DOMAIN WINTER CAMP

Department of Computer Science and Engineering

Name:- Aman Kumar Pal UID:- 22BCS10188 Section:- 22KPIT-901-A

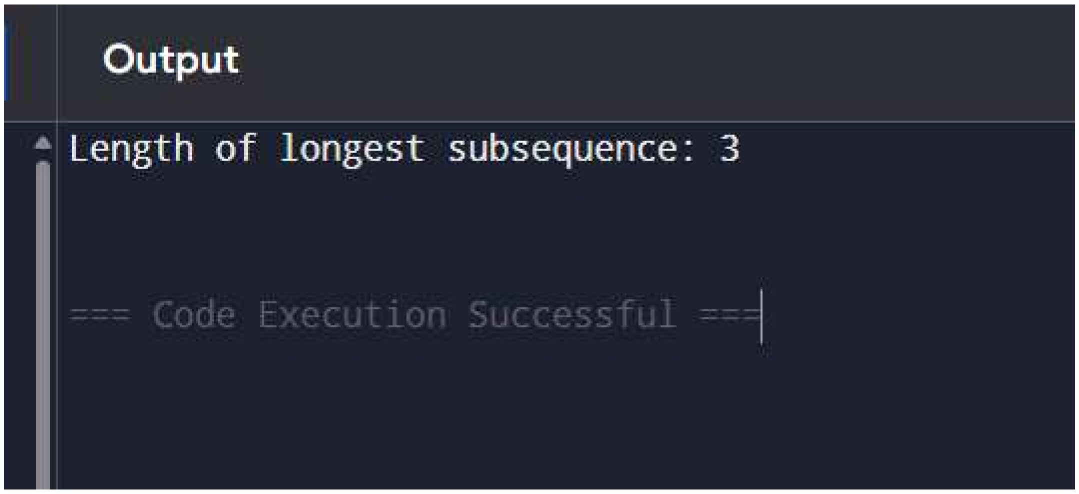
DAY-4

Q.1. You are given an integer array nums and an integer k. Find the longest subsequence of nums that meets the following requirements: • The subsequence is strictly increasing and • The difference between adjacent elements in the subsequence is at most k. Return the length of the longest subsequence that meets the requirements. A subsequence is an array that can be derived from another array by deleting some or no elements without changing the order of the remaining elements. using cpp

Program Code:-

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

int longestSubsequence(vector<int>& nums, int k) {
    map<int, int> dp; // dp[value] stores the length of the longest subsequence ending with value
    int maxLength = 0;

    for (int num : nums) {
        // Check for possible predecessors within the range [num - k, num - 1]
        int maxPrevLength = 0;
        for (int i = num - k; i < num; ++i) {
            if (dp.find(i) != dp.end()) {
                maxPrevLength = max(maxPrevLength, dp[i]);
            }
        }

        dp[num] = maxPrevLength + 1; // Update the current number's subsequence length
        maxLength = max(maxLength, dp[num]); // Update the global maximum length
    }

    return maxLength;
}

int main() {
    vector<int> nums = {3, 10, 2, 1, 20};
    int k = 10;
    cout << "Length of longest subsequence: " << longestSubsequence(nums, k) << endl;

