TECHNICAL TRAINING DSA - CODING PRACTICE

PROBLEMS

Name: Rishi Kumar S
Dept: CSBS
Date: 10-11-2024

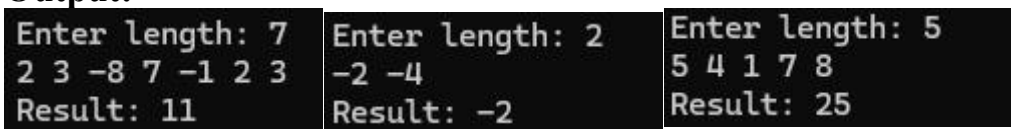
Question 1:
Maximum Subarray Sum – Kadane‟s Algorithm:
Given an array arr[], the task is to find the subarray that has the maximum sum and
return its
sum.
Input: arr[] = {2, 3, -8, 7, -1, 2, 3}
Output: 11
Explanation: The subarray {7, -1, 2, 3} has the largest sum 11.
Input: arr[] = {-2, -4}
Output: –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.

Answer:

```
#include <iostream>
#include <vector>

using namespace std;
int main(){
   cout<<"Enter length: ";
   int n;
   cin>>n;
   long long arr[n];
   for(int i=0;i<n;i++){
      cin>>arr[i];
   }
   long long maxi=arr[0];
   long long ans=arr[0];
   for(int i=1;i<n;i++){
      maxi=max(maxi+arr[i],arr[i]);
      ans=max(ans,maxi);
```

```
    }
    cout<<"Result: "<<ans;
    }
```

Output:



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}

Answer:

```
#include <iostream>
#include <vector>

using namespace std;
int main(){
    cout<<"Enter length: ";
    int n;
    cin>>n;
    long long arr[n];
    for(int i=0;i<n;i++){
        cin>>arr[i];
    }
    long long maxi=arr[0];
    long long mini=arr[0];
    long long prod=arr[0];
    for(int i=1;i<n;i++){
```

```
        if(arr[i]<0) swap(maxi,mini);
        mini=min(mini*arr[i],arr[i]);
        maxi=max(maxi*arr[i],arr[i]);
        prod=max(maxi,prod);
    }
    cout<<"Result: "<<prod;
    }
```

Output:

```
Enter length: 6
-2 6 -3 -10 0 2
Result: 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

Answer:

```
#include <iostream>
#include <vector>

using namespace std;
int main(){
    cout<<"Enter length: ";
    int n;
    cin>>n;
    long long arr[n];
    for(int i=0;i<n;i++){
        cin>>arr[i];
    }
    int target;
```

```
    cout<<"Enter target: ";
    cin>>target;
    int left=0,right=n-1;
    while(left<right){
        int mid=(left+right)/2;
        if(arr[mid]==target){
            cout<<mid;
            break;
        }
        else if(arr[mid]>=arr[left]){
            if(arr[left]<=target && target<=arr[mid]){
                right=mid-1;
            }
            else left=mid+1;
        }
        else{
            if(arr[mid]<=target && target<=arr[right]){
                left=mid+1;
            }
            else right=mid-1;
        }
    }
}
```

Output:

```
Enter length: 7
4 5 6 7 0 1 2
Enter target: 3
-1
```

```
Enter length: 5
50 10 20 30 40
Enter target: 10
1
```

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

QUESTION 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

Answer:

```
#include <iostream>
#include <vector>

using namespace std;
int main(){
   cout<<"Enter length: ";
   int n;
   cin>>n;
   long long arr[n];
   for(int i=0;i<n;i++){
      cin>>arr[i];
   }
   int maxi=0;
   int left=0,right=n-1;
   while(left<right){
      int minn=min(arr[left],arr[right]);
      int area=minn*(right-left);
      maxi=max(maxi,area);
      if(arr[left]>arr[right]){
         right--;
      }
      else left++;
   }
   cout<<maxi;
}
```

Output:

