

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.

Input: arr[] = {2, 3, -8, 7, -1, 2, 3}

Output: 11 Explanation: The subarray {7, -1, 2, 3} has the largest sum 11.

Input: arr[] = {-2, -4}

Output: -2 Explanation: The subarray {-2} has the largest sum -2. Input: arr[] = {5, 4, 1, 7, 8} Output: 25 Explanation: The subarray {5, 4, 1, 7, 8} has the largest sum 25.

Solution:

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

public class Maximumsubarray{
    public static void main(String[] args) {
        maxsubarray(new int[] {2, 3, -8, 7, -1, 2, 3 });
        maxsubarray(new int[] {5, 4, 1, 7, 8});
    }

    public static void maxsubarray(int[] arr){
        int n = arr.length;
        int sum =0;
        int maxsum = Integer.MIN_VALUE;
        for(int i=0;i<n;i++){
            sum+=arr[i];
            if(arr[i]>sum) sum = arr[i];
            maxsum = Math.max(maxsum,sum);
        }
        System.out.println(Arrays.toString(arr)+":"+maxsum);
    }
}
```

Output :

```
DSA Practice Questions cd "c:\Users\gokul\Documents\Projects\DSA Practice Questions\" ; if ($?) { javac Maximumsubarray.java } ; if ($?) { java M
aximumsubarray }
[2, 3, -8, 7, -1, 2, 3]:11
[5, 4, 1, 7, 8]:25
DSA Practice Questions
```

Time Complexity: $O(n)$

Space Complexity : $O(1)$

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

Input: `arr[] = {-2, 6, -3, -10, 0, 2}`

Output: 180

Explanation: The subarray with maximum product is {6, -3, -10} with product = $6 * (-3) * (-10) = 180$

Input: `arr[] = {-1, -3, -10, 0, 60}`

Output: 60 Explanation: The subarray with maximum product is {60}.

Solution:

```
import java.util.*;

public class Maximumsubarray{

    public static void main(String[] args) {

        maxsubarray(new int[] {-2, 6, -3, -10, 0, 2} );

        maxsubarray(new int[] {-1, -3, -10, 0, 60} );

    }

    public static void maxsubarray(int[] num){

        int n = num.length;

        int maxpro = num[0];

        int currpro = num[0];

        int negpro = num[0];

        for(int i=1;i<n;i++){

            if(num[i]<0){

                int temp = currpro;

                currpro = negpro;

                negpro = temp;

            }

        }

    }

}
```

```

        currpro = Math.max(num[i],num[i]*currpro);

        negpro = Math.min(num[i],num[i]*negpro);

        maxpro = Math.max(maxpro, currpro);

    }

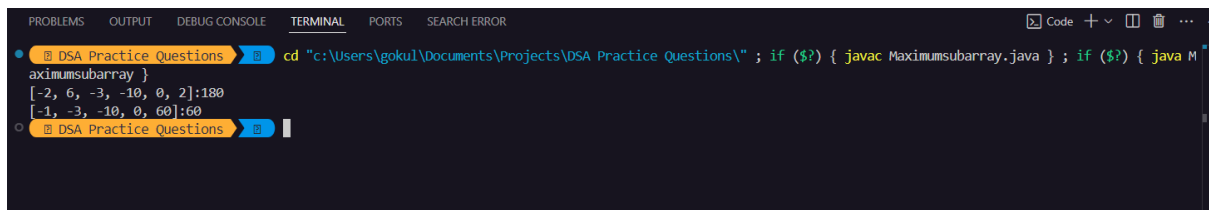
    System.out.println(Arrays.toString(num)+":"+maxpro);

}

}

```

Ouput:



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR
DSA Practice Questions cd "c:\Users\gokul\Documents\Projects\DSA Practice Questions\" ; if ($?) { javac Maximumsubarray.java } ; if ($?) { java M
aximumsubarray }
[-2, 6, -3, -10, 0, 2]:180
[-1, -3, -10, 0, 60]:60
DSA Practice Questions

```

Time Complexity : $O(n)$

Space Complexity : $O(1)$

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.

