(09-11-2024) TECHNICAL TRAINING DSA - CODING PRACTICE PROBLEMS

Question 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.

CODE:

```
import java.util.*;
public class ex1{
    public static void main(String[]ex1){
        ArrayList<Long> arr=new ArrayList<>();
        Scanner sc=new Scanner(System.in);
        System.out.println("Array size:");
        int size=sc.nextInt();
        for(int i=0;i<size;i++){
            long n=sc.nextLong();
            arr.add(n);
```

```
        }
        long maxend=arr.get(0);
        long res=arr.get(0);
        for(int i=1;i<size;i++){
            maxend=Math.max(maxend+arr.get(i),arr.get(i));
            res=Math.max(res,maxend);
        }
        System.out.print("Result:"+res);
    }
}
```

Constraints:

OUTPUT:

```
PS D:\java>  & 'C:\Program Files\Java\jdk-22\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\av
ast\AppData\Roaming\Code\User\workspaceStorage\2b8587d1eb5f31426461b619d591d0d1\redhat.java\jdt_ws\java_72e8cc1b\bin' '
ex1'
Array size:
7
2
3
-8
2
2
2
2
2
3
Result:11
PS D:\java>  & 'C:\Program Files\Java\jdk-22\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\av
ast\AppData\Roaming\Code\User\workspaceStorage\2b8587d1eb5f31426461b619d591d0d1\redhat.java\jdt_ws\java_72e8cc1b\bin' '
ex1'
Array size:
2
-2
-4
Result:-2
PS D:\java>
```

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

Question 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}.

Code:

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

public class ex2 {
    public static void main(String[] ex2){
        ArrayList<Long> arr=new ArrayList<>();
        Scanner sc=new Scanner(System.in);
        System.out.println("Array size:");
        int size=sc.nextInt();
        for(int i=0;i<size;i++){
            long n=sc.nextLong();
            arr.add(n);
        }
        long prod=arr.get(0);
        long mini=arr.get(0);
        long maxi=arr.get(0);
        for(int i=1;i<size;i++){
```

```
            long temp=Math.max(arr.get(i),Math.max(maxi*arr.get(i),mini*arr.get(i)));
            mini=Math.min(arr.get(i),Math.min(maxi*arr.get(i),mini*arr.get(i)));
            maxi=temp;
            prod=Math.max(prod,maxi);
        }
        System.out.println(prod);
    }
}
```

Output:

```
PS D:\java>  & 'C:\Program Files\Java\jdk-22\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\av
ast\AppData\Roaming\Code\User\workspaceStorage\2b8587d1eb5f31426461b619d591d0d1\redhat.java\jdt_ws\java_72e8cc1b\bin' '
ex2'
Array size:
6
2
6
-3
-10
0
2
360
PS D:\java>  & 'C:\Program Files\Java\jdk-22\bin\java.exe' '-XX:+ShowCodeDetailsInExceptionMessages' '-cp' 'C:\Users\av
ast\AppData\Roaming\Code\User\workspaceStorage\2b8587d1eb5f31426461b619d591d0d1\redhat.java\jdt_ws\java_72e8cc1b\bin' '
ex2'
Array size:
6
-2
6
-3
-10
0
2
180
```

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

Question 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

Code:

```
#include<iostream>
using namespace std;
int binarySearch(int nums[],int left,int right,int target){
   while (left <= right) {
      int mid = (left + right) / 2;

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

```
            } else {
                right = mid - 1;
            }
        }
    }

    return -1;
}
int main(){
    int n,key;
    cout << "Enter the number of elements: ";
    cin >> n;
    int arr[n];
    cout << "Enter " << n << " elements:" << endl;
    for (int i = 0; i < n; i++) {
        cin >> arr[i];
    }
    cout<<"enter the element to find:";
    cin>>key;
    int low=0;
    int high=n-1;
    int b1=binarySearch(arr,low,high,key);
    cout<<b1;
}
```

Output:

```
PS D:\c++\output> & .\'ex3.exe'
Enter the number of elements: 5
Enter 5 elements:
50
10
20
30
40
Enter the element to find: 10
Element found at index: 1
PS D:\c++\output> & .\'ex3.exe'
Enter the number of elements: 7
Enter 7 elements:
4
5
6
7
0
1
2
Enter the element to find: 3
Element not found in the array.
PS D:\c++\output> & .\'ex3.exe'
Enter the number of elements: 7
Enter 7 elements:
4
5
6
7
0
1
2
Enter the element to find: 0
Element found at index: 4
PS D:\c++\output>
```

Question 3:

Container with Most Water

Given n non-negative integers $a_1, a_2, \ldots, a_n$ 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.