```
Enter length: 4
1 5 4 3
6
```

```
Enter length: 5
3 1 2 4 5
12
```

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

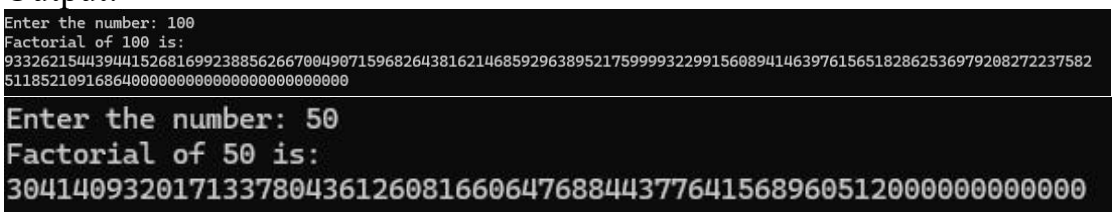
QUESTION 5
Find the Factorial of a large number
Input: 100
Output:
933262154439441526816992388562667004907159682643816214685929638952175999932
299
156089414639761565182862536979208272237582511852109168640000000000000000
000
00
Input: 50
Output:
30414093201713378043612608166064768844377641568960512000000000000

Answer:

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

void multiply(std::vector<int>& result, int num) {
    int carry = 0;
    for (auto& digit : result) {
        int product = digit * num + carry;
        digit = product % 10;
        carry = product / 10;
    }
    while (carry) {
        result.push_back(carry % 10);
        carry /= 10;
    }
}

void largeFactorial(int n) {
    std::vector<int> result = {1};

    for (int i = 2; i <= n; ++i) {
        multiply(result, i);
    }

    std::cout << "Factorial of " << n << " is:\n";
    for (auto it = result.rbegin(); it != result.rend(); ++it) {
```

```
        std::cout << *it;
    }
    std::cout << std::endl;
}

int main() {
    int n;
    cin>>n;
    largeFactorial(n);
    return 0;
}
```

Output:



Time Complexity: O(n*2 logn)
Space Complexity: O(n)

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}

Answer:

```
#include <iostream>
#include <vector>
#include <algorithm>

int trappingRainwater(const std::vector<int>& arr) {
    int n = arr.size();
    if (n < 3) return 0;
    std::vector<int> leftMax(n), rightMax(n);
    leftMax[0] = arr[0];
    for (int i = 1; i < n; i++) {
        leftMax[i] = std::max(leftMax[i - 1], arr[i]);
    }
    rightMax[n - 1] = arr[n - 1];
    for (int i = n - 2; i >= 0; I--) {
        rightMax[i] = std::max(rightMax[i + 1], arr[i]);
    }
    int totalWater = 0;
    for (int i = 0; i < n; i++) {
        totalWater += std::min(leftMax[i], rightMax[i]) - arr[i];
    }

    return totalWater;
}

int main() {
    std::vector<int> arr1 = {3, 0, 1, 0, 4, 0, 2};
    std::vector<int> arr2 = {3, 0, 2, 0, 4};
    std::vector<int> arr3 = {1, 2, 3, 4};
    std::vector<int> arr4 = {10, 9, 0, 5};

    std::cout << "Trapped water for arr1: " << trappingRainwater(arr1) <<
std::endl;
    std::cout << "Trapped water for arr2: " << trappingRainwater(arr2) <<
std::endl;
    std::cout << "Trapped water for arr3: " << trappingRainwater(arr3) <<
std::endl;
    std::cout << "Trapped water for arr4: " << trappingRainwater(arr4) <<
std::endl;

    return 0;
}
```

Output:

```
Trapped water for arr1: 10
Trapped water for arr2: 7
Trapped water for arr3: 0
Trapped water for arr4: 5
```

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

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.

Answer:

```
#include <iostream>
#include <vector>
#include <algorithm>
#include <climits>

using namespace std;
```

```
int minDifference(const vector<int>& arr, int m) {
    int n = arr.size();
    if (m > n) return -1;

    vector<int> sortedArr = arr;
    sort(sortedArr.begin(), sortedArr.end());

    int minDiff = INT_MAX;
    for (int i = 0; i <= n - m; ++i) {
        int diff = sortedArr[i + m - 1] - sortedArr[i];
        minDiff = min(minDiff, diff);
    }

    return minDiff;
}

int main() {
    vector<int> arr1 = {7, 3, 2, 4, 9, 12, 56};
    int m1 = 3;
    cout << minDifference(arr1, m1) << endl;
    vector<int> arr2 = {7, 3, 2, 4, 9, 12, 56};
    int m2 = 5;
    cout << minDifference(arr2, m2) << endl;

    return 0;
}
```

Output:

```
2
7
```

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

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].

