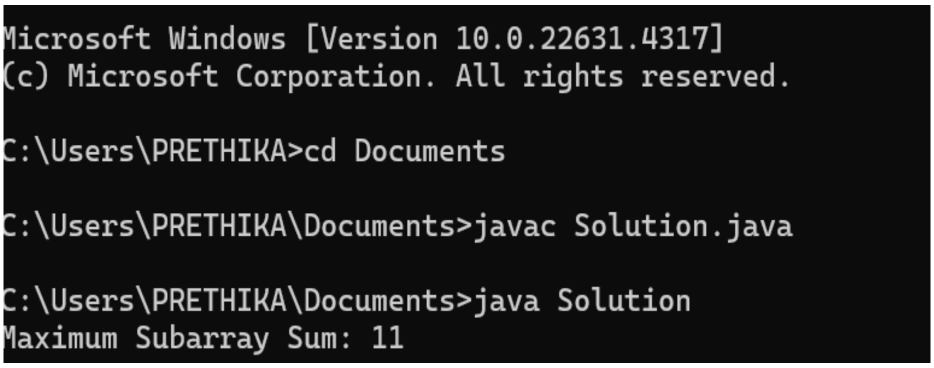
DSA Practice -1

1. Maximum Subarray Sum – Kadane's Algorithm

```
public class Solution {
  public static int maxSubarraySum(int[] arr) {
    int maxCurrent = arr[0];
    int maxGlobal = arr[0];
    for (int i = 1; i < arr.length; i++) {
      maxCurrent = Math.max(arr[i], maxCurrent + arr[i]);
      if (maxCurrent > maxGlobal) {
        maxGlobal = maxCurrent;
      }
    }
    return maxGlobal;
  }
  public static void main(String[] args) {
    int arr[] = {2, 3, -8, 7, -1, 2, 3};
    System.out.println("Maximum Subarray Sum: " + maxSubarraySum(arr));
  }
}
```

Output :



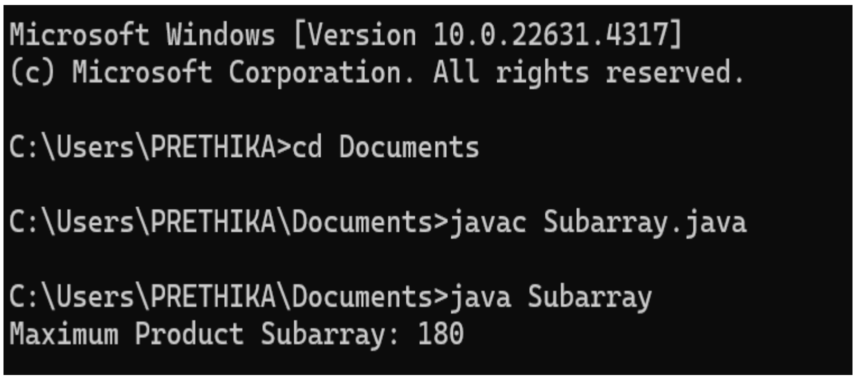
```
licrosoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

:\Users\PRETHIKA>cd Documents

:\Users\PRETHIKA\Documents>javac Solution.java

:\Users\PRETHIKA\Documents>java Solution
laximum Subarray Sum: 11
```

Time Complexity : O(n)

2. Maximum Product Subarray

```
public class Subarray {
  public static int maxProductSubarray(int[] arr) {
      if (arr.length == 0) return 0;
      int maxProduct = arr[0];
      int minProduct = arr[0];
      int result = arr[0];
      for (int i = 1; i < arr.length; i++) {
        int current = arr[i];
        if (current < 0) {
          int temp = maxProduct;
          maxProduct = minProduct;
          minProduct = temp;
        }
        maxProduct = Math.max(current, maxProduct * current);
        minProduct = Math.min(current, minProduct * current);
        result = Math.max(result, maxProduct);
     }
     return result;
  }
  public static void main(String[] args) {
     int arr[] = {-2, 6, -3, -10, 0, 2};
     System.out.println("Maximum Product Subarray: " + maxProductSubarray(arr));
  }
}
```

Output :



```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\PRETHIKA>cd Documents

C:\Users\PRETHIKA\Documents>javac Subarray.java

C:\Users\PRETHIKA\Documents>java Subarray
Maximum Product Subarray: 180
```

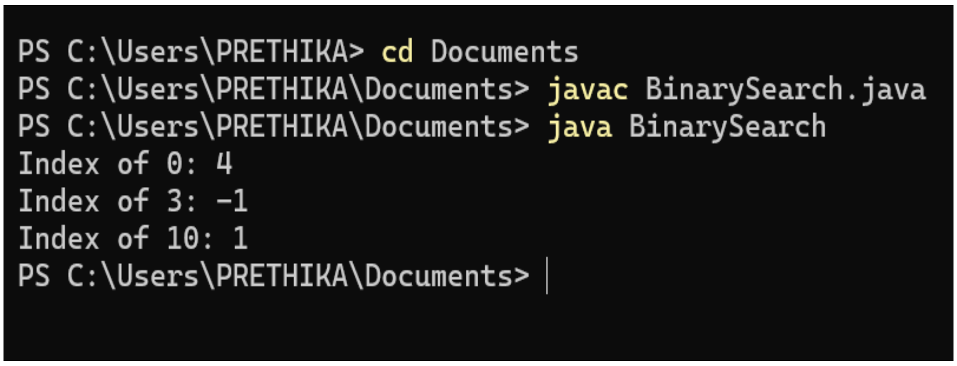
Time Complexity : O(n)

3. Search in a sorted and rotated Arra

```
public class BinarySearch {
  public static int search(int arr[], int key) {
  int low = 0;
  int high = arr.length - 1;
  while (low <= high) {
    int mid = low + (high - low) / 2;
    if (arr[mid] == key) {
      return mid;
    }
    if (arr[low] <= arr[mid]) {
      if (key >= arr[low] && key < arr[mid]) {
          high = mid - 1;
      } else {
          low = mid + 1;
      }
    }
    else {
      if (key > arr[mid] && key <= arr[low]) {
          low = mid + 1;
      } else {
          high = mid - 1;
      }
      }
   }
   return -1;
 }
 public static void main(String[] args) {
   int[] arr1 = {4, 5, 6, 7, 0, 1, 2};
   int key1 = 0;
   System.out.println("Index of " + key1 + ": " + search(arr1, key1));
   int[] arr2 = {4, 5, 6, 7, 0, 1, 2};
```

```
    int key2 = 3;
    System.out.println("Index of " + key2 + ": " + search(arr2, key2));
    int[] arr3 = {50, 10, 20, 30, 40};
    int key3 = 10;
    System.out.println("Index of " + key3 + ": " + search(arr3, key3));
  }
}
```