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

Code:

```
#include<iostream>
using namespace std;
int main() {
    int n;
    cout << "Enter the number of elements: ";
    cin >> n;
    int height[n];
    cout << "Enter " << n << " elements:" << endl;
    for (int i = 0; i < n; i++) {
        cin >> height[i];
    }
    int max_area = 0;
    int left = 0;
    int right = n - 1;

    while (left < right) {
        int h = min(height[left], height[right]);
```

```
        int w = right - left;
        max_area = max(max_area, h * w);

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

    cout<<max_area;
}
```

Output:

```
Enter the number of elements: 4
Enter 4 elements:
1
5
4
3
6
PS D:\c++\output> & .\'ex4.exe'
Enter the number of elements: 5
Enter 5 elements:
3
1
2
4
5
12
```

Question 5:

Find the Factorial of a large number

Input: 100

Output:

93326215443944152681699238856266700490715968264381621468592963895217599993229
915608941463976156518286253697920827223758251185210916864000000000000000000000
0000

Input: 50

Output: 30414093201713378043612608166064768844377641568960512000000000000

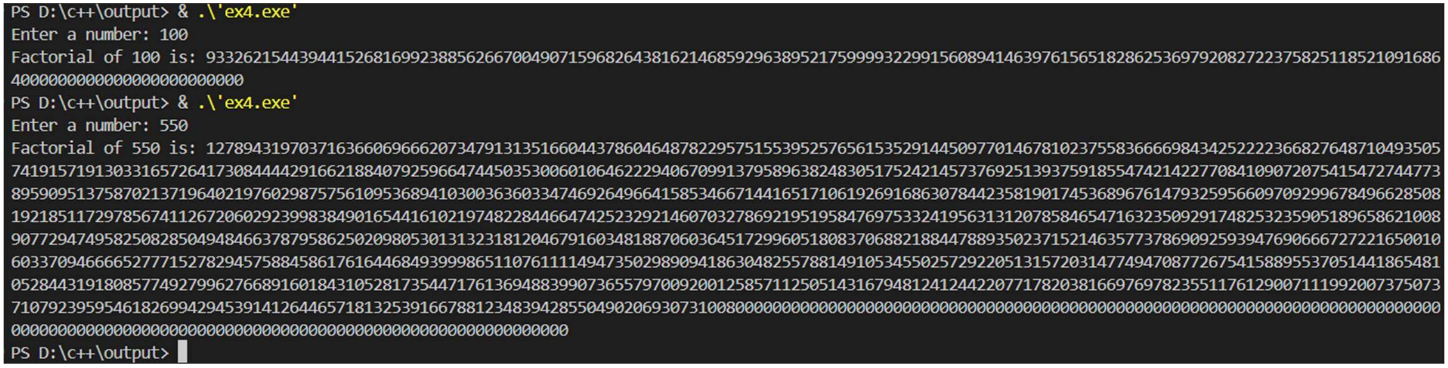
Code:

```
#include <iostream>
#include <vector>
using namespace std;

void multiply(vector<int>& result, int num) {
    int carry = 0;
    for (int i = 0; i < result.size(); i++) {
        int product = result[i] * num + carry;
        result[i] = product % 10;
        carry = product / 10;
    }
    while (carry) {
        result.push_back(carry % 10);
        carry /= 10;
    }
}
void factorial(int n) {
    vector<int> result;
```

```
    result.push_back(1);
    for (int i = 2; i <= n; i++) {
        multiply(result, i);
    }
    for (int i = result.size() - 1; i >= 0; i--) {
        cout << result[i];
    }
    cout << endl;
}
int main() {
    int n;
    cout << "Enter a number: ";
    cin >> n;
    cout << "Factorial of " << n << " is: ";
    factorial(n);
    return 0;
}
```