Answer:

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

vector<vector<int>> mergeIntervals(vector<vector<int>>& intervals) {
    if (intervals.empty()) return {};

    sort(intervals.begin(), intervals.end());

    vector<vector<int>> merged;
    merged.push_back(intervals[0]);

    for (int i = 1; i < intervals.size(); ++i) {
        if (merged.back()[1] >= intervals[i][0]) {
            merged.back()[1] = max(merged.back()[1], intervals[i][1]);
        } else {
            merged.push_back(intervals[i]);
        }
    }

    return merged;
}

int main() {
    vector<vector<int>> arr1 = {{1, 3}, {2, 4}, {6, 8}, {9, 10}};
```

```
    vector<vector<int>> arr2 = {{7, 8}, {1, 5}, {2, 4}, {4, 6}};

    vector<vector<int>> result1 = mergeIntervals(arr1);
    vector<vector<int>> result2 = mergeIntervals(arr2);

    for (const auto& interval : result1) {
        cout << "[" << interval[0] << ", " << interval[1] << "] ";
    }
    cout << endl;

    for (const auto& interval : result2) {
        cout << "[" << interval[0] << ", " << interval[1] << "] ";
    }
    cout << endl;

    return 0;
}
```

Output:

```
[1, 4] [6, 8] [9, 10]
[1, 6] [7, 8]
```

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

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}}

Answer:

```
#include <iostream>
#include <vector>

using namespace std;

void modifyMatrix(vector<vector<int>>& mat) {
    int M = mat.size();
    int N = mat[0].size();
    vector<bool> row(M, false), col(N, false);

    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;
            }
        }
    }
}

int main() {
    vector<vector<int>> mat1 = {{1, 0}, {0, 0}};
    vector<vector<int>> mat2 = {{0, 0, 0}, {0, 0, 1}};
    vector<vector<int>> mat3 = {{1, 0, 0, 1}, {0, 0, 1, 0}, {0, 0, 0, 0}};

    modifyMatrix(mat1);
    modifyMatrix(mat2);
    modifyMatrix(mat3);

    for (auto& row : mat1) {
        for (auto& cell : row) {
```

```
            cout << cell << " ";
        }
        cout << endl;
    }
    cout << endl;

    for (auto& row : mat2) {
        for (auto& cell : row) {
            cout << cell << " ";
        }
        cout << endl;
    }
    cout << endl;

    for (auto& row : mat3) {
        for (auto& cell : row) {
            cout << cell << " ";
        }
        cout << endl;
    }

    return 0;
}
```

Output:

```
1 1
1 0

0 0 1
1 1 1

1 1 1 1
1 1 1 1
1 0 1 1

Process returned 0 (0x0)   execution time : 0.099 s
Press any key to continue.
```

Time Complexity: O(m * n)
Space Complexity: O(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.

Output:

```
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(m * n)
Space Complexity: O(1)

QUESTION
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 = “())((())”
Output: Not Balanced

Answer:

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

int main(){
   //int n;
   //cout<<"Enter length: ";
   //cin>>n;
   //int arr[n];
   //for(int i=0;i<n;i++){
```

```
    //    cin>>arr[i];
    //}
    string s;
    cin>>s;

    int n=s.size();
    int flag=false;
    stack<int> st;
    for(int i=0;i<n;i++){
        if(s[i]=='(') st.push(s[i]);
        else {
            if(st.empty()){
                flag=!flag;
            }
            else st.pop();
        }
    }
    if(!flag && st.empty()){
        cout<<"Balanced";
    }
    else cout<<"Not balanced";
    return 0;
}
```

Output:

```
Balanced
Not Balanced
```

QUESTION:

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 = “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.

Answer:

```
#include <iostream>
#include <unordered_map>

using namespace std;

bool areAnagrams(const string& s1, const string& s2) {
    if (s1.size() != s2.size()) {
        return false;
    }

    unordered_map<char, int> freqMap;

    for (char ch : s1) {
        freqMap[ch]++;
    }

    for (char ch : s2) {
        if (freqMap.find(ch) == freqMap.end() || freqMap[ch] == 0) {
            return false;
        }
        freqMap[ch]--;
    }

    return true;
}

int main() {
    string s1 = "geeks";
    string s2 = "kseeg";

    cout << (areAnagrams(s1, s2) ? "true" : "false") << endl;
```

```
    s1 = "allergy";
    s2 = "allergic";
    cout << (areAnagrams(s1, s2) ? "true" : "false") << endl;

    s1 = "g";
    s2 = "g";
    cout << (areAnagrams(s1, s2) ? "true" : "false") << endl;

    return 0;
}
```

Output:

