

1. Maximum Subarray Sum – Kadane's Algorithm:

Given an array arr[], the task is to find the subarray that has the maximum sum and return its sum.

Code:

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

```
import java.util.Arrays;
```

```
class maximumsum {
```

```
    static int[] maxSubarraySum(int[] arr) {
```

```
        int maxSum = arr[0];
```

```
        int maxEnding = arr[0];
```

```
        int start = 0;
```

```
        int end = 0;
```

```
        int tempStart = 0;
```

```
        for (int i = 1; i < arr.length; i++) {
```

```
            if (arr[i] > maxEnding + arr[i]) {
```

```
                maxEnding = arr[i];
```

```
                tempStart = i;
```

```
            } else {
```

```
                maxEnding += arr[i];
```

```
            }
```

```
        if (maxEnding > maxSum) {
```

```
            maxSum = maxEnding;
```

```
            start = tempStart;
```

```
            end = i;
```

```
    }  
}
```

```
int[] maxSubarray = Arrays.copyOfRange(arr, start, end + 1);
```

```
System.out.println("The maximum subarray is: " + Arrays.toString(maxSubarray));
```

```
return new int[] { maxSum };
```

```
}
```

```
public static void main(String[] args) {
```

```
    Scanner scanner = new Scanner(System.in);
```

```
    System.out.print("Enter the size of the array: ");
```

```
    int n = scanner.nextInt();
```

```
    int[] arr = new int[n];
```

```
    System.out.println("Enter the elements of the array:");
```

```
    for (int i = 0; i < n; i++) {
```

```
        arr[i] = scanner.nextInt();
```

```
    }
```

```
    System.out.println("The input array is: " + Arrays.toString(arr));
```

```
    int[] result = maxSubarraySum(arr);
```

```
    System.out.println("The maximum subarray sum is: " + result[0]);
```

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

```
}
```

```
}
```

OUTPUT:

```
C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>javac maximumsum.java

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>java maximumsum
Enter the size of the array: 7
Enter the elements of the array:
2
3
-8
7
-1
2
3
The maximum subarray sum is: 11

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>javac maximumsum.java

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>java maximumsum
Enter the size of the array: 7
Enter the elements of the array:
2
3
-8
7
-1
2
3
The input array is: [2, 3, -8, 7, -1, 2, 3]
The maximum subarray is: [7, -1, 2, 3]
The maximum subarray sum is: 11

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>
```

Time Complexity: $O(n)$

2. Maximum Product Subarray

Given an integer array, the task is to find the maximum product of any subarray.

Code:

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

```
class maximumproduct {
```

```
static int[] maxProduct(int[] nums) {  
    if (nums.length == 0) return new int[]{0};  
  
    int maxProd = nums[0];  
    int minProd = nums[0];  
    int result = nums[0];  
  
    int start = 0, end = 0, tempStart = 0;  
  
    for (int i = 1; i < nums.length; i++) {  
        if (nums[i] < 0) {  
            int temp = maxProd;  
            maxProd = minProd;  
            minProd = temp;  
        }  
  
        maxProd = Math.max(nums[i], maxProd * nums[i]);  
        minProd = Math.min(nums[i], minProd * nums[i]);  
  
        if (maxProd > result) {  
            result = maxProd;  
            start = tempStart;  
            end = i;  
        }  
  
        if (nums[i] > maxProd) {  
            tempStart = i;  
        }  
    }  
}
```

```

int[] maxSubarray = new int[end - start + 1];
for (int i = start; i <= end; i++) {
    maxSubarray[i - start] = nums[i];
}

System.out.println("The input array is: ");
for (int num : nums) {
    System.out.print(num + " ");
}
System.out.println();

System.out.println("The maximum product subarray is: ");
for (int i = start; i <= end; i++) {
    System.out.print(nums[i] + " ");
}
System.out.println();

return new int[] { result };
}

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

    System.out.print("Enter the size 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 = maxProduct(nums);

    System.out.println("The maximum product of a subarray is: " + result[0]);

    scanner.close();
}
}

```

Output:

```

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>javac maximumproduct.java

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>java maximumproduct
Enter the size of the array: 6
Enter the elements of the array:
-2
6
-3
-10
0
2
The input array is:
-2 6 -3 -10 0 2
The maximum product subarray is:
-2 6 -3 -10
The maximum product of a subarray is: 180

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>

```

Time Complexity: $O(n)$

3. Search in a sorted and rotated Array Given a sorted and rotated array arr[] of n distinct elements, the task is to find the index of given key in the array. If the key is not present in the array, return -1.