Output :



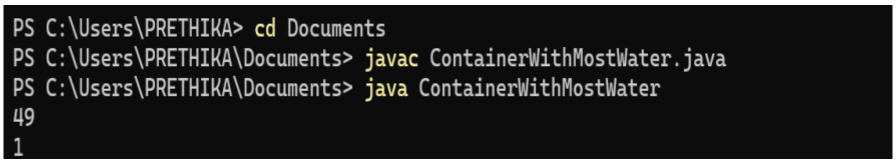
```
PS C:\Users\PRETHIKA> cd Documents
PS C:\Users\PRETHIKA\Documents> javac BinarySearch.java
PS C:\Users\PRETHIKA\Documents> java BinarySearch
Index of 0: 4
Index of 3: -1
Index of 10: 1
PS C:\Users\PRETHIKA\Documents>
```

Time Complexity : O(log n)

4. Container with Most Water

```
public class ContainerWithMostWater {
    public static int maxArea(int[] height) {
        int left = 0, right = height.length - 1;
        int maxArea = 0;
        while (left < right) {
            int currentArea = Math.min(height[left], height[right]) * (right - left);
            maxArea = Math.max(maxArea, currentArea);
            if (height[left] < height[right]) {
                left++;
            } else {
                right--;
            }
        }
        return maxArea;
    }
    public static void main(String[] args) {
        int[] height1 = {1, 8, 6, 2, 5, 4, 8, 3, 7};
        int[] height2 = {1, 1};
        System.out.println(maxArea(height1));
        System.out.println(maxArea(height2));
    }
}
```

Output :



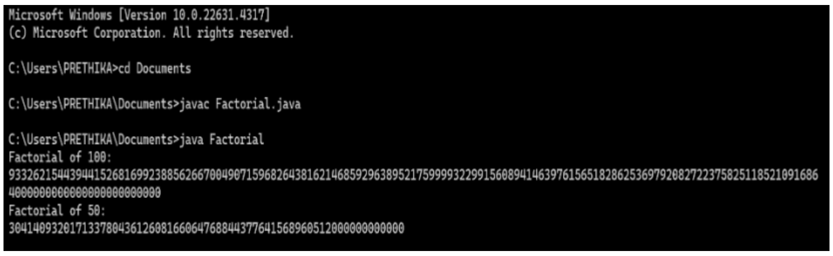
```
PS C:\Users\PRETHIKA> cd Documents
PS C:\Users\PRETHIKA\Documents> javac ContainerWithMostWater.java
PS C:\Users\PRETHIKA\Documents> java ContainerWithMostWater
49
1
```

Time Complexity : O(n)

5. Find the Factorial of a large number

```
import java.math.BigInteger;
class Factorial {
        public static BigInteger factorial(int n) {
        BigInteger result = BigInteger.ONE;
        for (int i = 2; i <= n; i++) {
            result = result.multiply(BigInteger.valueOf(i));
        }
        return result;
    }
    public static void main(String[] args) {
        int num1 = 100;
        System.out.println("Factorial of " + num1 + ":");
        System.out.println(factorial(num1));
        int num2 = 50;
        System.out.println("Factorial of " + num2 + ":");
        System.out.println(factorial(num2));
    }
}
```

Output :



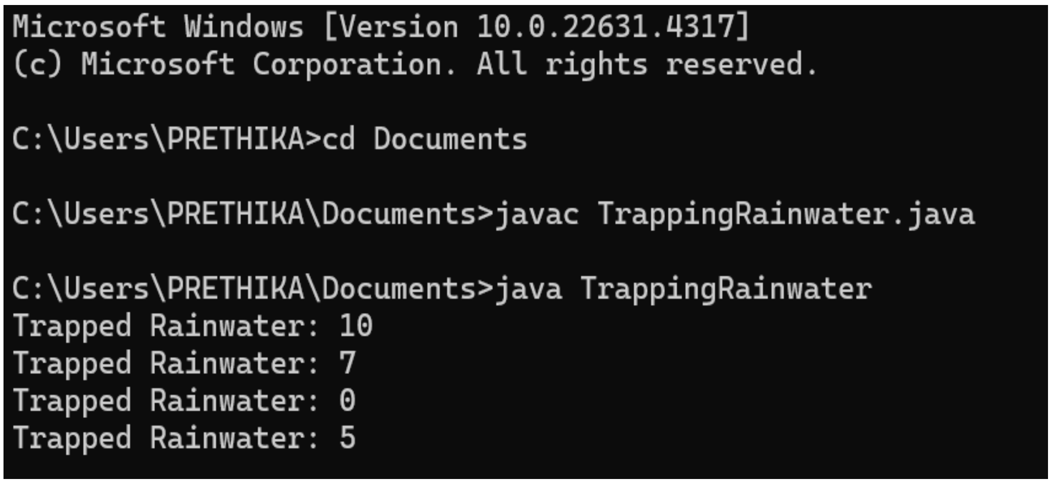
```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\PRETHIKA>cd Documents

C:\Users\PRETHIKA\Documents>javac Factorial.java

C:\Users\PRETHIKA\Documents>java Factorial
Factorial of 100:
9332621544394415268169923885626670049071596826438162146859296389521759999322991560894146397615651828625369792082722375825118521091686
4000000000000000000000000
Factorial of 50:
30414093201713378043612608166064768844377641568960512000000000000
```

Time Complexity : O(n * m)