Output:



```
PS D:\c++\output> & .\'ex4.exe'
Enter a number: 100
Factorial of 100 is: 9332621544394415268169923885626670049071596826438162146859296389521759999322991560894146397615651828625369792082722375825118521091686
4000000000000000000000000
PS D:\c++\output> & .\'ex4.exe'
Enter a number: 550
Factorial of 550 is: 1278943197037163660696662073479131351660443786046487822957515539525765615352914450977014678102375583666698434252222366827648710493505
7419157191303316572641730844442916621884079259664744503530060106462229406709913795896382483051752421457376925139375918554742142277084109072075415472744773
8959095137587021371964021976029875756109536894103003636033474692649664158534667144165171061926916863078442358190174536896761479325956609709299678496628508
1921851172978567411267206029239983849016544161021974822844664742523292146070327869219519584769753324195631312078584654716323509291748253235905189658621008
9077294749582508285049484663787958625020980530131323181204679160348188706036451729960518083706882188447889350237152146357737869092593947690666727221650010
6033709466665277715278294575884586176164468493999865110761111494735029890941863048255788149105345502572922051315720314774947087726754158895537051441865481
0528443191808577492799627668916018431052817354471761369488399073655797009200125857112505143167948124124422077178203816697697823551176129007111992007375073
7107923959546182699429453914126446571813253916678812348394285504902069307310080000000000000000000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000000000000000000000000000
PS D:\c++\output> 
```

Question 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.

Input: arr[] = {3, 0, 1, 0, 4, 0, 2}

Output: 10

Explanation: The expected rainwater to be trapped is shown in the above image.

Input: arr[] = {3, 0, 2, 0, 4}

Output: 7

Explanation: We trap 0 + 3 + 1 + 3 + 0 = 7 units.

Input: arr[] = {1, 2, 3, 4}

Output: 0

Explanation : We cannot trap water as there is no height bound on both sides

Input: arr[] = {10, 9, 0, 5}

Output: 5

Explanation : We trap 0 + 0 + 5 + 0 = 5

Code:

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

int trap(vector<int>& height) {
    int i = 0, j = height.size() - 1;
    int left_max = height[i], right_max = height[j];
    int sum = 0;
```

```
    while (i < j) {
        if (left_max <= right_max) {
            sum += left_max - height[i];
            i++;
            left_max = max(left_max, height[i]);
        } else {
            sum += right_max - height[j];
            j--;
            right_max = max(right_max, height[j]);
        }
    }
    return sum;
}

int main() {
    int n;
    cout << "Enter number of elements: ";
    cin >> n;
    vector<int> height(n);
    cout << "Enter the heights: ";
    for (int i = 0; i < n; i++) {
        cin >> height[i];
    }
    cout << "Amount of water trapped: " << trap(height) << endl;
    return 0;
}
```

Output:

```
Enter number of elements: 7
Enter the heights: 3
0
1
0
4
0
2
Amount of water trapped: 10
PS D:\c++\output> & .\'ex4.exe'
Enter number of elements: 5
Enter the heights: 3
0
2
0
4
Amount of water trapped: 7
```

Question 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.

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.

Input: 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.

Code:

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <limits.h>
using namespace std;

int findMinDiff(vector<int> &arr, int m) {
    int n = arr.size();
    sort(arr.begin(), arr.end());
    int minDiff = INT_MAX;

    for (int i = 0; i + m - 1 < n; i++) {
        int diff = arr[i + m - 1] - arr[i];
        if (diff < minDiff)
            minDiff = diff;
    }
    return minDiff;
}

int main() {
    int n, m;
    cout << "Enter the number of elements in the array: ";
    cin >> n;
    vector<int> arr(n);
    cout << "Enter the elements of the array: ";
    for (int i = 0; i < n; i++) {
```

```
        cin >> arr[i];
    }
    cout << "Enter the value of m: ";
    cin >> m;
    cout << "Minimum difference: " << findMinDiff(arr, m) << endl;
    return 0;
}
```

Output:

```
Enter the number of elements in the array: 7
Enter the elements of the array: 7 3 2 4 9 12 56
Enter the value of m: 3
Minimum difference: 2
```

Question 8:

Merge Overlapping Intervals

Given an array of time intervals where arr[i] = [starti, endi], 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].