Code:

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

```
class searchinsorted {  
    public int search(int[] nums, int target) {  
        int low = 0, high = nums.length - 1;  
  
        while (low <= high) {  
            int mid = (low + high) / 2;  
  
            if (nums[mid] == target) {  
                return mid;  
            }  
  
            if (nums[low] <= nums[mid]) {  
                if (nums[low] <= target && target < nums[mid]) {  
                    high = mid - 1;  
                } else {  
                    low = mid + 1;  
                }  
            } else {  
                if (nums[mid] < target && target <= nums[high]) {  
                    low = mid + 1;  
                } else {  
                    high = mid - 1;  
                }  
            }  
        }  
    }  
}
```

```

    }

    return -1;
}

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

    System.out.print("Enter the size 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();
    }

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

    searchinsorted solution = new searchinsorted();
    int result = solution.search(nums, target);
    System.out.println(result);
    scanner.close();
}
}

```


Output:

```
C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>javac searchinsorted.java

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>java searchinsorted
Enter the size of the array: 6
Enter the elements of the array:
4
5
6
0
1
2
Enter the target value to search for: 0
3

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>
```

Time complexity : $O(\log n)$

4. Container with Most Water

Given n non-negative integers a_1, a_2, \dots, a_n , where each represents a point at coordinate (i, a_i) . ' n ' vertical lines are drawn such that the two endpoints of line i is at (i, a_i) and $(i, 0)$. Find two lines, which together with x-axis forms a container, such that the container contains the most water.

The program should return an integer which corresponds to the maximum area of water that can be contained (maximum area instead of maximum volume sounds weird but this is the 2D plane we are working with for simplicity).

Note: You may not slant the container.

Code:

```
import java.util.Scanner;

class ContainerArea {
    public int maxArea(int[] height) {
```

```

int left = 0;

int 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) {

    Scanner scanner = new Scanner(System.in);

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

    int[] height = new int[n];

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

```

```

ContainerArea containerArea = new ContainerArea();

int result = containerArea.maxArea(height);

System.out.println("The maximum area is: " + result);

scanner.close();
}
}

```

Output:

```

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>java Containerarea
Error: Could not find or load main class Containerarea
Caused by: java.lang.NoClassDefFoundError: Containerarea (wrong name: ContainerArea)

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>java ContainerArea
Enter the size of the array: 4
Enter the elements of the array:
1
5
4
3
The maximum area is: 6

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>

```

Time Complexity: $O(n)$

5. Find the Factorial of a large number

Input: 100

Output: 9332621544394415268169923885626670049071596826438162146859296389521
75999932299156089414639761565182862536979208272237582511852109168640000000
0000000000000000

Input: 50

Output: 3041409320171337804361260816606476884437764156896051200000000000

Code:

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

```
class Factorial {
```

```
    static void factorial(int n)
```

```
    {
```

```
        int res[] = new int[500];
```

```
        res[0] = 1;
```

```
        int res_size = 1;
```

```
        for (int x = 2; x <= n; x++)
```

```
            res_size = multiply(x, res, res_size);
```

```
        System.out.println("Factorial of given number is ");
```

```
        for (int i = res_size - 1; i >= 0; i--)
```

```
            System.out.print(res[i]);
```

```
    }
```

```
    static int multiply(int x, int res[], int res_size)
```

```
    {
```

```
        int carry = 0;
```

```
        for (int i = 0; i < res_size; i++) {
```

```
            int prod = res[i] * x + carry;
```

```
            res[i] = prod % 10;
```

```
            carry = prod / 10;
```

```
        }
```

```

while (carry != 0) {
    res[res_size] = carry % 10;
    carry = carry / 10;
    res_size++;
}
return res_size;
}

```

```

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

    System.out.print("Enter a number to find its factorial: ");
    int num = scanner.nextInt();

    factorial(num);

    scanner.close();
}
}

```

Output:

```

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>javac Factorial.java
C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>java Factorial
Enter a number to find its factorial: 100
Factorial of given number is
9332621544394415268169923885626670049071596826438162146859296389521759999322991560894146397615651828625369792082722375825118521091686400000000000000
000000000
C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>

```

Time Complexity: $O(n \log n)$

6. Trapping Rainwater Problem states that given an array of n non-negative integers arr[] representing an elevation map where the width of each bar is 1, compute how much water it can trap after rain.

