Data Structures [Day 1]

UID: 24MCI10204

Name: Rahul Saxena

Branch: 24MCA – AI & ML

Question 1: You are working as a data analyst for a healthcare company that monitors real-time vital signs from two different types of wearable sensors worn by patients—Sensor A and Sensor B. Both sensors continuously record patient health scores at fixed intervals, and the data is sorted in increasing order as it's time-based.

At the end of the day, the company wants to generate a single median health score for each patient to summarize their condition for doctors. However, since both sensors are accurate but have slightly different sampling rates, the number of entries in the two arrays can vary.

You are asked to write a C/C++ program that takes the two sorted arrays of health scores collected from Sensor A and Sensor B and computes the median of the combined dataset, without fully merging them, in O(log(min(n, m))) time complexity.

Code:

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
double findMedianHealthScore(const vector<int>& sensorA, const vector<int>& sensorB) {
   if (sensorA.size() > sensorB.size())
      return findMedianHealthScore(sensorB, sensorA);
   int n1 = sensorA.size();
   int n2 = sensorB.size();
   int low = 0, high = n1;
   while (low <= high) {
      int cut1 = (low + high) / 2;
      int cut2 = (n1 + n2 + 1) / 2 - cut1;
      int left1 = (cut1 == 0) ? INT_MIN : sensorA[cut1 - 1];
      int left2 = (cut2 == 0) ? INT_MIN : sensorB[cut2 - 1];

      int right1 = (cut1 == n1) ? INT_MAX : sensorA[cut1];
      int right2 = (cut2 == n2) ? INT_MAX : sensorB[cut2];
      if (left1 <= right2 && left2 <= right1) {
         if ((n1 + n2) % 2 == 0) {
            return (max(left1, left2) + min(right1, right2)) / 2.0;
         } else {
            return max(left1, left2);
         }
      }
      else if (left1 > right2) {\
         high = cut1 - 1;
      }
      else {
         low = cut1 + 1;
      }
   }
   return -1.0;
```

```
}
int main() {
   vector<int> sensorA = {72, 75, 80, 85};
   vector<int> sensorB = {70, 78, 82, 88, 90};
   double median = findMedianHealthScore(sensorA, sensorB);
   cout << "Median health score: " << median << endl;
   return 0;
}
```

Question 2: You are building a feature for a text-processing application that uses a doubly linked list to store characters of a string in memory. As part of a new feature, you need to check if the content stored in the list forms a palindrome — that is, it reads the same forward and backward.

Your task is to write a C/C++ program that takes a doubly linked list of integers or characters as input and determines whether it is a palindrome.

Code:

```
#include <iostream>
using namespace std;
struct Node {
   char data;
   Node* prev;
   Node* next;
   Node(char value) {
      data = value;
      prev = nullptr;
      next = nullptr;
   }
};
void insertEnd(Node*& head, char value) {
   Node* newNode = new Node(value);
   if (head == nullptr) {
      head = newNode;
      return;
   }
   Node* temp = head;
   while (temp->next != nullptr) {
      temp = temp->next;
   }
   temp->next = newNode;
   newNode->prev = temp;
}
bool isPalindrome(Node* head) {
   if (head == nullptr || head->next == nullptr)
      return true;
   Node* left = head;
   Node* right = head;
   while (right->next != nullptr) {
      right = right->next;
   }
   while (left != nullptr && right != nullptr && left != right && right->next != left) {
      if (left->data != right->data)
```

```
            return false;
        left = left->next;
        right = right->prev;
    }
    return true;
}
int main() {
    Node* head = nullptr;
    string input;
    cout << "Enter string: ";
    cin >> input;
    for (char ch : input) {
        insertEnd(head, ch);
    }
    if (isPalindrome(head)) {
        cout << "The list is a palindrome." << endl;
    } else {
        cout << "The list is NOT a palindrome." << endl;
    }
    return 0;
}
```