Code:

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;

vector<vector<int>> mergeOverlap(vector<vector<int>> &arr) {
    int n = arr.size();
    sort(arr.begin(), arr.end());
    vector<vector<int>> res;

    for (int i = 0; i < n; i++) {
        int start = arr[i][0];
        int end = arr[i][1];
        if (!res.empty() && res.back()[1] >= end)
            continue;
        for (int j = i + 1; j < n; j++) {
            if (arr[j][0] <= end)
                end = max(end, arr[j][1]);
        }
        res.push_back({start, end});
    }
    return res;
}

int main() {
    int n;
    cout << "Enter the number of intervals: ";
```

```
    cin >> n;
    vector<vector<int>> arr(n, vector<int>(2));
    cout << "Enter the intervals (start end):" << endl;
    for (int i = 0; i < n; i++) {
        cin >> arr[i][0] >> arr[i][1];
    }
    vector<vector<int>> res = mergeOverlap(arr);
    for (auto interval : res) {
        cout << interval[0] << " " << interval[1] << endl;
    }
    return 0;
}
```

Output:

```
Enter the number of intervals: 4
Enter the intervals (start end):
7 8
1 5
2 4
4 6
1 6
7 8
```

Question 9:

A Boolean Matrix Question

Given a boolean matrix mat[M][N] of size M X N, modify it such that if a matrix cell mat[i][j] is

1 (or true) then make all the cells of ith row and jth 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}}

Code:

```
#include<bits/stdc++.h>
using namespace std;

void setZeroes(vector<vector<int>>& matrix) {
    int rows = matrix.size(), cols = matrix[0].size();
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            if (matrix[i][j] == 1) {
                int ind = i - 1;
                while (ind >= 0) {
                    if (matrix[ind][j] != 1) {
                        matrix[ind][j] = -1;
                    }
                    ind--;
                }
                ind = i + 1;
                while (ind < rows) {
                    if (matrix[ind][j] != 1) {
                        matrix[ind][j] = -1;
                    }
                    ind++;
                }
                ind = j - 1;
                while (ind >= 0) {
```

```
                if (matrix[i][ind] != 1) {
                    matrix[i][ind] = -1;
                }
                ind--;
            }
            ind = j + 1;
            while (ind < cols) {
                if (matrix[i][ind] != 1) {
                    matrix[i][ind] = -1;
                }
                ind++;
            }
        }
    }
}
for (int i = 0; i < rows; i++) {
    for (int j = 0; j < cols; j++) {
        if (matrix[i][j] < 0) {
            matrix[i][j] = 1;
        }
    }
}
}

int main() {
    int rows, cols;
    cout << "Enter number of rows: ";
    cin >> rows;
```

```
    cout << "Enter number of columns: ";
    cin >> cols;
    vector<vector<int>> arr(rows, vector<int>(cols));
    cout << "Enter the matrix elements: " << endl;
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            cin >> arr[i][j];
        }
    }
    setZeroes(arr);
    cout << "The Final Matrix is: " << endl;
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            cout << arr[i][j] << " ";
        }
        cout << "\n";
    }
}
```

Output:

```
PS D:\c++\output> & .\'ex4.exe'
Enter number of rows: 3
Enter number of columns: 4
Enter the matrix elements:
1 0 0 1
0 0 1 0
0 0 0 0
The Final Matrix is:
1 1 1 1
1 1 1 1
1 0 1 1
```

Question 10:

Print a given matrix in spiral form

Given an m x 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.

Code:

```
#include <bits/stdc++.h>
using namespace std;

void spiralPrint(int m, int n, vector<vector<int>>& a) {
   int top = 0, bottom = m - 1, left = 0, right = n - 1;

   while (top <= bottom && left <= right) {
      for (int i = left; i <= right; ++i) {
         cout << a[top][i] << " ";
      }
      top++;
```