Code:

```
import java.util.*;

class Trapping {

    public int trap(int[] height) {

        int left = 0;

        int right = height.length - 1;

        int leftMax = height[left];

        int rightMax = height[right];

        int water = 0;

        while (left < right) {

            if (leftMax < rightMax) {

                left++;

                leftMax = Math.max(leftMax, height[left]);

                water += leftMax - height[left];

            } else {

                right--;

                rightMax = Math.max(rightMax, height[right]);

                water += rightMax - height[right];

            }

        }

        return water;

    }

    public static void main(String[] args){

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the size of the array:");
```

```

int n = scanner.nextInt();

int[] height = new int[n];

System.out.println("Enter the array elements:");

for (int i = 0; i < n; i++) {
    height[i] = scanner.nextInt();
}

Trapping trapping = new Trapping();

int trappedWater = trapping.trap(height);

System.out.println();

System.out.println(trappedWater);

scanner.close();
}
}

```

OUTPUT:

```

(actual and formal argument lists differ in length)
1 error

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>javac Trapping.java

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>java Trapping
Enter the size of the array:7
Enter the array elements:
3
0
1
0
4
0
2

10

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>

```

Time Complexity : $O(n)$

7. Chocolate Distribution Problem Given an array arr[] of n integers where arr[i] represents the number of chocolates in ith packet. Each packet can have a variable number of chocolates. There are m students, the task is to distribute chocolate packets such that: Each student gets exactly one packet. The difference between the maximum and minimum number of chocolates in the packets given to the students is minimized.

CODE:

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

class Chocolate {

    static int findMinDiff(int[] arr, int m) {
        int n = arr.length;

        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];

            if (diff < minDiff)
                minDiff = diff;
        }
        return minDiff;
    }

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



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

int[] arr = new int[n];
System.out.println("Enter the number of chocolates in each packet:");
for (int i = 0; i < n; i++) {
    arr[i] = scanner.nextInt();
}

System.out.print("Enter the number of students: ");
int m = scanner.nextInt();
scanner.close();

if (m > n) {
    System.out.println("Number of students cannot be greater than number of packets.");
} else {
    System.out.println("Minimum difference is: " + findMinDiff(arr, m));
}
}
```

Output:

```

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>javac Chocolate.java

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>java Chocolate
Enter the number of packets: 7
Enter the number of chocolates in each packet:
7
3
2
4
9
12
56
Enter the number of students: 3
Minimum difference is: 2

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>

```

Time complexity: $O(n \log n)$

8. Merge Overlapping Intervals

Given an array of time intervals where $arr[i] = [start_i, end_i]$, the task is to merge all the overlapping intervals into one and output the result which should have only mutually exclusive intervals.

Code:

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

```
class Merge {
```

```
    public int[][] merge(int[][] intervals) {
```

```
        Arrays.sort(intervals, (a, b) -> Integer.compare(a[0], b[0]));
```

```
        List<int[]> merged = new ArrayList<>();
```

```
        int[] prev = intervals[0];
```

```
        for (int i = 1; i < intervals.length; i++) {
```

```
            int[] interval = intervals[i];
```

```
            if (interval[0] <= prev[1]) {
```

```
                prev[1] = Math.max(prev[1], interval[1]);
```

```

    } else {
        merged.add(prev);
        prev = interval;
    }
}

merged.add(prev);

return merged.toArray(new int[merged.size()][]);
}

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

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

    int[][] intervals = new int[n][2];
    System.out.println("Enter the intervals (start and end):");

    for (int i = 0; i < n; i++) {
        System.out.print("Interval " + (i + 1) + ": ");
        intervals[i][0] = scanner.nextInt();
        intervals[i][1] = scanner.nextInt();
    }

    Merge mergeObj = new Merge();
    int[][] mergedIntervals = mergeObj.merge(intervals);

```