Input : `arr[] = {4, 5, 6, 7, 0, 1, 2}`, key = 0

Output : 4

Input : `arr[] = { 4, 5, 6, 7, 0, 1, 2 }`, key = 3

Output : -1

Input : `arr[] = {50, 10, 20, 30, 40}`, key = 10

Output : 1

Solution:

```

import java.util.*;

public class Maximumsubarray{

    public static void main(String[] args) {

        search(new int[]{4, 5, 6, 7, 0, 1, 2}, 0);

        search(new int[] { 4, 5, 6, 7, 0, 1, 2 },3);

        search(new int[] {50, 10, 20, 30, 40}, 10 );

    }
}

```

```

public static void search(int[] num ,int key){

    int n = num.length;

    int res = -1;

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

        if(num[i]==key){

            res = i;

            break;

        }

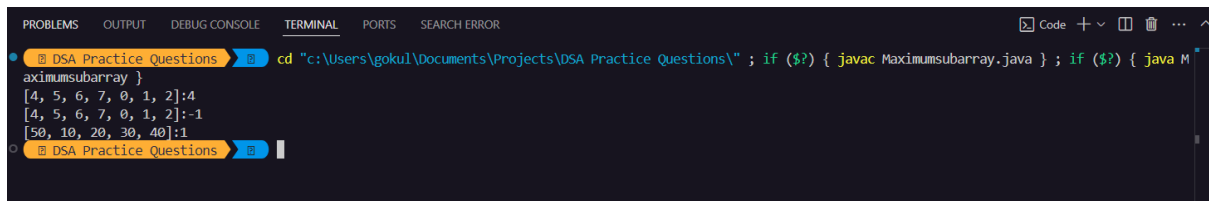
    }

    System.out.println(Arrays.toString(num)+"."+res);

}
}

```

Output:



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR
• DSA Practice Questions cd "c:\Users\gokul\Documents\Projects\DSA Practice Questions\" ; if ($?) { javac Maximumsubarray.java } ; if ($?) { java M
aximumsubarray }
[4, 5, 6, 7, 0, 1, 2]:4
[4, 5, 6, 7, 0, 1, 2]:-1
[50, 10, 20, 30, 40]:1
○ DSA Practice Questions

```

Time Complexity: $O(n)$

Space Complexity: $O(1)$

4. Container with Most Water

Input: arr = [1, 5, 4, 3]

Output: 6 Explanation: 5 and 3 are distance 2 apart. So the size of the base = 2. Height of container = $\min(5, 3) = 3$. So total area = $3 * 2 = 6$

Input: arr = [3, 1, 2, 4, 5]

Output: 12 Explanation: 5 and 3 are distance 4 apart. So the size of the base = 4. Height of container = $\min(5, 3) = 3$. So total area = $4 * 3 = 12$

Solution:

```

import java.util.*;

public class Maximumsubarray{

    public static void main(String[] args) {

        search(new int[]{1, 5, 4, 3});
    }
}

```

```

        search(new int[] { 3, 1, 2, 4, 5});
    }

    public static void search(int[] num){

        int n = num.length;

        int maxarea = Integer.MIN_VALUE;

        int left=0,right=n-1;

        while(left<right){

            int area = (right-left)*Math.min(num[left],num[right]);

            maxarea = Math.max(maxarea,area);

            if(num[left]<num[right]) left++;

            else right--;

        }

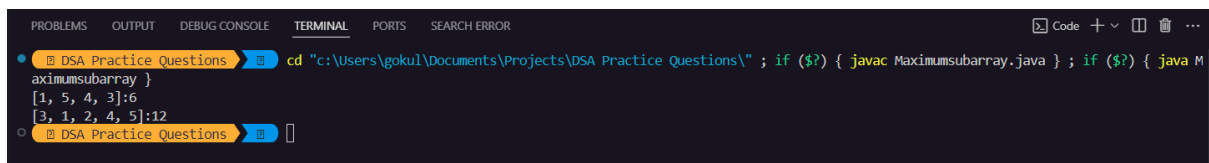
        System.out.println(Arrays.toString(num)+":"+maxarea);

    }

}

```

Output:



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR
DSA Practice Questions cd "c:\Users\gokul\Documents\Projects\DSA Practice Questions\" ; if ($?) { javac Maximumsubarray.java } ; if ($?) { java M
aximumsubarray }
[1, 5, 4, 3]:6
[3, 1, 2, 4, 5]:12
DSA Practice Questions

```

Time Complexity: $O(N)$

Space Complexit : $O(1)$

5. Find the Factorial of a large number

Input: 100

Output:

933262154439441526816992388562667004907159682643816214685929638952175999932299
1560894146397615651828625369792082722375825118521091686400000000000000000000 00

Input: 50

Output: 30414093201713378043612608166064768844377641568960512000000000000

Solution :

```
import java.math.BigInteger;
```

```

public class Maximumsubarray{

    public static void main(String[] args) {

        System.out.println(solution(100));

        System.out.println(solution(10));

    }

    public static BigInteger solution(int num){

        if(num == 1) return BigInteger.ONE;

        else return BigInteger.valueOf(num).multiply(solution(num-1));

    }

}

```

7. Chocolate Distribution Problem

Given an array `arr[]` of `n` integers where `arr[i]` represents the number of chocolates in `i`th 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.

Input: `arr[] = {7, 3, 2, 4, 9, 12, 56}`, `m = 3`

Output: 2 Explanation: If we distribute chocolate packets {3, 2, 4}, we will get the minimum difference, that is 2. |

nput: `arr[] = {7, 3, 2, 4, 9, 12, 56}`, `m = 5`

Output: 7 Explanation: If we distribute chocolate packets {3, 2, 4, 9, 7}, we will get the minimum difference, that is $9 - 2 = 7$.

Solution:

```

import java.util.Arrays;

public class Maximumsubarray{

    public static void main(String[] args) {

        solution(new int[]{7, 3, 2, 4, 9, 12, 56},3);

        solution(new int[]{7, 3, 2, 4, 9, 12, 56},5);

    }

    public static void solution(int[] num , int m){

        int n = num.length;

        int res = Integer.MAX_VALUE;

        Arrays.sort(num);

        for(int i=0;i<n-(m-1);i++){

```

```

        res = Math.min(res, num[i+(m-1)]-num[i]);
    }

    System.out.println(Arrays.toString(num)+"."+res);
}
}

```

Output:

```

c:\Users\gokul\Documents\Projects\DSA Practice Questions\" ; if ($?) { javac Maximumsubarray.java } ; if ($?) { java M
aximumsubarray }
[2, 3, 4, 7, 9, 12, 56]:2
[2, 3, 4, 7, 9, 12, 56]:7

```

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

Space Complexity : $O(1)$

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.

Input: $arr[] = [[1, 3], [2, 4], [6, 8], [9, 10]]$

Output: $[[1, 4], [6, 8], [9, 10]]$ Explanation: In the given intervals, we have only two overlapping intervals $[1, 3]$ and $[2, 4]$. Therefore, we will merge these two and return $[[1, 4], [6, 8], [9, 10]]$.

Input: $arr[] = [[7, 8], [1, 5], [2, 4], [4, 6]]$

Output: $[[1, 6], [7, 8]]$ Explanation: We will merge the overlapping intervals $[[1, 5], [2, 4], [4, 6]]$ into a single interval $[1, 6]$.

Solution:

```

import java.util.ArrayList;

import java.util.List;

import java.util.Arrays;

public class MergeIntervals {

    public static void main(String[] args) {

        solution(new int[][] {{1, 3}, {2, 4}, {6, 8}, {9, 10}});

        solution(new int[][] {{7, 8}, {1, 5}, {2, 4}, {4, 6}});

    }

    public static void solution(int[][] intervals) {

        if (intervals == null || intervals.length == 0) {

```

```

        System.out.println(Arrays.toString(new int[0][0]));
        return;
    }

    Arrays.sort(intervals, (a, b) -> Integer.compare(a[0], b[0]));
    List<int[]> merged = new ArrayList<>();

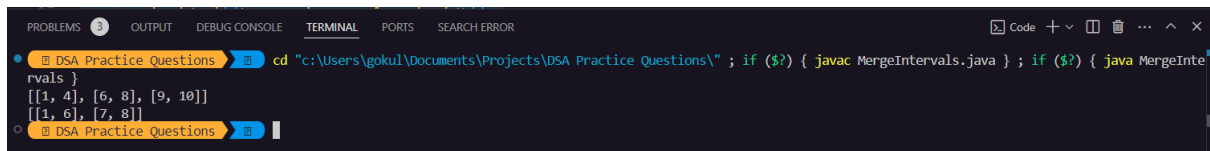
    for (int[] interval : intervals) {
        if (merged.isEmpty() || merged.get(merged.size() - 1)[1] < interval[0]) {
            merged.add(interval);
        } else {
            merged.get(merged.size() - 1)[1] = Math.max(merged.get(merged.size() - 1)[1], interval[1]);
        }
    }

    System.out.println(listToString(merged));
}

public static String listToString(List<int[]> list) {
    StringBuilder sb = new StringBuilder();
    sb.append("[");
    for (int i = 0; i < list.size(); i++) {
        sb.append(Arrays.toString(list.get(i)));
        if (i < list.size() - 1) sb.append(", ");
    }
    sb.append("]");
    return sb.toString();
}
}

```

Output:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR
DSA Practice Questions cd "c:\Users\gokul\Documents\Projects\DSA Practice Questions\" ; if ($?) { javac MergeIntervals.java } ; if ($?) { java MergeInte
rvals }
[[1, 4], [6, 8], [9, 10]]
[[1, 6], [7, 8]]
DSA Practice Questions
```

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

Space 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.

Input: $\{\{1, 0\}, \{0, 0\}\}$

Output: $\{\{1, 1\}, \{1, 0\}\}$

Input: $\{\{0, 0, 0\}, \{0, 0, 1\}\}$

Output: $\{\{0, 0, 1\}, \{1, 1, 1\}\}$ Input: $\{\{1, 0, 0, 1\}, \{0, 0, 1, 0\}, \{0, 0, 0, 0\}\}$ Output: $\{\{1, 1, 1, 1\}, \{1, 1, 1, 1\}, \{1, 0, 1, 1\}\}$

Solution:

```
public class BooleanMatrix {

    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}};

        modifyMatrix(mat1);

        printMatrix(mat1);

        modifyMatrix(mat2);

        printMatrix(mat2);

        modifyMatrix(mat3);

        printMatrix(mat3);

    }

    public static void modifyMatrix(int[][] mat) {

        int M = mat.length;
```

```

int N = mat[0].length;

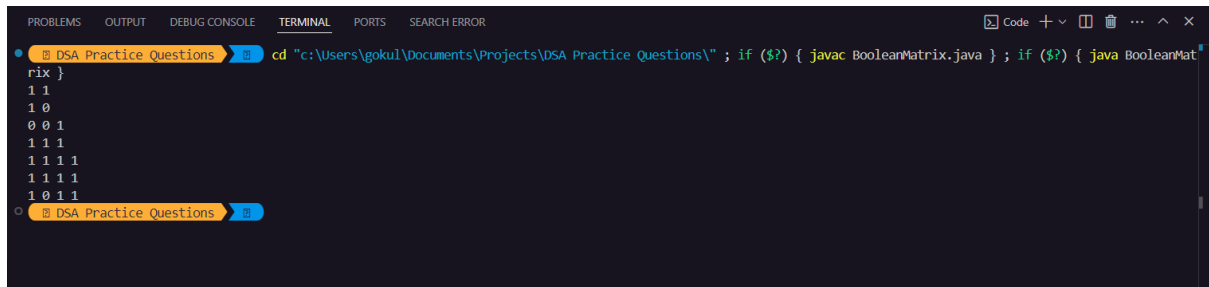
boolean[] row = new boolean[M];
boolean[] col = new boolean[N];
for (int i = 0; i < M; i++) {
    for (int j = 0; j < N; j++) {
        if (mat[i][j] == 1) {
            row[i] = true;
            col[j] = true;
        }
    }
}

for (int i = 0; i < M; i++) {
    for (int j = 0; j < N; j++) {
        if (row[i] || col[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();
    }
}

```

Output:



```
cd "c:\Users\gokul\Documents\Projects\DSA Practice Questions\" ; if ($?) { javac BooleanMatrix.java } ; if ($?) { java BooleanMat
rix }
1 1
1 1
1 0
0 0 1
1 1 1
1 1 1 1
1 1 1 1
1 0 1 1
0 1 1 1 1
1 1 1 1 1 1
1 1 1 1 1 1 1
```

Time Complexity: $O(M * N)$

Space Complexity: $O(M + N)$

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.

Input: matrix = {{1, 2, 3, 4}, {5, 6, 7, 8}, {9, 10, 11, 12}, {13, 14, 15, 16 }}

Output: 1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10

Input: matrix = { {1, 2, 3, 4, 5, 6}, {7, 8, 9, 10, 11, 12}, {13, 14, 15, 16, 17, 18}}

Output: 1 2 3 4 5 6 12 18 17 16 15 14 13 7 8 9 10 11 Explanation: The output is matrix in spiral format.

Solution:

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