```
true
false
true
```

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

QUESTION
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”,
“eeksskee” etc.
But the substring “geeksskeeg” is the longest among all.
Input: str = “Geeks”
Output: “ee”
Input: str = “abc”
Output: “a”
Input: str = “”
Output: “”

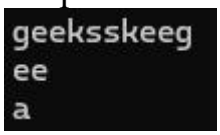
Answer:

```
#include <iostream>
#include <string>
```

```
using namespace std;

string longestPalindromicSubstring(const string& str) {
    int n = str.size();
    if (n == 0) return "";

    int start = 0, maxLength = 1;

    for (int i = 1; i < n; ++i) {
        int left = i - 1, right = i + 1;
        while (left >= 0 && right < n && str[left] == str[right]) {
            if (right - left + 1 > maxLength) {
                start = left;
                maxLength = right - left + 1;
            }
            --left;
            ++right;
        }
        left = i - 1, right = i;
        while (left >= 0 && right < n && str[left] == str[right]) {
            if (right - left + 1 > maxLength) {
                start = left;
                maxLength = right - left + 1;
            }
            --left;
            ++right;
        }
    }

    return str.substr(start, maxLength);
}

int main() {
    string str1 = "forgeeksskeegfor";
    string str2 = "Geeks";
    string str3 = "abc";
    string str4 = "";

    cout << longestPalindromicSubstring(str1) << endl;
    cout << longestPalindromicSubstring(str2) << endl;
    cout << longestPalindromicSubstring(str3) << endl;
    cout << longestPalindromicSubstring(str4) << endl;
    return 0;
```

}

Output:



Time Complexity : O(n^2)
Space Complexity : O(1)

QUESTION
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 string

Answer:

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;

string longestCommonPrefix(vector<string>& arr) {
    if (arr.empty()) return "-1";

    sort(arr.begin(), arr.end());

    string first = arr[0];
    string last = arr[arr.size() - 1];
    int n = min(first.size(), last.size());
```

```
    int i = 0;

    while (i < n && first[i] == last[i]) {
        i++;
    }

    return i > 0 ? first.substr(0, i) : "-1";
}

int main() {
    vector<string> arr1 = {"geeksforgeeks", "geeks", "geek", "geezer"};
    vector<string> arr2 = {"hello", "world"};

    cout << longestCommonPrefix(arr1) << endl;
    cout << longestCommonPrefix(arr2) << endl;

    return 0;
}
```

Output:

```
gee
-1
```

Time Complexity : O(n log n + k)
Space Complexity : O(1)

QUESTION
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]

Answer:

```
#include <iostream>
#include <stack>

using namespace std;
```

```
void deleteMiddleElement(stack<int>& s, int currentIndex, int middleIndex) {
    if (currentIndex == middleIndex) {
        s.pop();
        return;
    }

    int temp = s.top();
    s.pop();
    deleteMiddleElement(s, currentIndex + 1, middleIndex);
    s.push(temp);
}

void deleteMiddle(stack<int>& s) {
    int size = s.size();
    int middleIndex = size / 2;
    deleteMiddleElement(s, 0, middleIndex);
}

int main() {
    stack<int> s1;
    s1.push(1);
    s1.push(2);
    s1.push(3);
    s1.push(4);
    s1.push(5);

    deleteMiddle(s1);

    while (!s1.empty()) {
        cout << s1.top() << " ";
        s1.pop();
    }
    cout << endl;

    stack<int> s2;
    s2.push(1);
    s2.push(2);
    s2.push(3);
    s2.push(4);
    s2.push(5);
    s2.push(6);
```

```
    deleteMiddle(s2);

    while (!s2.empty()) {
        cout << s2.top() << " ";
        s2.pop();
    }
    cout << endl;

    return 0;
}
```

Output:

```
5 4 2 1
6 5 4 2 1
```

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

QUESTION
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
5 –> 25

2 –> 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 –> -1
7 –> 12
6 –> 12
12 –> -1

Explanation: 13 and 12 don‟t have any element greater than them present on the right
Side

Answer:

```
#include <iostream>
#include <stack>
#include <vector>

using namespace std;

void nextGreaterElement(const vector<int>& arr) {
    stack<int> s;
    vector<int> result(arr.size(), -1);

    for (int i = 0; i < arr.size(); ++i) {
        while (!s.empty() && arr[s.top()] < arr[i]) {
            result[s.top()] = arr[i];
            s.pop();
        }
        s.push(i);
    }

    for (int i = 0; i < arr.size(); ++i) {
        cout << arr[i] << " -> " << result[i] << endl;
    }
}

int main() {
    vector<int> arr1 = {4, 5, 2, 25};
    vector<int> arr2 = {13, 7, 6, 12};

    nextGreaterElement(arr1);
    cout << endl;
    nextGreaterElement(arr2);

    return 0;
}
```

Output:

```
4 -> 5
5 -> 25
2 -> 25
25 -> -1

13 -> -1
7 -> 12
6 -> 12
12 -> -1
```

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

QUESTION
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.