```

        System.out.println("Merged intervals:");
        for (int[] interval : mergedIntervals) {
            System.out.println(Arrays.toString(interval));
            scanner.close();
        }
    }
}

```

OUTPUT:

```

at Merge.main(Merge.java:36)

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>java Merge
Enter the number of intervals: 4
Enter the intervals (start and end):
Interval 1: 1
3
Interval 2: 2
4
Interval 3: 6
8
Interval 4: 9
10
Merged intervals:
[1, 4]
[6, 8]
[9, 10]

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>

```

Time Complexity: $O(n)$

9. A Boolean Matrix Question

Given a boolean matrix `mat[M][N]` of size $M \times N$, modify it such that if a matrix cell `mat[i][j]` is 1 (or true) then make all the cells of i th row and j th column as 1.

Code:

```

import java.util.Scanner;

class Matrix {
    public static void modifyMatrix(int mat[][], int R, int C) {

```

```
int row[] = new int[R];
```

```
int col[] = new int[C];
```

```
int i, j;
```

```
for (i = 0; i < R; i++)
```

```
    row[i] = 0;
```

```
for (i = 0; i < C; i++)
```

```
    col[i] = 0;
```

```
for (i = 0; i < R; i++) {
```

```
    for (j = 0; j < C; j++) {
```

```
        if (mat[i][j] == 1) {
```

```
            row[i] = 1;
```

```
            col[j] = 1;
```

```
        }
```

```
    }
```

```
}
```

```
for (i = 0; i < R; i++) {
```

```
    for (j = 0; j < C; j++) {
```

```
        if (row[i] == 1 || col[j] == 1)
```

```
            mat[i][j] = 1;
```

```
    }
```

```
}
```

```
}
```

```
public static void printMatrix(int mat[][], int R, int C) {
```

```
    for (int i = 0; i < R; i++) {
```

```
        for (int j = 0; j < C; j++)
```

```

        System.out.print(mat[i][j] + " ");
        System.out.println();
    }
}

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

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

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

    int[][] mat = new int[R][C];
    System.out.println("Enter the elements of the matrix (0 or 1):");
    for (int i = 0; i < R; i++) {
        for (int j = 0; j < C; j++) {
            mat[i][j] = scanner.nextInt();
        }
    }

    System.out.println("Matrix Initially:");
    printMatrix(mat, R, C);

    modifyMatrix(mat, R, C);

    System.out.println("Matrix after modification:");
    printMatrix(mat, R, C);
}

```

```
}  
}
```

Output:

```
C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>java Matrix  
Enter the number of rows: 3  
Enter the number of columns: 3  
Enter the elements of the matrix (0 or 1):  
0  
0  
0  
0  
0  
0  
0  
1  
0  
0  
Matrix Initially:  
0 0 0  
0 0 0  
1 0 0  
Matrix after modification:  
1 0 0  
1 0 0  
1 1 1  
  
C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>
```

Time Complexity: $O(r*c)$

10. Print a given matrix in spiral form

Given an $m \times n$ matrix, the task is to print all elements of the matrix in spiral form.

Code:

```
import java.util.ArrayList;  
  
import java.util.List;  
  
import java.util.Scanner;  
  
class Spiral {  
    public List<Integer> spiralOrder(int[][] matrix) {  
        int rows = matrix.length;  
        int cols = matrix[0].length;
```

```

int x = 0;

int y = 0;

int dx = 1;

int dy = 0;

List<Integer> res = new ArrayList<>();

for (int i = 0; i < rows * cols; i++) {

    res.add(matrix[y][x]);

    matrix[y][x] = -101;

    if (!(0 <= x + dx && x + dx < cols && 0 <= y + dy && y + dy < rows) || matrix[y + dy][x +
dx] == -101) {

        int temp = dx;

        dx = -dy;

        dy = temp;

    }

    x += dx;

    y += dy;

}

return res;

}

public static void main(String[] args) {

    Scanner scanner = new Scanner(System.in);

    System.out.print("Enter the number of rows: ");

    int rows = scanner.nextInt();

```



```

System.out.print("Enter the number of columns: ");

int cols = scanner.nextInt();

int[][] matrix = new int[rows][cols];

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

for (int i = 0; i < rows; i++) {
    for (int j = 0; j < cols; j++) {
        matrix[i][j] = scanner.nextInt();
    }
}