6. Trapping Rainwater

```
public class TrappingRainwater {
    public static int trap(int[] arr) {
        int n = arr.length;
        if (n == 0) return 0;
        int[] leftMax = new int[n];
        int[] rightMax = new int[n];
        leftMax[0] = arr[0];
        for (int i = 1; i < n; i++) {
            leftMax[i] = Math.max(arr[i], leftMax[i - 1]);
        }
        rightMax[n - 1] = arr[n - 1];
        for (int i = n - 2; i >= 0; i--) {
            rightMax[i] = Math.max(arr[i], rightMax[i + 1]);
        }
        int totalWater = 0;
        for (int i = 0; i < n; i++) {
            totalWater += Math.min(leftMax[i], rightMax[i]) - arr[i];
        }

        return totalWater;
    }
    public static void main(String[] args) {
        int[] arr1 = {3, 0, 1, 0, 4, 0, 2};
        System.out.println("Trapped Rainwater: " + trap(arr1));
        int[] arr2 = {3, 0, 2, 0, 4};
        System.out.println("Trapped Rainwater: " + trap(arr2));
        int[] arr3 = {1, 2, 3, 4};
        System.out.println("Trapped Rainwater: " + trap(arr3));
        int[] arr4 = {10, 9, 0, 5};
        System.out.println("Trapped Rainwater: " + trap(arr4));
    }
}
```

Output :



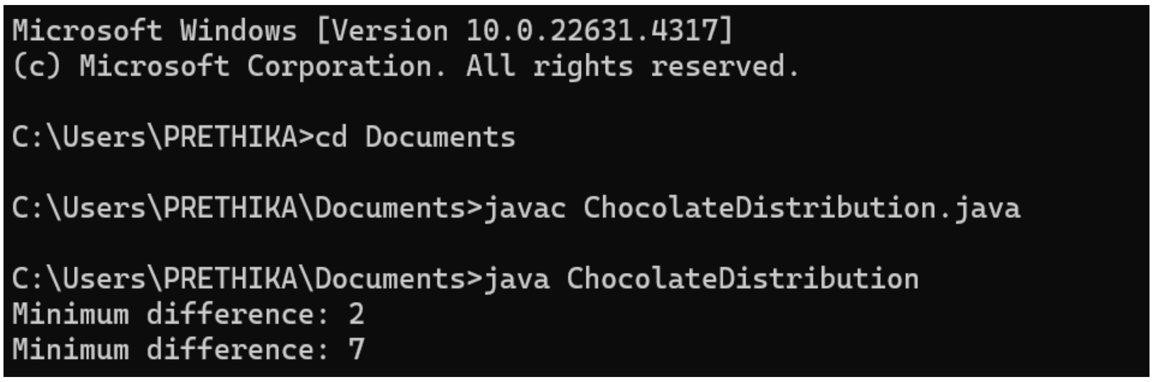
```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\PRETHIKA>cd Documents

C:\Users\PRETHIKA\Documents>javac TrappingRainwater.java

C:\Users\PRETHIKA\Documents>java TrappingRainwater
Trapped Rainwater: 10
Trapped Rainwater: 7
Trapped Rainwater: 0
Trapped Rainwater: 5
```

Time Complexity : O(n)

7. Chocolate Distribution Problem

```
import java.util.Arrays;
public class ChocolateDistribution {
    public static int distributeChocolates(int[] arr, int m) {
        int n = arr.length;
        if (n < m) {
            return -1;
        }
        Arrays.sort(arr);
        int minDiff = Integer.MAX_VALUE;
        for (int i = 0; i + m - 1 < n; i++) {
            int diff = arr[i + m - 1] - arr[i];
            minDiff = Math.min(minDiff, diff);
        }
        return minDiff;
    public static void main(String[] args) {
        int[] arr1 = {7, 3, 2, 4, 9, 12, 56};
        int m1 = 3;
        System.out.println("Minimum difference: " + distributeChocolates(arr1, m1));
        int[] arr2 = {7, 3, 2, 4, 9, 12, 56};
        int m2 = 5;
        System.out.println("Minimum difference: " + distributeChocolates(arr2, m2));
    }
}
```

Output :



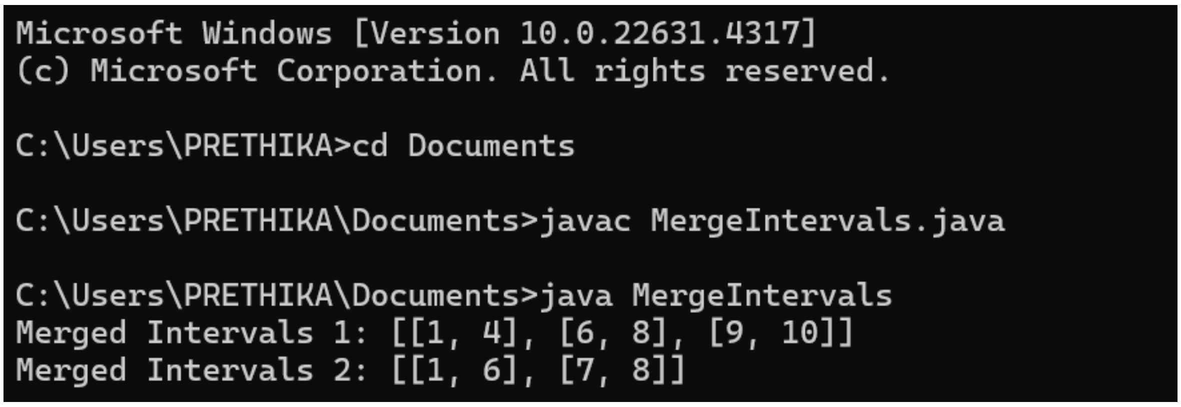
```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\PRETHIKA>cd Documents

C:\Users\PRETHIKA\Documents>javac ChocolateDistribution.java

C:\Users\PRETHIKA\Documents>java ChocolateDistribution
Minimum difference: 2
Minimum difference: 7
```

Time Complexity : O(n log n)

