int target) {
        int left = 0, right = nums.length - 1;
        while (left <= right) {
            int mid = (left + right) / 2;
            if (target == nums[mid]) return mid;
            if (target < nums[mid]) {
                right = mid - 1;
            } else {
                left = mid + 1;
            }
        }
        return left;
    }
    public static void main(String[] args) {
        Main main = new Main();
        int[] nums = {1, 3, 5, 6};
        int target = 5;
        int result = main.searchInsert(nums, target);
        System.out.println("Index of target: " + result);  // Should print 2
        target = 2;
        result = main.searchInsert(nums, target);
        System.out.println("Index of target: " + result);  // Should print 1
        target = 7;
        result = main.searchInsert(nums, target);
        System.out.println("Index of target: " + result);  // Should print 4
        target = 0;
        result = main.searchInsert(nums, target);
        System.out.println("Index of target: " + result);  // Should print 0
    }
}
```

OUTPUT



```
C:\Users\abhim\Desktop\java>java Main
Index of target: 2
Index of target: 1
Index of target: 4
Index of target: 0

C:\Users\abhim\Desktop\java>
```

Time Complexity:O(logn)
Space Complexity:O(1)

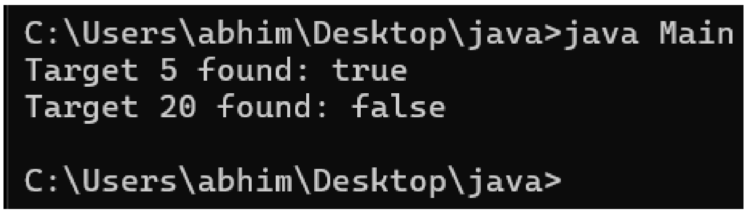
16) search in 2D matrix

CODE

```
public class Main {
  public boolean searchMatrix(int[][] matrix, int target) {
    int rowStart = 0, colStart = 0, rowEnd = matrix.length - 1, colEnd = matrix[0].length - 1;
    while (rowStart <= rowEnd) {
      int mid = (rowStart + rowEnd) / 2;
      if (target >= matrix[mid][0] && target <= matrix[mid][colEnd]) {
        colStart = 0;
        colEnd = matrix[0].length - 1;
        while (colStart <= colEnd) {
          int mid2 = (colStart + colEnd) / 2;
          if (matrix[mid][mid2] == target) return true;
          if (matrix[mid][mid2] > target) {
            colEnd = mid2 - 1;
          } else {
            colStart = mid2 + 1;
          }
        }
        return false;
      }
      if (target < matrix[mid][0]) {
        rowEnd = mid - 1;
      } else {
        rowStart = mid + 1;
      }
    }
    return false;
  }
  public static void main(String[] args) {
    Main main = new Main();
    int[][] matrix = {
      {1, 4, 7, 11},
      {2, 5, 8, 12},
      {3, 6, 9, 16},
      {10, 13, 14, 17}
    };
```

```
        int target = 5;
        boolean result = main.searchMatrix(matrix, target);
        System.out.println("Target " + target + " found: " + result);  // Should print true
        target = 20;
        result = main.searchMatrix(matrix, target);
        System.out.println("Target " + target + " found: " + result);  // Should print false
    }
}
```

OUTPUT



```
C:\Users\abhim\Desktop\java>java Main
Target 5 found: true
Target 20 found: false

C:\Users\abhim\Desktop\java>
```

Time Complexity:O(logn)

Space Complexity:O(1)

17)   find peak element

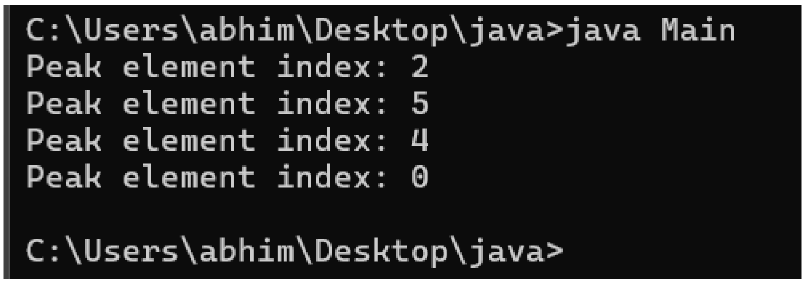
CODE

```
public class Main {
    public 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]) {
                left = mid + 1;
            } else {
                right = mid;
            }
        }
        return left;
    }
    public static void main(String[] args) {
        Main main = new Main();
        int[] nums = {1, 2, 3, 1};
        int result = main.findPeakElement(nums);
        System.out.println("Peak element index: " + result);

        nums = new int[]{1, 2, 1, 3, 5, 6, 4};
        result = main.findPeakElement(nums);
        System.out.println("Peak element index: " + result);

        nums = new int[]{1, 2, 3, 4, 5};
        result = main.findPeakElement(nums);
        System.out.println("Peak element index: " + result);
        nums = new int[]{5, 4, 3, 2, 1};
        result = main.findPeakElement(nums);
        System.out.println("Peak element index: " + result);
    }
}
```

OUTPUT



```
C:\Users\abhim\Desktop\java>java Main
Peak element index: 2
Peak element index: 5
Peak element index: 4
Peak element index: 0