    return 0;
}
```

Output:-



```
Length of longest subsequence: 3


=== Code Execution Successful ===
```

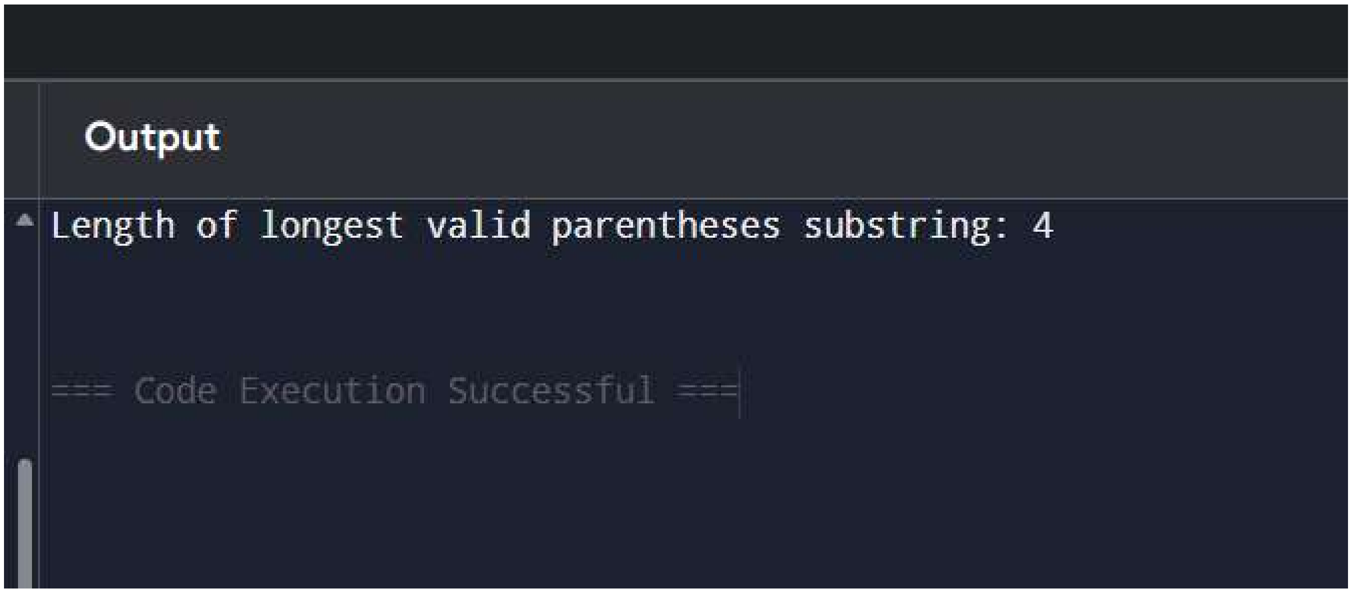
Q.2.

Given a string containing just the characters '(' and ')', return the length of the longest valid (well-formed) parentheses substring. using cpp

Program Code:-

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

int longestValidParentheses(string s) {
   stack<int> stk;
   stk.push(-1); // Base for calculating lengths
   int maxLength = 0;

   for (int i = 0; i < s.length(); ++i) {
      if (s[i] == '(') {
         stk.push(i);
      } else {
         stk.pop();
         if (stk.empty()) {
            stk.push(i);
         } else {
            maxLength = max(maxLength, i - stk.top());
         }
      }
   }

   return maxLength;
}

int main() {
   string s = "(()))())(";
   cout << "Length of longest valid parentheses substring: " <<
longestValidParentheses(s) << endl;

   return 0;
}
```

Output:-


Output

```
Length of longest valid parentheses substring: 4


=== Code Execution Successful ===
```

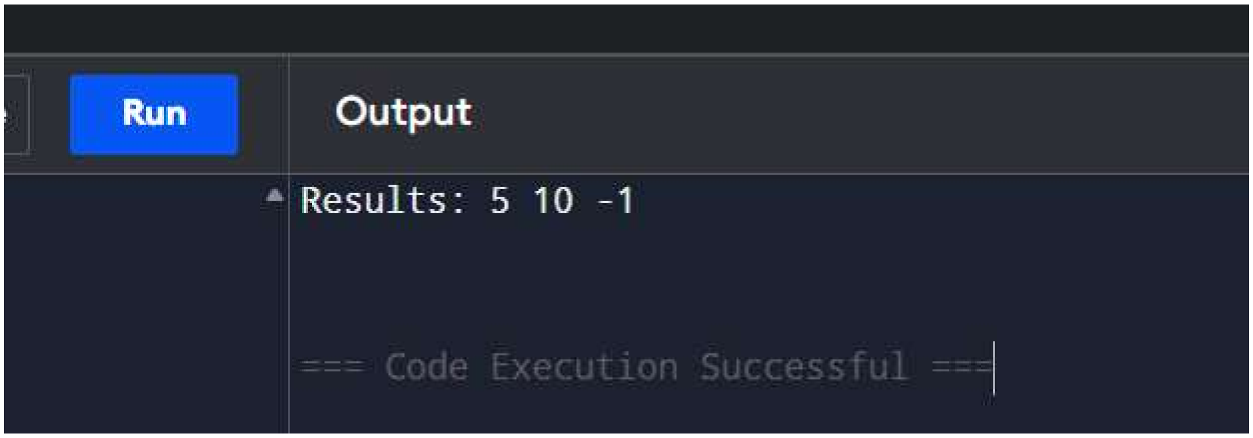
Q.3.