```
        for (int i = top; i <= bottom; ++i) {
            cout << a[i][right] << " ";
        }
        right--;

        if (top <= bottom) {
            for (int i = right; i >= left; --i) {
                cout << a[bottom][i] << " ";
            }
            bottom--;
        }

        if (left <= right) {
            for (int i = bottom; i >= top; --i) {
                cout << a[i][left] << " ";
            }
            left++;
        }
    }
}

int main() {
    int m, n;
    cin >> m >> n;
    vector<vector<int>> matrix(m, vector<int>(n));

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

```
        for (int j = 0; j < n; j++) {
            cin >> matrix[i][j];
        }
    }

    spiralPrint(m, n, matrix);
    return 0;
}
```

Output:

```
PS D:\c++\output> & .\'ex4.exe'
4 4
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
1 2 3 4 8 12 16 15 14 13 9 5 6 7 11 10
```

Time complexity: O(m * n)

Space complexity: O(m * n).

Question 11:

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 = “())((())”

Output: Not Balanced

Code:

```
#include <bits/stdc++.h>
```

```
using namespace std;

bool checkMatch(char c1, char c2){
    return (c1 == '(' && c2 == ')') || (c1 == '[' && c2 == ']') || (c1 == '{' && c2 == '}');
}

bool ispar(string s){
    int top = -1;
    for (int i = 0; i < s.length(); ++i){
        if (top < 0 || !checkMatch(s[top], s[i])){
            ++top;
            s[top] = s[i];
        }
        else{
            --top;
        }
    }
    return top == -1;
}

int main(){
    string s;
    getline(cin, s);
    s.erase(remove(s.begin(), s.end(), ' '), s.end());
    cout << (ispar(s) ? "true" : "false") << endl;
    return 0;
}
```

Output:

```
PS D:\c++\output> & .\'ex4.exe'
{()} []
true
```

Time Complexity:O(n)

Space Complexity:O(1)

Question 12:

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 = “geeks” s2 = “kseeg”

Output: true

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

Input: s1 = “allergy” s2 = “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 = “g”, s2 = “g”

Output: true

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

Code:

```
#include <iostream>
#include <unordered_map>
#include <algorithm>
using namespace std;
```

```
bool areAnagrams(string &s1, string &s2) {
    unordered_map<char, int> charCount;
    for (char ch : s1) charCount[ch]++;
    for (char ch : s2) charCount[ch]--;
    for (auto &pair : charCount) {
        if (pair.second != 0) return false;
    }
    return true;
}

int main() {
    string s1, s2;
    getline(cin, s1);
    getline(cin, s2);
    s1.erase(remove(s1.begin(), s1.end(), ' '), s1.end());
    s2.erase(remove(s2.begin(), s2.end(), ' '), s2.end());
    cout << (areAnagrams(s1, s2) ? "true" : "false") << endl;
    return 0;
}
```

Output:

```
PS D:\c++\output> & .\'ex4.exe'
greeks
kseeg
false
PS D:\c++\output> & .\'ex4.exe'
geeks
kseeg
true
```

Time Complexity: O(m + n)

Space Complexity:O(1)

Question 13:

. 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”, “eeksskee” etc.

But the substring “geeksskeeg” is the longest among all.

Input: str = “Geeks”

Output: “ee”

Input: str = “abc”

Output: “a”

Input: str = “”

Output: “”

Code:

```
#include <bits/stdc++.h>
using namespace std;

bool checkPal(const string &s, int low, int high) {
    while (low < high) {
        if (s[low] != s[high])
            return false;
        low++;
        high--;
    }
    return true;
```

```
}

string longestPalSubstr(string& s) {
    int n = s.size();
    int maxLen = 1, start = 0;

    for (int i = 0; i < n; i++) {
        for (int j = i; j < n; j++) {
            if (checkPal(s, i, j) && (j - i + 1) > maxLen) {
                start = i;
                maxLen = j - i + 1;
            }
        }
    }

    return s.substr(start, maxLen);
}

int main() {
    string s;
    cout << "Enter the string: ";
    getline(cin, s);
    s.erase(remove(s.begin(), s.end(), ' '), s.end());
    cout << longestPalSubstr(s) << endl;
    return 0;
}
```

Output:

```
PS D:\c++\output> & .\'ex4.exe'
Enter the string: forgeeksskeegfor
geeksskeeg
```

Time Complexity:O(n^3)

Space Complexity:O(1)

Question 14:

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.

Code:

```
#include <bits/stdc++.h>
using namespace std;

string longestCommonPrefix(vector<string>& arr) {
    if (arr.empty()) return "-1";
    sort(arr.begin(), arr.end());
    string first = arr.front();
    string last = arr.back();
    int minLength = min(first.size(), last.size());
    int i = 0;
```

```
    while (i < minLength && first[i] == last[i]) {
        i++;
    }
    if (i == 0) return "-1";
    return first.substr(0, i);
}

int main() {
    vector<string> arr = {"geeksforgeeks", "geeks", "geek", "geezer"};
    cout << longestCommonPrefix(arr) << endl;
    return 0;
}
```

Output:

```
gee
```

Time Complexity: O(n log n + m)

Space Complexity: O(1)

Question 15:

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]

Code:

```
#include <iostream>
#include <stack>
using namespace std;

void deleteMiddle(stack<int>& st, int current, int size) {
    if (current == size / 2) {
        st.pop();
        return;
    }
    int top = st.top();
    st.pop();
    deleteMiddle(st, current + 1, size);
    st.push(top);
}

int main() {
    stack<int> st;
    st.push(1);
    st.push(2);
    st.push(3);
    st.push(4);
    st.push(5);
    int size = st.size();
    deleteMiddle(st, 0, size);
    while (!st.empty()) {
```

```
        cout << st.top() << " ";
        st.pop();
    }
    cout << endl;
    return 0;
}
```

Output:

```
[Running] cd "d:\c++\" && g++ ex4.cpp -o ex4 && "d:\c++\"ex4
5 4 2 1
```

Time Complexity:O(n)

Space Complexity:O(n)

Question 16:

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

Code:

```
#include<bits/stdc++.h>

using namespace std;
class Solution {
  public:
    vector < int > nextGreaterElements(vector < int > & nums) {
      int n = nums.size();
      vector < int > nge(n, -1);
      stack < int > st;
      for (int i = 2 * n - 1; i >= 0; i--) {
        while (!st.empty() && st.top() <= nums[i % n]) {
          st.pop();
        }

        if (i < n) {
          if (!st.empty()) nge[i] = st.top();
        }
        st.push(nums[i % n]);
      }
      return nge;
    }
};
int main() {
```

```
  Solution obj;
  vector < int > v {5,7,1,2,6,0};
  vector < int > res = obj.nextGreaterElements(v);
  cout << "The next greater elements are" << endl;
  for (int i = 0; i < res.size(); i++) {
    cout << res[i] << " ";
  }
}
```

Output:

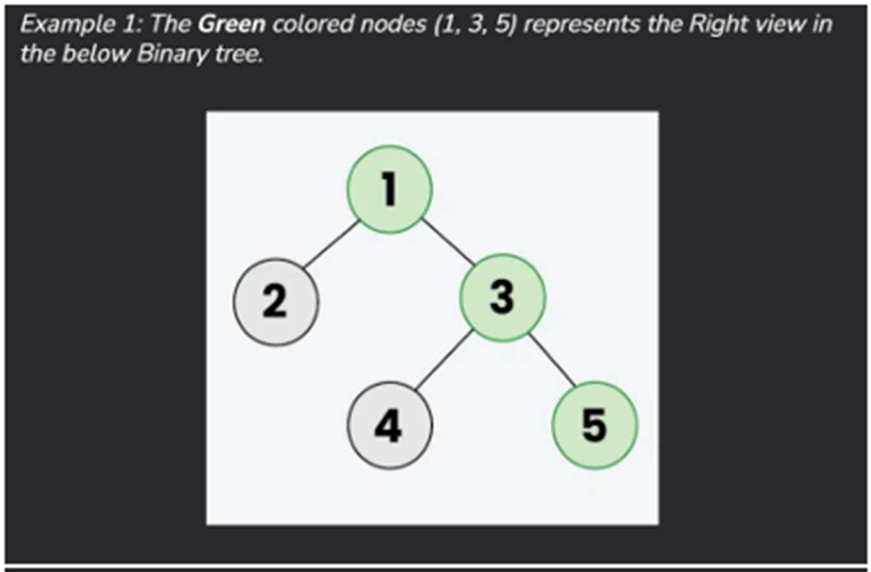
```
[Running] cd "d:\c++\" && g++ ex4.cpp -o ex4 && "d:\c++\"ex4
The next greater elements are
5 25 25 -1
```

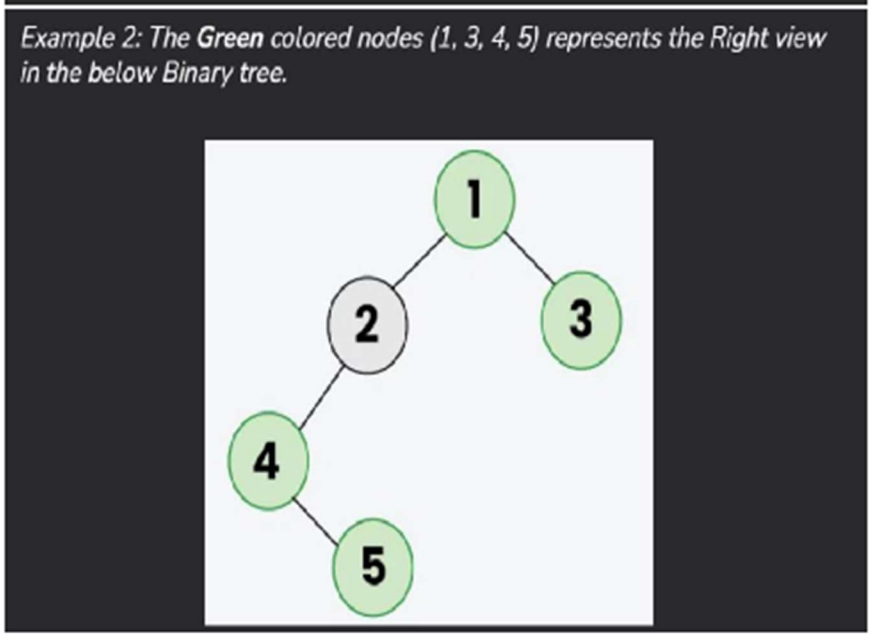
Time Complexity: O(N)

Space Complexity: O(N)

Question 17:

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:

```
#include <bits/stdc++.h>
using namespace std;

class Node {
public:
    int data;
    Node *left, *right;
    Node(int x) {
        data = x;
```

```
        left = right = nullptr;
    }
};

void RecursiveRightView(Node* root, int level, int& maxLevel, vector<int>& result) {
    if (!root) return;
    if (level > maxLevel) {
        result.push_back(root->data);
        maxLevel = level;
    }
    RecursiveRightView(root->right, level + 1, maxLevel, result);
    RecursiveRightView(root->left, level + 1, maxLevel, result);
}

vector<int> rightView(Node *root) {
    vector<int> result;
    int maxLevel = -1;
    RecursiveRightView(root, 0, maxLevel, result);
    return result;
}

int main() {
    string treeInput;
    cout << "Enter tree structure as level-order (comma separated, no spaces): ";
    cin >> treeInput;

    treeInput.erase(remove(treeInput.begin(), treeInput.end(), ' '), treeInput.end());
    stringstream ss(treeInput);
```

```
queue<Node*> q;

int data;
char delimiter;
ss >> data;
Node *root = new Node(data);
q.push(root);

while (!q.empty()) {
    Node *node = q.front();
    q.pop();

    if (ss >> delimiter >> data) {
        if (data != -1) {
            node->left = new Node(data);
            q.push(node->left);
        }
    }

    if (ss >> delimiter >> data) {
        if (data != -1) {
            node->right = new Node(data);
            q.push(node->right);
        }
    }
}

vector<int> result = rightView(root);
```

```
    for (int val : result) {
        cout << val << " ";
    }
    cout << endl;


    return 0;
}
```

Output:

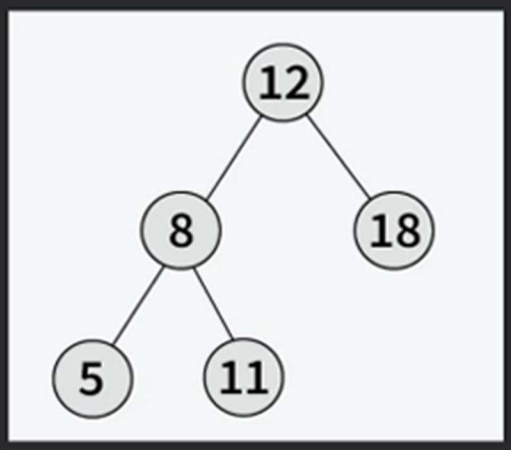
```
Enter tree structure as level-order (comma separated, no spaces): 1,2,3,-1
    ,-1,4,5
1 3 5
```

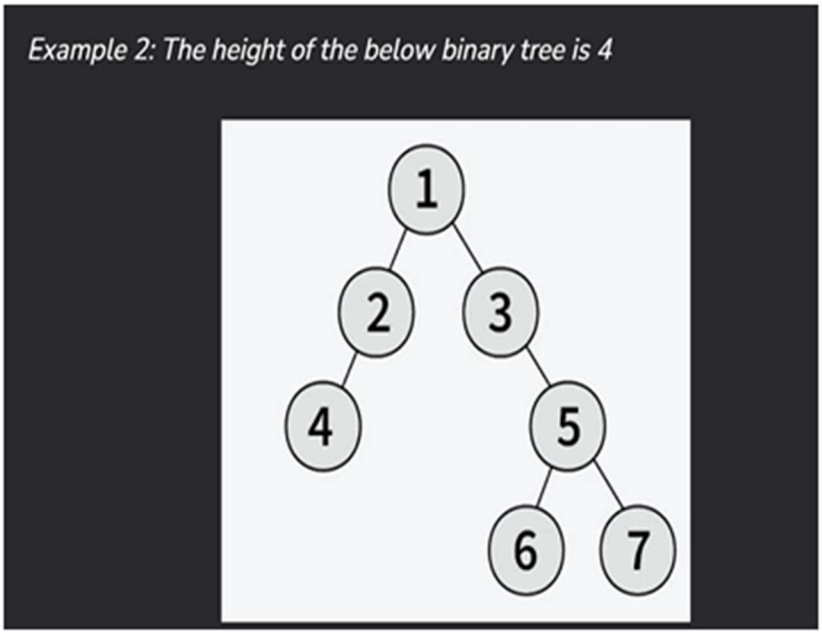
Question 18:

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.

Example 1: The height of the below binary tree is 3.





Code:

```
#include <bits/stdc++.h>
using namespace std;

struct Node {
    int data;
    Node *left;
    Node *right;
    Node(int val) {
        data = val;
        left = nullptr;
        right = nullptr;
    }
};

int maxDepth(Node *node) {
    if (node == nullptr)
```

```
        return 0;
    int lDepth = maxDepth(node->left);
    int rDepth = maxDepth(node->right);
    return max(lDepth, rDepth) + 1;
}

int main() {
    int n, val;
    cout << "Enter the number of nodes: ";
    cin >> n;

    if (n == 0) {
        cout << "Tree is empty" << endl;
        return 0;
    }

    cout << "Enter root value: ";
    cin >> val;
    Node *root = new Node(val);

    queue<Node*> q;
    q.push(root);

    for (int i = 1; i < n; i++) {
        cout << "Enter left or right child of node " << q.front()->data << ": ";
        cin >> val;
        Node *child = new Node(val);
        cout << "Is it left (1) or right (0)? ";
```

```
        int choice;
        cin >> choice;

        if (choice == 1)
            q.front()->left = child;
        else
            q.front()->right = child;

        q.push(child);
        q.pop();
    }

    cout << "Max Depth of the Tree: " << maxDepth(root) << endl;
    return 0;
}
```

Output:

```
Enter the number of nodes: 5
Enter root value: 1
Enter left or right child of node 1: 2
Is it left (1) or right (0)? 1
Enter left or right child of node 2: 3
Is it left (1) or right (0)? 0
Enter left or right child of node 3: 4
Is it left (1) or right (0)? 1
Enter left or right child of node 4: 5
Is it left (1) or right (0)? 0
Max Depth of the Tree: 4
```