C:\Users\abhim\Desktop\java>
```

Time Complexity:O(logn)

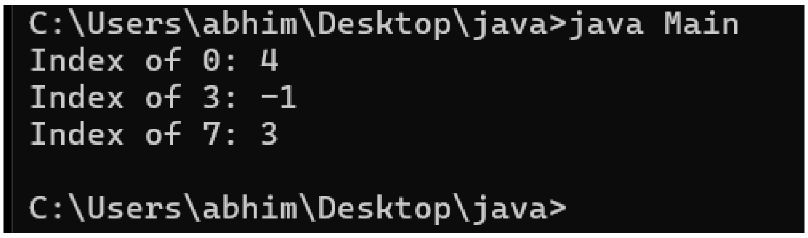
Space Complexity:O(1)

18) search in rotated sorted array

CODE

```
public class Main {
    public int search(int[] arr, int key) {
        int start = 0;
        int end = arr.length - 1;
        while (start <= end) {
            int mid = (start + end) / 2;
            if (arr[mid] == key) {
                return mid;
            }
            if (arr[start] <= arr[mid]) {
                if (arr[start] <= key && key < arr[mid]) {
                    end = mid - 1;
                } else {
                    start = mid + 1;
                }
            } else {
                if (arr[mid] < key && key <= arr[end]) {
                    start = mid + 1;
                } else {
                    end = mid - 1;
                }
            }
        }
        return -1;
    }
    public static void main(String[] args) {
        Main main = new Main();
        int[] arr = {4, 5, 6, 7, 0, 1, 2};
        int key = 0;
        int result = main.search(arr, key);
        System.out.println("Index of " + key + ": " + result);
        key = 3;
        result = main.search(arr, key);
        System.out.println("Index of " + key + ": " + result);
        arr = new int[]{1, 3, 5, 7, 9, 11, 13};
        key = 7;
        result = main.search(arr, key);
        System.out.println("Index of " + key + ": " + result);
    }
}
```

OUTPUT



```
C:\Users\abhim\Desktop\java>java Main
Index of 0: 4
Index of 3: -1
Index of 7: 3

C:\Users\abhim\Desktop\java>
```

Time Complexity:O(logn)

Space Complexity:O(1)

19) find first and last position of a sorted array

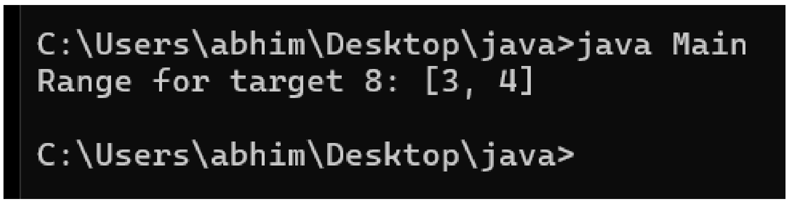
CODE

```
import java.util.Arrays;
public class Main {
    public int[] searchRange(int[] nums, int target) {
        int first = findBound(nums, target, true);
        int last = findBound(nums, target, false);
        return new int[]{first, last};
    }
    private int findBound(int[] nums, int target, boolean isFirst) {
        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 (isFirst) {
                    right = mid - 1;
                } else {
                    left = mid + 1;
                }
            } else if (nums[mid] < target) {
                left = mid + 1;
            } else {
                right = mid - 1;
            }
        }
        return result;
    }
    public static void main(String[] args) {
        Main main = new Main();
        int[] nums1 = {5, 7, 7, 8, 8, 10};
        int target1 = 8;
        int[] result1 = main.searchRange(nums1, target1);
        System.out.println("Range for target " + target1 + ": " + Arrays.toString(result1));
    }
}
```

OUTPUT



```
C:\Users\abhim\Desktop\java>java Main
Range for target 8: [3, 4]

C:\Users\abhim\Desktop\java>
```

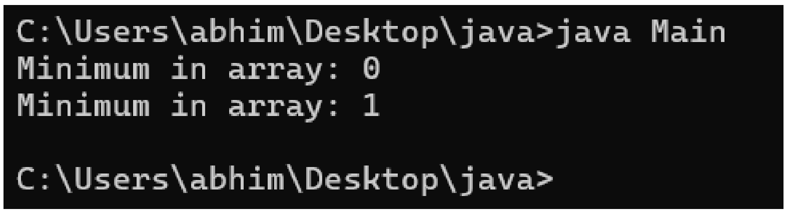
Time Complexity:O(logn)
Space Complexity:O(1)

20) find minimum in rotated sorted array

CODE

```
public class Main {
    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) {
        Main main = new Main();
        int[] nums1 = {4, 5, 6, 7, 0, 1, 2};
        System.out.println("Minimum in array: " + main.findMin(nums1));
        int[] nums2 = {1, 2, 3, 4, 5};
        System.out.println("Minimum in array: " + main.findMin(nums2));
    }
}
```

OUTPUT



```
C:\Users\abhim\Desktop\java>java Main
Minimum in array: 0
Minimum in array: 1

C:\Users\abhim\Desktop\java>
```

Time Complexity:O(logn)

Space Complexity:O(1)