You are given an integer array nums of length n and an integer array queries. Let gcdPairs denote an array obtained by calculating the GCD of all possible pairs (nums[i], nums[j]), where 0 <= i < j < n, and then sorting these values in ascending order. For each query queries[i], you need to find the element at index queries[i] in gcdPairs. Return an integer array answer, where answer[i] is the value at gcdPairs[queries[i]] for each query. The term gcd(a, b) denotes the greatest common divisor of a and b.

Program code:-

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

vector<int> gcdQuery(vector<int>& nums, vector<int>& queries) {
    vector<int> gcdPairs;
    int n = nums.size();

    // Generate all GCD pairs
    for (int i = 0; i < n; ++i) {
        for (int j = i + 1; j < n; ++j) {
            gcdPairs.push_back(gcd(nums[i], nums[j]));
        }
    }

    // Sort the GCD pairs
    sort(gcdPairs.begin(), gcdPairs.end());

    // Answer the queries
    vector<int> answer;
    for (int q : queries) {
        if (q < gcdPairs.size()) {
            answer.push_back(gcdPairs[q]);
        } else {
            answer.push_back(-1); // Handle out-of-range queries
        }
    }

    return answer;
}

int main() {
    vector<int> nums = {10, 15, 20};
    vector<int> queries = {0, 2, 5};

    vector<int> result = gcdQuery(nums, queries);

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

    return 0;
}
```

Output:-



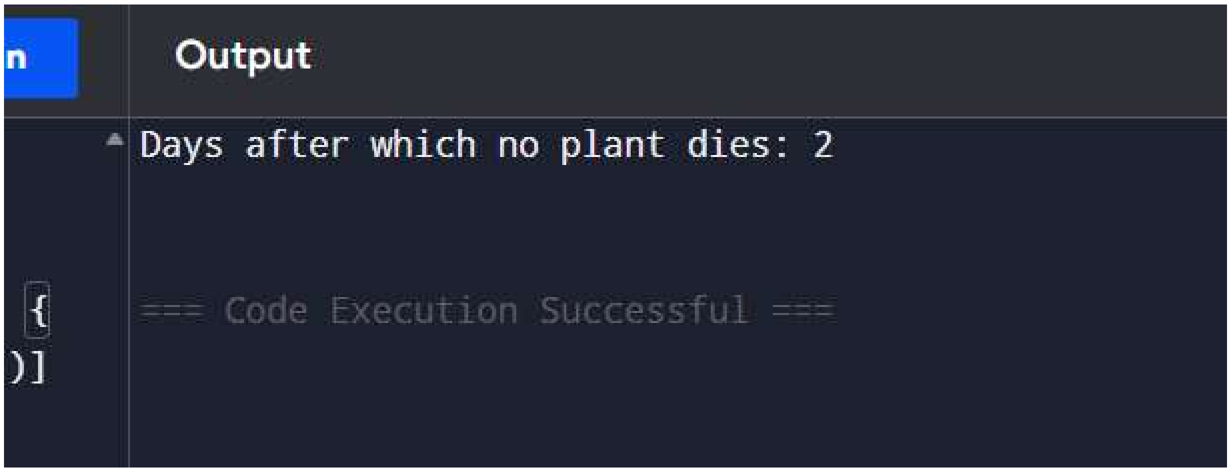
```
Results: 5 10 -1