8. Merge Overlapping Intervals

```
import java.util.List;
import java.util.ArrayList;
import java.util.Arrays;
public class MergeIntervals {
    public static List<int[]> mergeIntervals(int[][] intervals) {
        if (intervals.length == 0) {
            return new ArrayList<>();
        }
        Arrays.sort(intervals, (a, b) -> Integer.compare(a[0], b[0]));
        List<int[]> merged = new ArrayList<>();
        int[] current = intervals[0];
        merged.add(current);
        for (int i = 1; i < intervals.length; i++) {
            if (intervals[i][0] <= current[1]) {
                current[1] = Math.max(current[1], intervals[i][1]);
            } else {
                current = intervals[i];
                merged.add(current);
            }
        }
        return merged;
    }
    public static void main(String[] args) {
        int[][] intervals1 = {{1, 3}, {2, 4}, {6, 8}, {9, 10}};
        int[][] intervals2 = {{7, 8}, {1, 5}, {2, 4}, {4, 6}};
        System.out.println("Merged Intervals 1: " +
Arrays.deepToString(mergeIntervals(intervals1).toArray(new int[0][])));
        System.out.println("Merged Intervals 2: " +
Arrays.deepToString(mergeIntervals(intervals2).toArray(new int[0][])));
    }
}
```

Output :



```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\PRETHIKA>cd Documents

C:\Users\PRETHIKA\Documents>javac MergeIntervals.java

C:\Users\PRETHIKA\Documents>java MergeIntervals
Merged Intervals 1: [[1, 4], [6, 8], [9, 10]]
Merged Intervals 2: [[1, 6], [7, 8]]
```

Time Complexity : O(n log n)

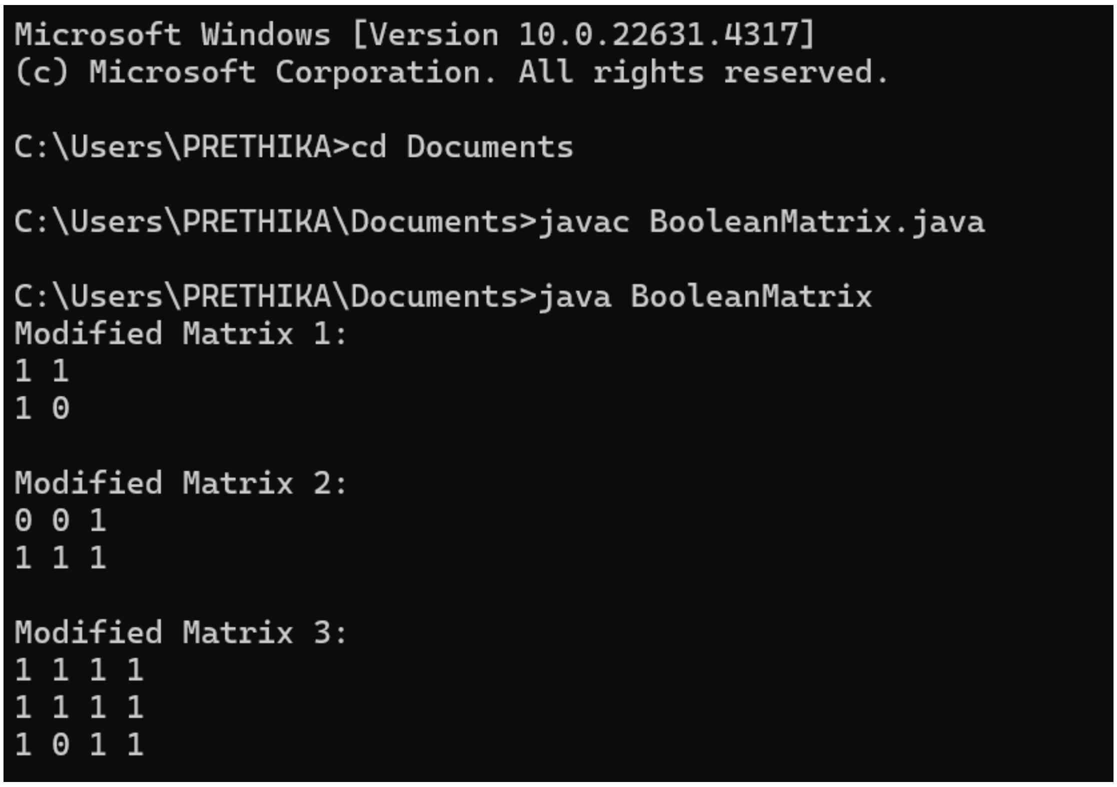
9. A Boolean Matrix

```
public class BooleanMatrix {
```

```
public static void modifyMatrix(int[][] mat) {
    int M = mat.length;
    int N = mat[0].length;
    boolean[] rowFlag = new boolean[M];
    boolean[] colFlag = new boolean[N];
    for (int i = 0; i < M; i++) {
        for (int j = 0; j < N; j++) {
            if (mat[i][j] == 1) {
                rowFlag[i] = true;
                colFlag[j] = true;
            }
        }
    }
    for (int i = 0; i < M; i++) {
        for (int j = 0; j < N; j++) {
            if (rowFlag[i] || colFlag[j]) {
                mat[i][j] = 1;
            }
        }
    }
}
public static void printMatrix(int[][] mat) {
    for (int i = 0; i < mat.length; i++) {
        for (int j = 0; j < mat[i].length; j++) {
            System.out.print(mat[i][j] + " ");
        }
        System.out.println();
    }
}
public static void main(String[] args) {
    int[][] mat1 = {{1, 0}, {0, 0}};
    int[][] mat2 = {{0, 0, 0}, {0, 0, 1}};
```

```
        int[][] mat3 = {{1, 0, 0, 1}, {0, 0, 1, 0}, {0, 0, 0, 0}};
        System.out.println("Modified Matrix 1:");
        modifyMatrix(mat1);
        printMatrix(mat1);
        System.out.println("\nModified Matrix 2:");
        modifyMatrix(mat2);
        printMatrix(mat2);
        System.out.println("\nModified Matrix 3:");
        modifyMatrix(mat3);
        printMatrix(mat3);
    }
}
```

Output :



```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\PRETHIKA>cd Documents

C:\Users\PRETHIKA\Documents>javac BooleanMatrix.java

C:\Users\PRETHIKA\Documents>java BooleanMatrix
Modified Matrix 1:
1 1
1 0

Modified Matrix 2:
0 0 1
1 1 1

Modified Matrix 3:
1 1 1 1
1 1 1 1
1 0 1 1
```

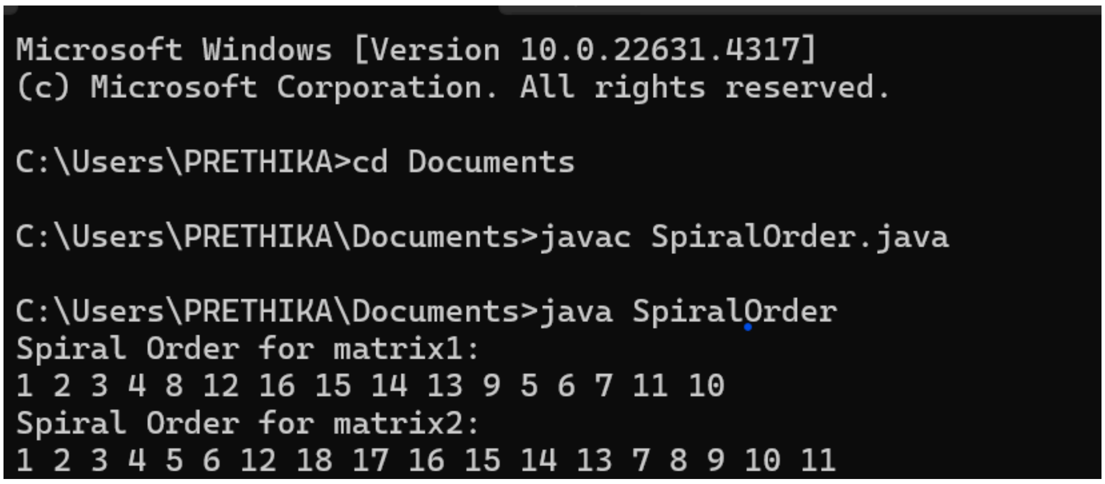
Time Complexity : O(M * N)

10. Print a given matrix in spiral form

```
public class SpiralOrder {
```

```
public static void printSpiral(int[][] matrix) {
    if (matrix == null || matrix.length == 0 || matrix[0].length == 0) {
        return;
    }
    int top = 0, bottom = matrix.length - 1;
    int left = 0, right = matrix[0].length - 1;
    while (top <= bottom && left <= right) {
        for (int i = left; i <= right; i++) {
            System.out.print(matrix[top][i] + " ");
        }
        top++;
        for (int i = top; i <= bottom; i++) {
            System.out.print(matrix[i][right] + " ");
        }
        right--;
        if (top <= bottom) {
            for (int i = right; i >= left; i--) {
                System.out.print(matrix[bottom][i] + " ");
            }
            bottom--;
        }
        if (left <= right) {
            for (int i = bottom; i >= top; i--) {
                System.out.print(matrix[i][left] + " ");
            }
            left++;
        }
    }
}
public static void main(String[] args) {
    int[][] matrix1 = {
        {1, 2, 3, 4},
        {5, 6, 7, 8},
        {9, 10, 11, 12},
        {13, 14, 15, 16}
    };
    int[][] matrix2 = {
        {1, 2, 3, 4, 5, 6},
        {7, 8, 9, 10, 11, 12},
        {13, 14, 15, 16, 17, 18}
    };
    System.out.println("Spiral Order for matrix1:");
    printSpiral(matrix1);
    System.out.println("\nSpiral Order for matrix2:");
    printSpiral(matrix2);
}
```

Output :



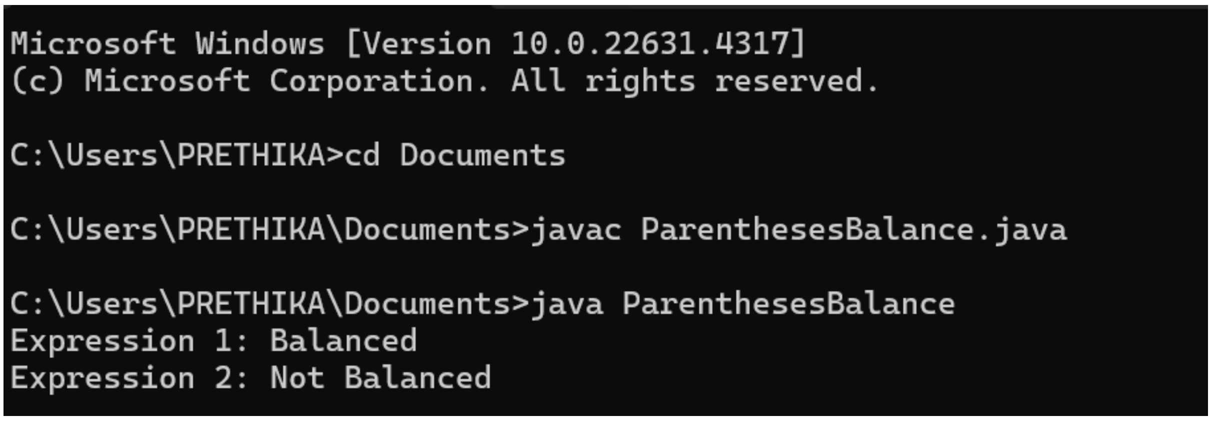
```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\PRETHIKA>cd Documents

C:\Users\PRETHIKA\Documents>javac SpiralOrder.java

C:\Users\PRETHIKA\Documents>java SpiralOrder
Spiral Order for matrix1:
1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10
Spiral Order for matrix2:
1 2 3 4 5 6 12 18 17 16 15 14 13 7 8 9 10 11
```

Time Complexity : O(m * n)

11.Check if given Parentheses expression is balanced or not

```
public class ParenthesesBalance {
    public static String checkBalanced(String str) {
        LinkedList<Character> stack = new LinkedList<>();
        for (char ch : str.toCharArray()) {
            if (ch == '(') {
                stack.addFirst(ch);
            }
            else if (ch == ')') {
                if (stack.isEmpty()) {
                    return "Not Balanced";
                }
                stack.removeFirst();
            }
        }
        return stack.isEmpty() ? "Balanced" : "Not Balanced";
    }
    public static void main(String[] args) {
        String expression1 = "((()))()()";
        String expression2 = "())((())";
        System.out.println("Expression 1: " + checkBalanced(expression1));
        System.out.println("Expression 2: " + checkBalanced(expression2));
    }
}
```

Output :



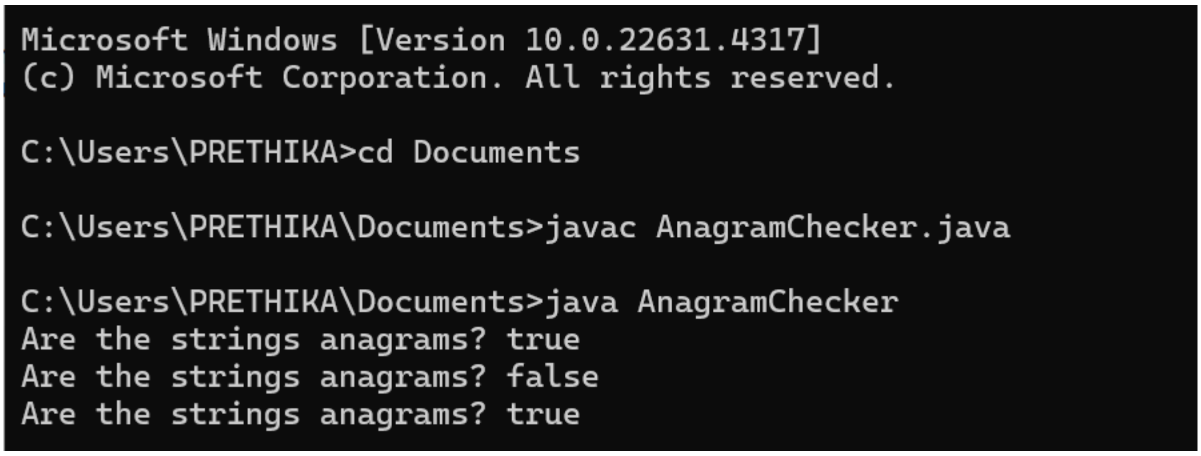
```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\PRETHIKA>cd Documents

C:\Users\PRETHIKA\Documents>javac ParenthesesBalance.java

C:\Users\PRETHIKA\Documents>java ParenthesesBalance
Expression 1: Balanced
Expression 2: Not Balanced
```

Time Complexity : O(n)

12. . Check if two Strings are Anagrams of each other

```
import java.util.Arrays;
public class AnagramChecker {
    public static boolean areAnagrams(String s1, String s2) {
        if (s1.length() != s2.length()) {
            return false;
        }
        char[] arr1 = s1.toCharArray();
        char[] arr2 = s2.toCharArray();
        Arrays.sort(arr1);
        Arrays.sort(arr2);
        return Arrays.equals(arr1, arr2);
    }
    public static void main(String[] args) {
        String s1 = "geeks";
        String s2 = "kseeg";
        System.out.println("Are the strings anagrams? " + areAnagrams(s1, s2));
        s1 = "allergy";
        s2 = "allergic";
        System.out.println("Are the strings anagrams? " + areAnagrams(s1, s2));
        s1 = "g";
        s2 = "g";
        cSystem.out.println("Are the strings anagrams? " + areAnagrams(s1, s2));
    }
}
```

Output :



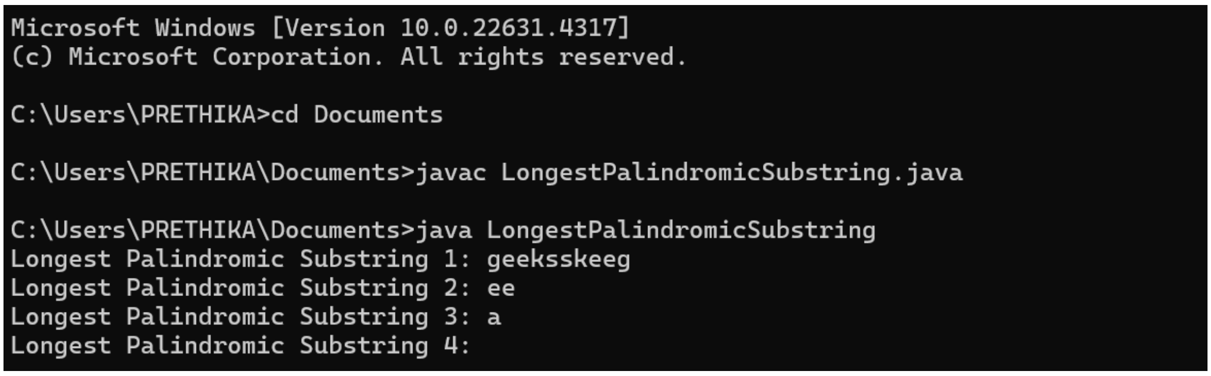
```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\PRETHIKA>cd Documents

C:\Users\PRETHIKA\Documents>javac AnagramChecker.java

C:\Users\PRETHIKA\Documents>java AnagramChecker
Are the strings anagrams? true
Are the strings anagrams? false
Are the strings anagrams? true
```

Time Complexity : O(n log n)

13. Longest Palindromic Substring

```
public class LongestPalindromicSubstring {
    public static String expandAroundCenter(String str, int left, int right) {
        while (left >= 0 && right < str.length() && str.charAt(left) == str.charAt(right)) {
            left--;
            right++;
        }
        return str.substring(left + 1, right);
    }
    public static String longestPalindrome(String str) {
        if (str == null || str.length() == 0) {
            return "";
        }
        String longest = "";
        for (int i = 0; i < str.length(); i++) {
            String oddPalindrome = expandAroundCenter(str, i, i);
            String evenPalindrome = expandAroundCenter(str, i, i + 1);
            if (oddPalindrome.length() > longest.length()) {
                longest = oddPalindrome;
            }
            if (evenPalindrome.length() > longest.length()) {
                longest = evenPalindrome;
            }
        }
        return longest;
    }
    public static void main(String[] args) {
        System.out.println("Longest Palindromic Substring 1: " +
longestPalindrome("forgeeksskeegfor"));
        System.out.println("Longest Palindromic Substring 2: " + longestPalindrome("Geeks"));
        System.out.println("Longest Palindromic Substring 3: " + longestPalindrome("abc"));
        System.out.println("Longest Palindromic Substring 4: " + longestPalindrome(""));
    }
}
```

Output :



```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\PRETHIKA>cd Documents

C:\Users\PRETHIKA\Documents>javac LongestPalindromicSubstring.java

C:\Users\PRETHIKA\Documents>java LongestPalindromicSubstring
Longest Palindromic Substring 1: geeksskeeg
Longest Palindromic Substring 2: ee
Longest Palindromic Substring 3: a
Longest Palindromic Substring 4:
```

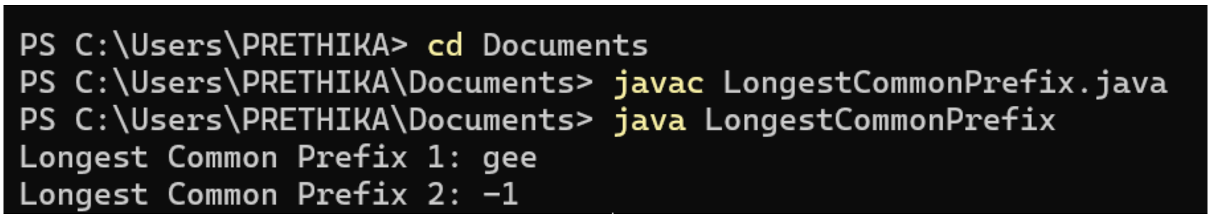
Time Complexity : O(n ^ 2)

14. Longest Common Prefix using Sorting

```
import java.util.Arrays;
public class LongestCommonPrefix {
  public static String longestCommonPrefix(String[] arr) {
    if (arr == null || arr.length == 0) {
      return "-1";
    }
    Arrays.sort(arr);
    String first = arr[0];
    String last = arr[arr.length - 1];
    int i = 0;
    while (i < first.length() && i < last.length() && first.charAt(i) == last.charAt(i)) {
      i++;
    }
    if (i == 0) {
      return "-1";
    }
    return first.substring(0, i);
  }
  public static void main(String[] args) {
    String[] arr1 = {"geeksforgeeks", "geeks", "geek", "geezer"};
    String[] arr2 = {"hello", "world"};
    System.out.println("Longest Common Prefix 1: " + longestCommonPrefix(arr1));
```

```
        System.out.println("Longest Common Prefix 2: " + longestCommonPrefix(arr2));
    }
}
```

Output :



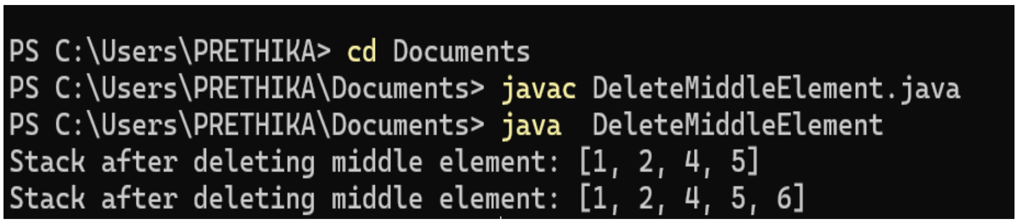
```
PS C:\Users\PRETHIKA> cd Documents
PS C:\Users\PRETHIKA\Documents> javac LongestCommonPrefix.java
PS C:\Users\PRETHIKA\Documents> java LongestCommonPrefix
Longest Common Prefix 1: gee
Longest Common Prefix 2: -1
```

Time Complexity : O(n log n)

15. Delete middle element of a stack

```
import java.util.Stack;
public class DeleteMiddleElement {
    public static void deleteMiddle(Stack<Integer> stack, int currentIndex, int size) {
        if (currentIndex == size / 2) {
            stack.pop();
            return;
        }
        int temp = stack.pop();
        deleteMiddle(stack, currentIndex + 1, size);
        stack.push(temp);
    }
    public static void deleteMiddleElement(Stack<Integer> stack) {
        int size = stack.size();
        if (size == 0) {
            return;
        }
        deleteMiddle(stack, 0, size);
    }
    public static void main(String[] args) {
        Stack<Integer> stack1 = new Stack<>();
        stack1.push(1);
        stack1.push(2);
        stack1.push(3);
        stack1.push(4);
        stack1.push(5);
        deleteMiddleElement(stack1);
        System.out.println("Stack after deleting middle element: " + stack1);
        Stack<Integer> stack2 = new Stack<>();
        stack2.push(1);
        stack2.push(2);
        stack2.push(3);
        stack2.push(4);
        stack2.push(5);
        stack2.push(6);
        deleteMiddleElement(stack2);
        System.out.println("Stack after deleting middle element: " + stack2);
    }
}
```

Output :



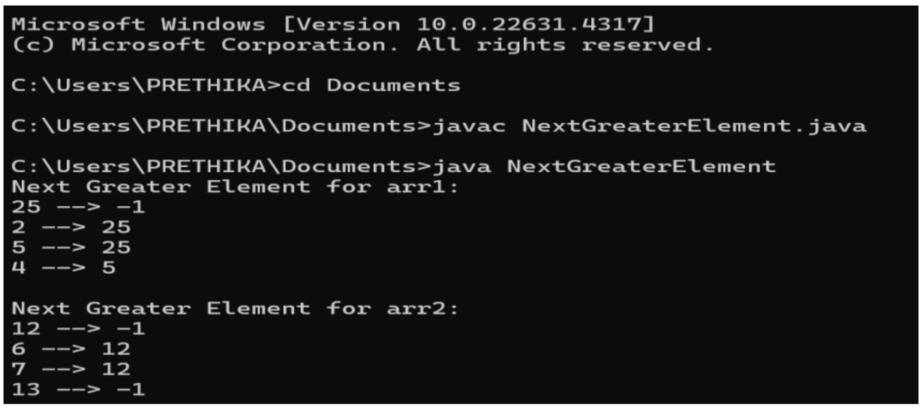
```
PS C:\Users\PRETHIKA> cd Documents
PS C:\Users\PRETHIKA\Documents> javac DeleteMiddleElement.java
PS C:\Users\PRETHIKA\Documents> java  DeleteMiddleElement
Stack after deleting middle element: [1, 2, 4, 5]
Stack after deleting middle element: [1, 2, 4, 5, 6]
```

Time Complexity : O(n)