Spiral spiral = new Spiral();

List<Integer> result = spiral.spiralOrder(matrix);

System.out.println("Spiral order of matrix:");

for (int num : result) {
    System.out.print(num + " ");
}

}

}

```

Output:

```

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>javac Spiral.java

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>java Spiral
Enter the number of rows: 4
Enter the number of columns: 4
Enter the elements of the matrix:
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
Spiral order of matrix:
1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10
C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>

```

Time Complexity: $O(\text{rows} * \text{columns})$

13. Check if given Parentheses expression is balanced or not

Given a string str of length N, consisting of „(„, „)„, „{„, and „}„ only, the task is to check whether it is balanced or not.

Code:

```

import java.util.Stack;

import java.util.Scanner;

class Boolean {

    public boolean isValid(String s) {

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

        for (char c : s.toCharArray()) {

            if (c == '(' || c == '{' || c == '[') {

                stack.push(c);
            }
        }
    }
}

```

```

    } else {
        if (stack.isEmpty()) return false;
        char top = stack.pop();
        if ((c == '(' && top != ')') ||
            (c == '[' && top != ']') ||
            (c == '{' && top != '}')) {
            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, braces, and brackets: ");
    String s = scanner.nextLine();

    Boolean booleanChecker = new Boolean();
    boolean result = booleanChecker.isValid(s);

    if (result) {
        System.out.println("The string has valid parentheses.");
    } else {
        System.out.println("The string has invalid parentheses.");
    }
}
}

```

OUTPUT:

```
C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>javac Boolean.java

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>java Boolean
Enter a string of parentheses, braces, and brackets: (((())))()()
Balanced
```

Time Complexity : $O(n)$ n is the length of the string

14. Check if two Strings are Anagrams of each other

Given two strings s_1 and s_2 consisting of lowercase characters, the task is to check whether the two given strings are anagrams of each other or not. An anagram of a string is another string that contains the same characters, only the order of characters can be different.

Code:

```
import java.util.Arrays;

import java.util.Scanner;

class Anagram {

    public boolean isAnagram(String s, String t) {

        char[] sChars = s.toCharArray();

        char[] tChars = t.toCharArray();

        Arrays.sort(sChars);

        Arrays.sort(tChars);

        return Arrays.equals(sChars, tChars);

    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter the first string: ");

        String s = scanner.nextLine();
```

```

System.out.print("Enter the second string: ");

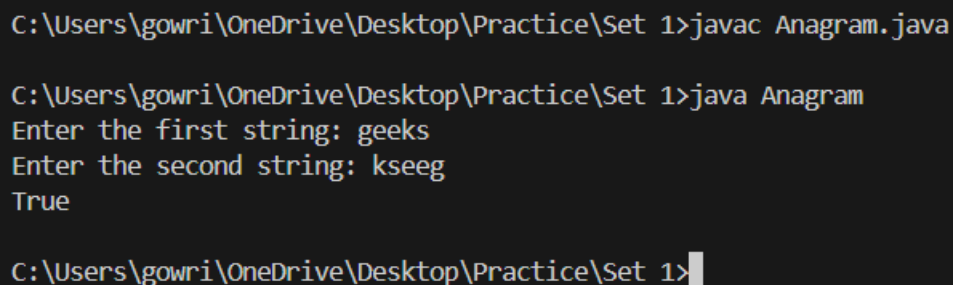
String t = scanner.nextLine();

Anagram anagramChecker = new Anagram();
boolean result = anagramChecker.isAnagram(s, t);

if (result) {
    System.out.println("The strings are anagrams.");
} else {
    System.out.println("The strings are not anagrams.");
}
}
}

```

Output:



```

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>javac Anagram.java

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>java Anagram
Enter the first string: geeks
Enter the second string: kseeg
True

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>

```

Time Complexity : $O(n \log n)$

15. Longest Palindromic Substring

Given a string str, the task is to find the longest substring which is a palindrome. If there are multiple answers, then return the first appearing substring.

Code:

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

```
public class Palindrome {
```

```
    static String longestPalSubstr(String s) {
```

```
        int n = s.length();
```

```
        if (n == 0) return "";
```

```
        int start = 0, maxLen = 1;
```

```
        for (int i = 0; i < n; i++) {
```

```
            for (int j = 0; j <= 1; j++) {
```

```
                int low = i;
```

```
                int hi = i + j;
```

```
                while (low >= 0 && hi < n && s.charAt(low) == s.charAt(hi)) {
```

```
                    int currLen = hi - low + 1;
```

```
                    if (currLen > maxLen) {
```

```
                        start = low;
```

```
                        maxLen = currLen;
```

```
                    }
```

```
                    low--;
```

```
                    hi++;
```

```
                }
```

```
            }
```

```
        }
```

```
        return s.substring(start, start + maxLen);
```

```

    }

    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("Longest Palindrome Substring: " + longestPalSubstr(s));

    }
}

```

Output:

```

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>javac Palindrome.java

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>java Palindrome
Enter a string: forgeeksskeegfor
Longest Palindrome Substring: geeksskeeg

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>java Palindrome
Enter a string: geeks
Longest Palindrome Substring: ee

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>

```

Time Complexity: $O(n^2)$

16. Longest Common Prefix using Sorting

Given an array of strings `arr[]`. The task is to return the longest common prefix among each and every strings present in the array. If there's no prefix common in all the strings, return "-1".

Code:

```

import java.util.Arrays;

import java.util.Scanner;

public class CommonPrefix {

```

```

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 minLength = Math.min(first.length(), last.length());

    int i = 0;
    while (i < minLength && first.charAt(i) == last.charAt(i)) {
        i++;
    }

    if (i == 0)
        return "-1";

    return first.substring(0, i);
}

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

    System.out.print("Enter the number of strings: ");
    int n = scanner.nextInt();
    scanner.nextLine(); // Consume the newline character
}

```



```

String[] arr = new String[n];

System.out.println("Enter the strings:");

for (int i = 0; i < n; i++) {
    arr[i] = scanner.nextLine();
}

System.out.println("The longest common prefix is: "
    + longestCommonPrefix(arr));
}
}

```

Output:

```

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>javac CommonPrefix.java
C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>java CommonPrefix
Enter the number of strings:
4
Enter the strings:
geeksforgeeks
geeks
geek
geezer
The longest common prefix is: gee

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>

```

Time Complexity: $O(n \log n)$

17. Delete middle element of a stack

Given a stack with push(), pop(), and empty() operations, The task is to delete the middle element of it without using any additional data structure.

Code:

```

import java.util.Stack;

import java.util.Vector;

import java.util.Scanner;

public class midstack {

```

```

public static void main(String[] args) {

    Scanner scanner = new Scanner(System.in);

    System.out.print("Enter the number of elements: ");

    int n = scanner.nextInt();

    Stack<Character> st = new Stack<Character>();

    System.out.println("Enter the elements:");

    for (int i = 0; i < n; i++) {

        st.push(scanner.next().charAt(0)); // Read a single character
    }

    Vector<Character> v = new Vector<Character>();

    while (!st.empty()) {

        v.add(st.pop());
    }

    n = v.size();

    if (n % 2 == 0) {

        int target = (n / 2);

        for (int i = 0; i < n; i++) {

            if (i == target) continue;

            st.push(v.get(i));
        }
    } else {

        int target = (int) Math.ceil(n / 2);

        for (int i = 0; i < n; i++) {

            if (i == target) continue;

            st.push(v.get(i));
        }
    }
}

```

```

    }
}

System.out.print("Printing stack after deletion of middle: ");
while (!st.empty()) {
    char p = st.pop();
    System.out.print(p + " ");
}
}
}

```

Output:

```

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>javac midstack.java

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>java midstack
Enter the number of elements: 5
Enter the elements:
1
2
3
4
5
Printing stack after deletion of middle: 1 2 4 5
C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>

```

Time Complexity: $O(n)$

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

Given an array, print the Next Greater Element (NGE) for every element.

Note: The Next greater Element for an element x is the first greater element on the right side of x in the array. Elements for which no greater element exist, consider the next greater element as -1.

Code:

```

import java.util.Scanner;

public class NGE {

```

```

static void printNGE(int arr[], int n) {
    int next, i, j;
    for (i = 0; i < n; i++) {
        next = -1;
        for (j = i + 1; j < n; j++) {
            if (arr[i] < arr[j]) {
                next = arr[j];
                break;
            }
        }
        System.out.println(arr[i] + " -- " + next);
    }
}

```