=== Code Execution Successful ===
```

Q.4. Suppose there is a circle. There are N petrol pumps on that circle. Petrol pumps are numbered 0 to (N-1) (both inclusive). You have two pieces of information corresponding to each of the petrol pump: (1) the amount of petrol that particular petrol pump will give, and (2) the distance from that petrol pump to the next petrol pump.

Program Code:-

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

int poisonousPlants(vector<int>& plants) {
   int n = plants.size();
   vector<int> days(n, 0); // Days each plant survives
   stack<int> stk; // Monotonic stack to track indices
   int maxDays = 0;

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

      // Pop elements from the stack while the current plant has less pesticide
      while (!stk.empty() && plants[stk.top()] >= plants[i]) {
         maxSurviveDays = max(maxSurviveDays, days[stk.top()]);
         stk.pop();
      }

      // If the stack is not empty, the current plant dies after surviving
maxSurviveDays + 1 days
      if (!stk.empty()) {
         days[i] = maxSurviveDays + 1;
      }

      stk.push(i); // Push current plant index to the stack
      maxDays = max(maxDays, days[i]); // Update the maximum days
   }

   return maxDays;
}

int main() {
   vector<int> plants = {6, 5, 8, 4, 7, 10, 9};
   cout << "Days after which no plant dies: " << poisonousPlants(plants) <<
endl;
   return 0;
}
```

Output:-



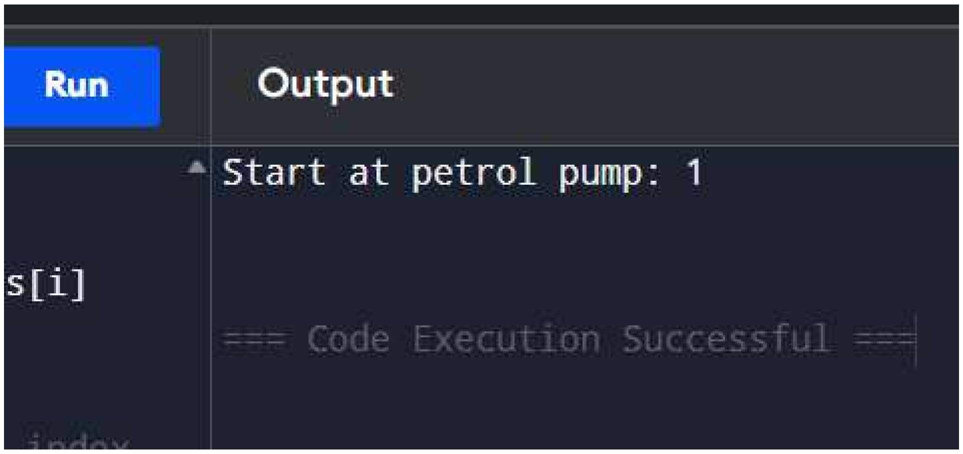
```
Days after which no plant dies: 2


=== Code Execution Successful ===
```

Q.5. You are given an array of integers nums, there is a sliding window of size k which is moving from the very left of the array to the very right. You can only see the k numbers in the window. Each time the sliding window moves right by one position. Return the max sliding window.

Program Code:-

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

int findStartingPetrolPump(vector<pair<int, int>>& petrolPumps) {
    int totalPetrol = 0, totalDistance = 0;
    int currentPetrol = 0, startIndex = 0;

    for (int i = 0; i < petrolPumps.size(); ++i) {
        totalPetrol += petrolPumps[i].first;
        totalDistance += petrolPumps[i].second;
        currentPetrol += petrolPumps[i].first - petrolPumps[i].second;

        // If current petrol becomes negative, reset start index
        if (currentPetrol < 0) {
            startIndex = i + 1;
            currentPetrol = 0;
        }
    }

    // Check if the tour is possible
    return (totalPetrol >= totalDistance) ? startIndex : -1;
}

int main() {
    vector<pair<int, int>> petrolPumps = {{4, 6}, {6, 5}, {7, 3}, {4, 5}};
    int startIndex = findStartingPetrolPump(petrolPumps);

    if (startIndex != -1) {
        cout << "Start at petrol pump: " << startIndex << endl;
    } else {
        cout << "No solution possible." << endl;
    }

    return 0;
}
```

Output:-



```
s[i]
index
```

```
Start at petrol pump: 1


=== Code Execution Successful ===
```