Answer:

```
#include <iostream>
#include <vector>
#include <queue>

using namespace std;

class TreeNode {
public:
    int val;
    TreeNode* left;
    TreeNode* right;

    TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
};

TreeNode* buildTree(const vector<int>& nodes) {
    if (nodes.empty() || nodes[0] == -1) return nullptr;

    TreeNode* root = new TreeNode(nodes[0]);
    queue<TreeNode*> q;
    q.push(root);
```

```
    int i = 1;
    while (i < nodes.size()) {
        TreeNode* current = q.front();
        q.pop();

        if (nodes[i] != -1) {
            current->left = new TreeNode(nodes[i]);
            q.push(current->left);
        }

        i++;

        if (i < nodes.size() && nodes[i] != -1) {
            current->right = new TreeNode(nodes[i]);
            q.push(current->right);
        }
        i++;
    }

    return root;
}

vector<int> rightView(TreeNode* root) {
    vector<int> result;
    if (!root) return result;

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

    while (!q.empty()) {
        int levelSize = q.size();
        for (int i = 0; i < levelSize; ++i) {
            TreeNode* node = q.front();
            q.pop();
            if (i == levelSize - 1) {
                result.push_back(node->val);
            }
            if (node->left) q.push(node->left);
            if (node->right) q.push(node->right);
        }
    }

    return result;
```

```
}

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

    vector<int> nodes(n);
    cout << "Enter tree nodes in level order (-1 for null nodes): ";
    for (int i = 0; i < n; ++i) {
        cin >> nodes[i];
    }

    TreeNode* root = buildTree(nodes);

    vector<int> rightViewNodes = rightView(root);
    cout << "Right View: ";
    for (int val : rightViewNodes) {
        cout << val << " ";
    }
    cout << endl;

    return 0;
}
```

Output:

```
Enter number of nodes in the tree: 7
Enter tree nodes in level order (-1 for null nodes): 1 2 3 -1 4 -1 5
Right View: 1 3 5
```

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

QUESTION
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.
Answer:

```
#include <iostream>
#include <queue>
```

```
#include <vector>
#include <string>
#include <sstream>

using namespace std;

class TreeNode {
public:
    int val;
    TreeNode* left, * right;
    TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}
};

TreeNode* buildTree(const vector<string>& values) {
    if (values.empty() || values[0] == "-1") return nullptr;

    TreeNode* root = new TreeNode(stoi(values[0]));
    queue<TreeNode*> q;
    q.push(root);

    int i = 1;
    while (i < values.size()) {
        TreeNode* current = q.front();
        q.pop();

        // Assign left child
        if (values[i] != "-1") {
            current->left = new TreeNode(stoi(values[i]));
            q.push(current->left);
        }
        cout << "Node " << current->val << " left child: " << (current->left ?
to_string(current->left->val) : "null") << endl;

        i++;

        // Assign right child if available
        if (i < values.size() && values[i] != "-1") {
            current->right = new TreeNode(stoi(values[i]));
            q.push(current->right);
        }
        cout << "Node " << current->val << " right child: " << (current-
>right ? to_string(current->right->val) : "null") << endl;
        i++;
```

```
    }

    return root;
}

int maxDepth(TreeNode* root) {
    if (root == nullptr) return 0;
    int leftDepth = maxDepth(root->left);
    int rightDepth = maxDepth(root->right);

    return max(leftDepth, rightDepth) + 1;
}

int main() {
    string line;
    cout << "Enter values in level order (use -1 for null nodes): ";
    getline(cin, line);

    stringstream ss(line);
    vector<string> values;
    string temp;
    while (ss >> temp) {
        values.push_back(temp);
    }

    TreeNode* root = buildTree(values);
    cout << "The height of the tree is: " << maxDepth(root) << endl;

    return 0;
}
```

Output: