CS-218 Entrance Exam

Shah Akshat Netid - ashah004, Sid - 862477556

1 Problem 1 - The Stairs [Submission ID - 313695560]

In this code, we first iterate through the sequence of integers representing steps and calculate differences between consecutive steps. Then we count the frequency of these differences using a map. We get the most frequent difference. Now, if we get only two differences and the first difference is not the most frequent one, it outputs 1. Otherwise, it checks for the first position where the difference deviates from the most frequent one and outputs that step.

2 Problem 2 - ABC String [Submission ID - 313313565]]

In this code, we use recursion to calculate the characters from the defined string structure. The catch here is to stop when we find the characters in which our sub-string lies. First, we calculate the length of the string at the particular k level. Then, for each case, we find the character between l and r by recursively iterating the part of the string where the character belongs to. The string is constructed, and it follows a pattern where each level is built by repeating the previous string and adding 'A', 'B', 'C' at specific positions.

3 Problem 3 - Easy Sorting [Submission ID - 313529048]

In this code, we first find the number of occurrences of each candy type. Then, we get three ideal positions where each candy type should have been placed. We calculate how many candies of type 2 and 3 are misplaced in each section and then calculate the minimum swaps required to put them in their correct places. Then we check if we still have any mismatches remaining and swap them to their correct positions and print the minimum swaps required.

4 Problem 4 - Longest Unimodal Subsequence [[Submission ID - 313645036]

In this code, we first calculate the longest increasing subsequence for each element and store it in one vector. Then we calculate the longest decreasing subsequence that starts from each element. At last, we add the LIS and LDS value for each element while subtracting 1 for counting elements twice. And the maximum value we get is the longest unimodal subsequence.

5 Problem 5 - Black Family Tree [Subsequence [[Submission ID -313996134]

In this code, we first build the tree based on given parent-child relationships along with marking the traitor nodes. Then for each root node, we perform BFS to get the largest connected sub-tree that doesn't include traitor nodes. We keep the max ne stored and return it once we have iterated through all the non-traitor nodes.

6 Problem 6 - Journalist [Subsequence [[Submission ID -313869710]

In this particular problem, we first find the shortest path from each venue to all the other venues using Dijkstra's algorithm and then find the maximum distance among all the venue pairs. Then we calculate the minimum speed with the maximum distance that we got divided by the time provided to us.