```

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

    System.out.print("Enter the number of elements: ");
    int n = scanner.nextInt();
    int arr[] = new int[n];

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

    printNGE(arr, n);
}
}

```

Output:

```
C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>javac NGE.java

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>java NGE
Enter the number of elements: 4
Enter the elements:
4
5
2
25
4 -- 5
5 -- 25
2 -- 25
25 -- -1

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>
```

Time Complexity: $O(n)$ **19. Print Right View of a Binary Tree**

Given a Binary Tree, the task is to print the Right view of it. The right view of a Binary Tree is a set of rightmost nodes for every level.

Code:

```
import java.util.ArrayList;
```

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

```
class Node {
```

```
    int data;
```

```
    Node left, right;
```

```
    Node(int x) {
```

```
        data = x;
```

```
        left = right = null;
```

```
    }
```

```
}
```

```
class RightView {
```

```

static void RecursiveRightView(Node root, int level,
    int[] maxLevel, ArrayList<Integer> result) {
    if (root == null) return;

    if (level > maxLevel[0]) {
        result.add(root.data);
        maxLevel[0] = level;
    }

    RecursiveRightView(root.right, level + 1, maxLevel, result);
    RecursiveRightView(root.left, level + 1, maxLevel, result);
}

```

```

static ArrayList<Integer> rightView(Node root) {
    ArrayList<Integer> result = new ArrayList<>();
    int[] maxLevel = new int[] {-1};

    RecursiveRightView(root, 0, maxLevel, result);

    return result;
}

```

```

static void printArray(ArrayList<Integer> arr) {
    for (int val : arr) {
        System.out.print(val + " ");
    }
    System.out.println();
}

```

```

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

```

```

System.out.print("Enter the number of nodes: ");

int n = scanner.nextInt();

Node root = null;


System.out.println("Enter the node values:");

for (int i = 0; i < n; i++) {
    int value = scanner.nextInt();
    root = insert(root, value);
}


ArrayList<Integer> result = rightView(root);
printArray(result);
}


static Node insert(Node root, int value) {
    if (root == null) {
        return new Node(value);
    }
    if (value < root.data) {
        root.left = insert(root.left, value);
    } else {
        root.right = insert(root.right, value);
    }
    return root;
}
}

```

Output:

```
C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>javac RightView.java

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>java RightView
Enter the number of nodes: 5
Enter the node values:
1
2
3
4
5
1 2 3 4 5

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>javac RightView.java

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>java RightView
1 3 5

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>
```

Time Complexity: $O(n)$ **20. Maximum Depth or Height of Binary Tree**

Given a binary tree, the task is to find the maximum depth or height of the tree. The height of the tree is the number of vertices in the tree from the root to the deepest node.

Code:

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

```
class Node {
    int data;
    Node left, right;

    Node(int val) {
        data = val;
        left = null;
        right = null;
    }
}
```



```

class Height {
    static int maxDepth(Node node) {
        if (node == null)
            return 0;

        int lDepth = maxDepth(node.left);
        int rDepth = maxDepth(node.right);

        return Math.max(lDepth, rDepth) + 1;
    }

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

        System.out.print("Enter the number of nodes: ");
        int n = sc.nextInt();

        int[] nodes = new int[n];
        System.out.print("Enter the node values: ");
        for (int i = 0; i < n; i++) {
            nodes[i] = sc.nextInt();
        }

        Node root = new Node(nodes[0]);
        Node current;
        for (int i = 1; i < n; i++) {
            current = root;
            while (true) {
                if (nodes[i] < current.data) {
                    if (current.left == null) {
                        current.left = new Node(nodes[i]);
                    }
                }
            }
        }
    }
}

```

```

        break;
    }
    current = current.left;
} else {
    if (current.right == null) {
        current.right = new Node(nodes[i]);
        break;
    }
    current = current.right;
}
}
}

System.out.println("Maximum depth of the tree: " + maxDepth(root));

sc.close();
}
}

```

Output:

```

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>javac Height.java

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>java Height
Enter the number of nodes: 5
Enter the node values: 12
8
18
5
11
Maximum depth of the tree: 3

C:\Users\gowri\OneDrive\Desktop\Practice\Set 1>

```

Time Complexity: $O(n)$