16. Next Greater Element (NGE) for every element in given Array

```
import java.util.Stack;
public class NextGreaterElement {
    public static void printNextGreaterElement(int[] arr) {
        int n = arr.length;
        Stack<Integer> stack = new Stack<>();
        for (int i = n - 1; i >= 0; i--) {
            while (!stack.isEmpty() && stack.peek() <= arr[i]) {
                stack.pop();
            }
            if (stack.isEmpty()) {
                System.out.println(arr[i] + " --> -1");
            } else {
                System.out.println(arr[i] + " --> " + stack.peek());
            }
            stack.push(arr[i]);
        }
    }
    public static void main(String[] args) {
        int[] arr1 = {4, 5, 2, 25};
        int[] arr2 = {13, 7, 6, 12};
        System.out.println("Next Greater Element for arr1:");
        printNextGreaterElement(arr1);
        System.out.println("\nNext Greater Element for arr2:");
        printNextGreaterElement(arr2);
    }
}
```

Output :



```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\PRETHIKA>cd Documents

C:\Users\PRETHIKA\Documents>javac NextGreaterElement.java

C:\Users\PRETHIKA\Documents>java NextGreaterElement
Next Greater Element for arr1:
25 --> -1
2 --> 25
5 --> 25
4 --> 5

Next Greater Element for arr2:
12 --> -1
6 --> 12
7 --> 12
13 --> -1
```

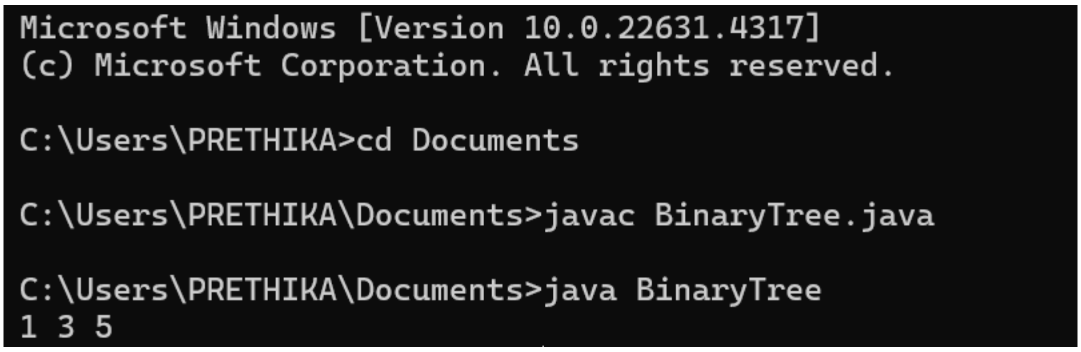
Time Complexity : O(n)

17. Print Right View of a Binary Tree

```
import java.util.LinkedList;
```

```
class Node {
    int data;
    Node left, right;
    Node(int item) {
        data = item;
        left = right = null;
    }
}
class BinaryTree {
    Node root;
    void printRightView(Node node) {
        if (node == null)
            return;
        LinkedList<Node> q = new LinkedList<>();
        q.add(node);
        while (!q.isEmpty()) {
            int n = q.size();
            for (int i = 1; i <= n; i++) {
                Node temp = q.poll();
                if (i == n)
                    System.out.print(temp.data + " ");
                if (temp.left != null)
                    q.add(temp.left);
                if (temp.right != null)
                    q.add(temp.right);
            }
        }
    }
    public static void main(String[] args) {
        BinaryTree tree = new BinaryTree();
        tree.root = new Node(1);
        tree.root.left = new Node(2);
        tree.root.right = new Node(3);
        tree.root.left.left = new Node(4);
        tree.root.left.right = new Node(5);
        tree.printRightView(tree.root);
    }
}
```

Output :



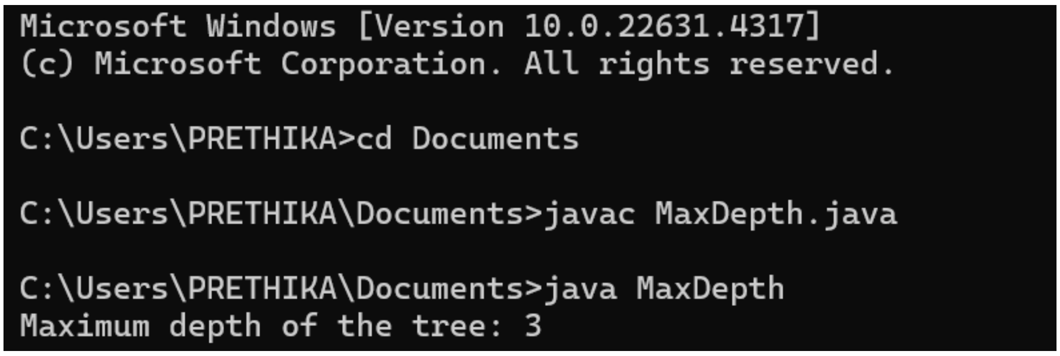
```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\PRETHIKA>cd Documents

C:\Users\PRETHIKA\Documents>javac BinaryTree.java

C:\Users\PRETHIKA\Documents>java BinaryTree
1 3 5
```

Time Complexity : O(N)

18. Maximum Depth or Height of Binary Tree

```
class TreeNode {
    int val;
    TreeNode left;
    TreeNode right;

    TreeNode(int val) {
        this.val = val;
    }
}
public class MaxDepth {
    public static int maxDepth(TreeNode root) {
        if (root == null) {
            return 0;
        }
        int leftDepth = maxDepth(root.left);
        int rightDepth = maxDepth(root.right);
        return Math.max(leftDepth, rightDepth) + 1;
    }
    public static void main(String[] args) {
        TreeNode root = new TreeNode(1);
        root.left = new TreeNode(2);
        root.right = new TreeNode(3);
        root.left.left = new TreeNode(4);
        root.left.right = new TreeNode(5);
        root.right.left = new TreeNode(6);
        root.right.right = new TreeNode(7);
        int maxDepth = maxDepth(root);
        System.out.println("Maximum depth of the tree: " + maxDepth);
    }
}
```

Output :



```
Microsoft Windows [Version 10.0.22631.4317]
(c) Microsoft Corporation. All rights reserved.

C:\Users\PRETHIKA>cd Documents

C:\Users\PRETHIKA\Documents>javac MaxDepth.java

C:\Users\PRETHIKA\Documents>java MaxDepth
Maximum depth of the tree: 3
```

Time Complexity : O(N)