```
import java.util.List;
```

```
public class Maximumsubarray{
```

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

```
        solution(new int[][] {{1, 2, 3, 4},{5, 6, 7, 8}, {9, 10, 11, 12}, {13, 14, 15, 16 }});
```

```
        solution(new int[][] {{1, 2, 3, 4, 5, 6},{7, 8, 9, 10, 11, 12}, {13, 14, 15, 16, 17, 18}});
```

```
    }
```

```
    public static void solution(int[][] num){
```

```
        List<Integer> result = new ArrayList<>();
```

```
        if (num == null || num.length == 0) System.out.println(result.toString());
```

```
int top = 0, bottom = num.length - 1;
int left = 0, right = num[0].length - 1;

while (top <= bottom && left <= right) {
    for (int i = left; i <= right; i++) {
        result.add(num[top][i]);
    }
    top++;

    for (int i = top; i <= bottom; i++) {
        result.add(num[i][right]);
    }
    right--;

    if (top <= bottom) {
        for (int i = right; i >= left; i--) {
            result.add(num[bottom][i]);
        }
        bottom--;
    }

    if (left <= right) {
        for (int i = bottom; i >= top; i--) {
            result.add(num[i][left]);
        }
        left++;
    }
}

System.out.println(result.toString());
}
```

Output:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR
DSA Practice Questions cd "c:\Users\gokul\Documents\Projects\DSA Practice Questions\" ; if ($?) { javac MaximumSubarray.java } ; if ($?) { java M
MaximumSubarray }
[1, 2, 3, 4, 8, 12, 16, 15, 14, 13, 9, 5, 6, 7, 11, 10]
[1, 2, 3, 4, 5, 6, 12, 18, 17, 16, 15, 14, 13, 7, 8, 9, 10, 11]
```

Time Complexity: $O(n*m)$

Space Complexity: $O(n*m)$

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.

Input: `str = "((()))()()"`

Output: Balanced Input: `str = "()()()()"`

Solution:

```
public class ParenthesesBalance {

    public static void main(String[] args) {

        System.out.println(isBalanced("((()))()()") ? "Balanced" : "Not Balanced");

        System.out.println(isBalanced "()()()()") ? "Balanced" : "Not Balanced");

    }

    public static boolean isBalanced(String str) {

        int count = 0;

        for (int i = 0; i < str.length(); i++) {

            if (str.charAt(i) == '(') {

                count++;

            } else if (str.charAt(i) == ')') {

                count--;

            }

            if (count < 0) {

                return false;

            }

        }

        return true;

    }

}
```

```

    }

    }

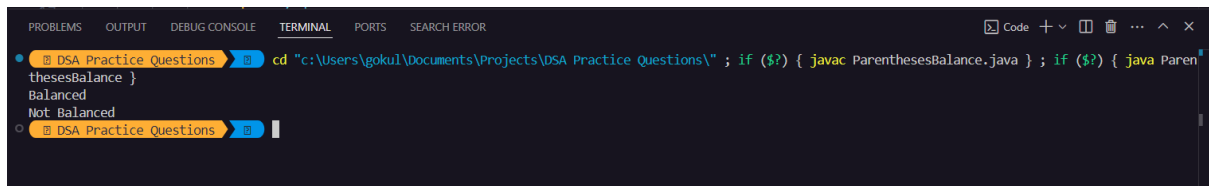
    return count == 0;

}

}

```

Output:



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR
DSA Practice Questions cd "c:\Users\gokul\Documents\Projects\DSA Practice Questions\" ; if ($?) { javac ParenthesesBalance.java } ; if ($?) { java ParenthesesBalance }
thesesBalance }
Balanced
Not Balanced
DSA Practice Questions

```

Time Complexity: $O(n)$

Space Complexity : $O(1)$

14. Check if two Strings are Anagrams of each other

Given two strings $s1$ and $s2$ 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.

Input: $s1 = \text{"geeks"}$ $s2 = \text{"kseeg"}$

Output: true

Explanation: Both the string have same characters with same frequency. So, they are anagrams.

Input: $s1 = \text{"allergy"}$ $s2 = \text{"allergic"}$

Output: false

Explanation: Characters in both the strings are not same. $s1$ has extra character „y“ and $s2$ has extra characters „i“ and „c“, so they are not anagrams.

Input: $s1 = \text{"g"}$, $s2 = \text{"g"}$

Output: true

Explanation: Characters in both the strings are same, so they are anagrams.

Solution:

```

import java.util.Arrays;

public class AnagramChecker {

    public static void main(String[] args) {

        System.out.println(areAnagrams("geeks", "kseeg"));
    }
}

```

```

        System.out.println(areAnagrams("allergy", "allergic"));

        System.out.println(areAnagrams("g", "g"));
    }

    public static boolean areAnagrams(String s1, String s2) {
        if (s1.length() != s2.length()) {
            return false;
        }

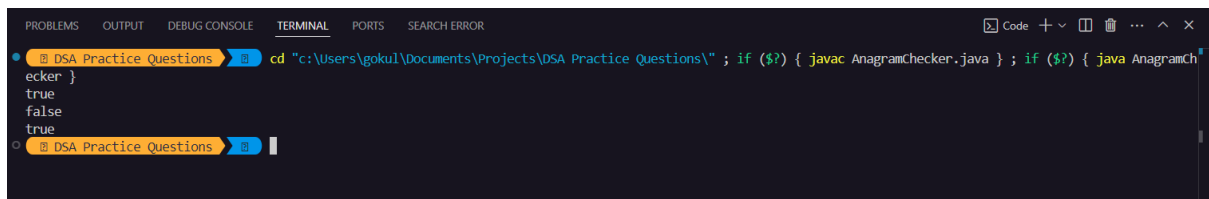
        char[] arr1 = s1.toCharArray();
        char[] arr2 = s2.toCharArray();

        Arrays.sort(arr1);
        Arrays.sort(arr2);

        return Arrays.equals(arr1, arr2);
    }
}

```

Output:



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR
• DSA Practice Questions cd "c:\Users\gokul\Documents\Projects\DSA Practice Questions\" ; if ($?) { javac AnagramChecker.java } ; if ($?) { java AnagramCh
ecker }
true
false
true
○ DSA Practice Questions

```

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

Space Complexity: $O(N)$

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.

Input: str = "forgeeksskeegfor"

Output: "geeksskeeg" Explanation: There are several possible palindromic substrings like "kssk", "ss", "eeksskeeg" etc. But the substring "geeksskeeg" is the longest among all.

Input: str = "Geeks"

Output: "ee"

Input: str = "abc"

Output: "a"

Input: str = "" Output: ""

Solution:

```
public class LongestPalindromicSubstring {  
    public static void main(String[] args) {  
        System.out.println(longestPalindrome("forgeeksskeegfor"));  
        System.out.println(longestPalindrome("Geeks"));  
        System.out.println(longestPalindrome("abc"));  
        System.out.println(longestPalindrome(""));  
    }  
  
    public static String longestPalindrome(String str) {  
        if (str == null || str.length() == 0) {  
            return "";  
        }  
  
        int start = 0;  
        int maxLength = 1;  
  
        for (int i = 0; i < str.length(); i++) {  
  
            int len1 = expandAroundCenter(str, i, i);  
  
            int len2 = expandAroundCenter(str, i, i + 1);  
  
            int len = Math.max(len1, len2);  
  
            if (len > maxLength) {
```



```

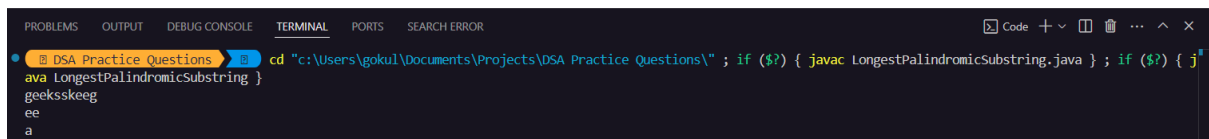
        maxLength = len;
        start = i - (len - 1) / 2;
    }
}

return str.substring(start, start + maxLength);
}

private static int expandAroundCenter(String str, int left, int right) {
    while (left >= 0 && right < str.length() && str.charAt(left) == str.charAt(right)) {
        left--;
        right++;
    }
    return right - left - 1;
}
}

```

Output:



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR
• DSA Practice Questions cd "c:\Users\gokul\Documents\Projects\DSA Practice Questions\" ; if ($?) { javac LongestPalindromicSubstring.java } ; if ($?) { j
ava LongestPalindromicSubstring }
geeksforgeeks
geeks
geek
geezer

```

Time Complexity: $O(N^2)$

Space Complexity: $O(1)$

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".

Input: `arr[] = ["geeksforgeeks", "geeks", "geek", "geezer"]`

Output: gee Explanation: "gee" is the longest common prefix in all the given strings.

Input: `arr[] = ["hello", "world"]`

Output: -1 Explanation: There's no common prefix in the given strings.

Solution:

```

import java.util.Arrays;

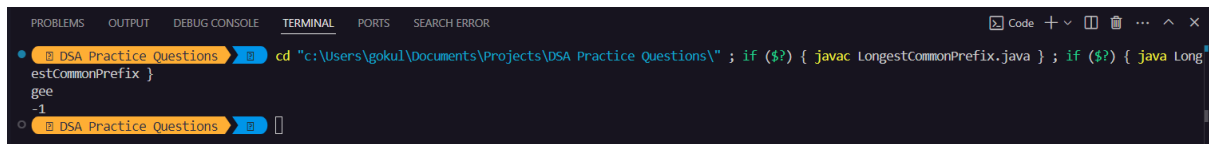
public class LongestCommonPrefix {
    public static void main(String[] args) {
        System.out.println(longestCommonPrefix(new String[]{"geeksforgeeks", "geeks", "geek",
"geezer"}));
        System.out.println(longestCommonPrefix(new String[]{"hello", "world"}));
    }

    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);
    }
}

```

Ouput:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR
DSA Practice Questions cd "c:\Users\gokul\Documents\Projects\DSA Practice Questions\" ; if ($?) { javac LongestCommonPrefix.java } ; if ($?) { java LongestCommonPrefix }
gee
-1
DSA Practice Questions
```

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

Space Complexity: $O(1)$

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.

Input : Stack[] = [1, 2, 3, 4, 5]

Output : Stack[] = [1, 2, 4, 5]

Input : Stack[] = [1, 2, 3, 4, 5, 6]

Output : Stack[] = [1, 2, 4, 5, 6]

Solution:

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

```
public class DeleteMiddleElement {
```

```
    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(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(stack2);
}

```

```

public static void deleteMiddleElement(Stack<Integer> stack) {
    int size = stack.size();
    int middle = size / 2;
    deleteMiddleHelper(stack, middle, 0);
}

```

```

private static void deleteMiddleHelper(Stack<Integer> stack, int middle, int count) {
    if (count == middle) {
        stack.pop();
        return;
    }

    int temp = stack.pop();
    deleteMiddleHelper(stack, middle, count + 1);
    stack.push(temp);
}
}

```

Output:

```

D:\DSA Practice Questions> cd "c:\Users\goku\Documents\Projects\DSA Practice Questions\" ; if ($?) { javac DeleteMiddleElement.java } ; if ($?) { java DeleteMiddleElement }
[1, 2, 4, 5]

```

Time Complexity: $O(n)$

Space 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.

Input: arr[] = [4 , 5 , 2 , 25]

Output: 4 5 2 -> 5 -> 25 -> 25 25 -> -1

Explanation: Except 25 every element has an element greater than them present on the right side

Input: arr[] = [13 , 7 , 6 , 12]

Output: 13 -> 7 -1 -> 12 6 12 -> 12 -> -1

Explanation: 13 and 12 don't have any element greater than them present on the right side

Solution:

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

```
public class NextGreaterElement {  
    public static void main(String[] args) {  
        int[] arr1 = {4, 5, 2, 25};  
        int[] arr2 = {13, 7, 6, 12};  
  
        printNextGreaterElements(arr1);  
        printNextGreaterElements(arr2);  
    }  
  
    public static void printNextGreaterElements(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("-1");  
            }  
        }  
    }  
}
```

```

    } else {

        System.out.println(stack.peek());

    }

    stack.push(arr[i]);

}

}

}

```

Output:

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR
DSA Practice Questions cd "c:\Users\gokul\Documents\Projects\DSA Practice Questions\" ; if ($?) { javac NextGreaterElement.java } ; if ($?) { java NextG
reaterElement }
-1
25
25
5
-1
12
12
-1

```

Time Complexity: $O(n)$

Space 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.

Solution:

```

import java.util.LinkedList;

import java.util.Queue;

```

```

class Node {

    int data;

    Node left, right;

    public Node(int data) {

        this.data = data;

        left = right = null;

    }
}

```

```
}
```

```
public class BinaryTree {
```

```
    Node root;
```

```
    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.root.right.right = new Node(6);
```

```
        tree.root.left.left.left = new Node(7);
```

```
        printRightView(tree.root);
```

```
    }
```

```
    public static void printRightView(Node root) {
```

```
        if (root == null) {
```

```
            return;
```

```
        }
```

```
        Queue<Node> queue = new LinkedList<>();
```

```
        queue.add(root);
```

```
        while (!queue.isEmpty()) {
```

```
            int size = queue.size();
```

```
            for (int i = 1; i <= size; i++) {
```

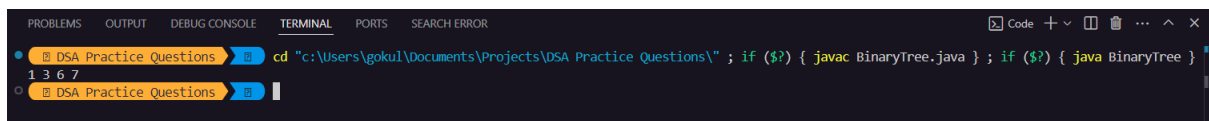
```
                Node node = queue.poll();
```

```

        if (i == size) {
            System.out.print(node.data + " ");
        }
        if (node.left != null) {
            queue.add(node.left);
        }
        if (node.right != null) {
            queue.add(node.right);
        }
    }
}
}
}
}

```

Output



```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR
DSA Practice Questions cd "c:\Users\gokul\Documents\Projects\DSA Practice Questions\" ; if ($?) { javac BinaryTree.java } ; if ($?) { java BinaryTree }
1 3 6 7
DSA Practice Questions

```

Time Complexity: $O(n)$

Space 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.

Solution:

```

class Node {
    int data;
    Node left, right;

    public Node(int data) {
        this.data = data;
        left = right = null;
    }
}

```



```
}  
}
```

```
public class BinaryTree {
```

```
    Node root;
```

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

```
        BinaryTree tree = new BinaryTree();
```

```
        tree.root = generateLargeTree(1, 1000);
```

```
        int height = maxDepth(tree.root);
```

```
        System.out.println("Height of the tree: " + height);
```

```
    }
```

```
    public static Node generateLargeTree(int startValue, int endValue) {
```

```
        Node root = null;
```

```
        for (int i = startValue; i <= endValue; i++) {
```

```
            root = insertNode(root, i);
```

```
        }
```

```
        return root;
```

```
    }
```

```
    // function to insert nodes in a binary search tree
```

```
    public static Node insertNode(Node root, int data) {
```

```
        if (root == null) {
```

```
            return new Node(data);
```

```
        }
```

```
        if (data < root.data) {
```

```
            root.left = insertNode(root.left, data);
```

```
        } else {
```

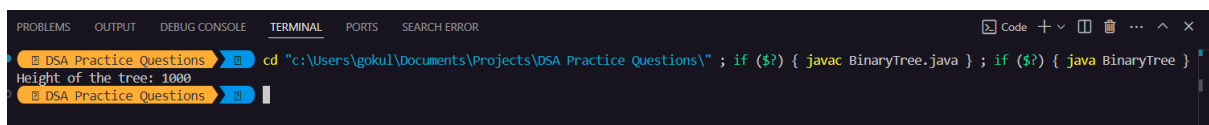
```
            root.right = insertNode(root.right, data);
```

```
        }
```

```
        return root;
```

```
}  
  
// Function to calculate the height  
  
public static int maxDepth(Node root) {  
    if (root == null) {  
        return 0;  
    }  
    int leftHeight = maxDepth(root.left);  
    int rightHeight = maxDepth(root.right);  
    return Math.max(leftHeight, rightHeight) + 1;  
}  
}
```

Output:



```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS SEARCH ERROR  
DSA Practice Questions cd "c:\Users\goku1\Documents\Projects\DSA Practice Questions\" ; if ($?) { javac BinaryTree.java } ; if ($?) { java BinaryTree }  
Height of the tree: 1000  
DSA Practice Questions
```

Time Complexity: $O(n)$

Space Complexity: $O(h)$