DSA ASSIGNMENT - JUNE 24

Name – Vijay Kumar Saini
Roll.No - 23/11/EC/027
LeetCode Id - vijay_kumar_saini
GeekforGeeks - sainivij3yo0
Github Repo Link: https://github.com/vijaysaini2613/jnu-ass

1. Design a Singly Linked List and write functions to
○ Insert at beginning
○ Insert at end
○ Insert at any index
○ Delete at beginning
○ Delete at end

○ Delete at index

○ Delete first occurrence of node with given value
○ Write at least 2 test cases for each operation

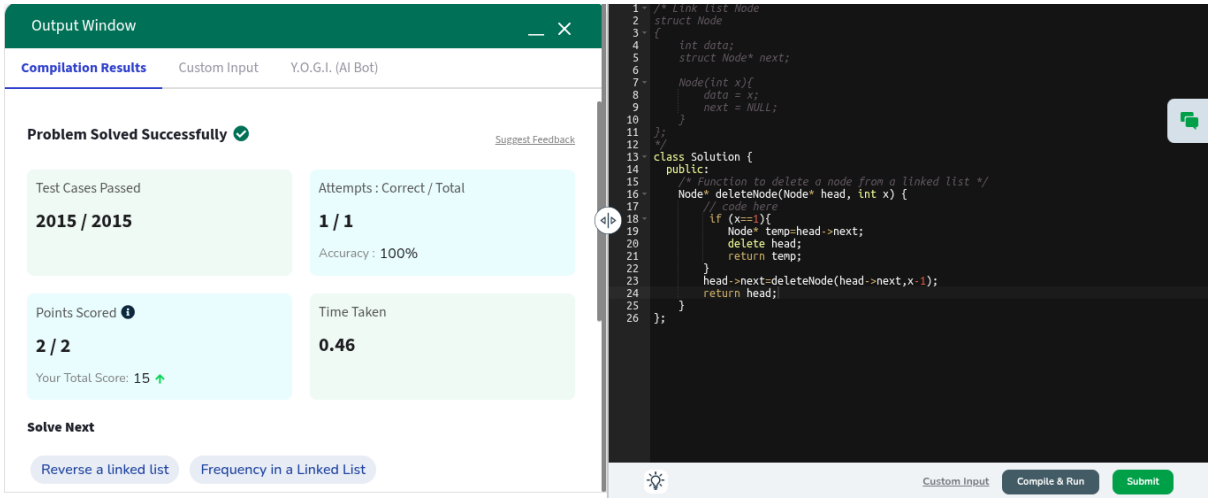
```
#include <iostream>
using namespace std;

// Node class to store data and a pointer to the next node
class Node {
public:
    int data;
    Node* next;

    Node(int val) {
        data = val;
        next = NULL;
    }
};

// Linked List class with basic operations
class SinglyLinkedList {
private:
    Node* head;

public:
    // Constructor: initially the list is empty
    SinglyLinkedList() {
        head = NULL;
    }

    // Insert a node at the beginning
    void insertAtBeginning(int val) {
        Node* newNode = new Node(val);
        newNode->next = head;
        head = newNode;
    }

    // Insert a node at the end
    void insertAtEnd(int val) {
        Node* newNode = new Node(val);
        if (head == NULL) {
            head = newNode;
            return;
        }
        Node* temp = head;
        while (temp->next != NULL) {
            temp = temp->next;
        }
```

```
    temp->next = newNode;
}

// Insert a node at a specific index
void insertAtIndex(int index, int val) {
    if (index == 0) {
        insertAtBeginning(val);
        return;
    }
    Node* newNode = new Node(val);
    Node* temp = head;
    for (int i = 0; i < index - 1 && temp != NULL; i++) {
        temp = temp->next;
    }
    if (temp == NULL) {
        cout << "Index out of range\n";
        return;
    }
    newNode->next = temp->next;
    temp->next = newNode;
}

// Delete the first node
void deleteAtBeginning() {
    if (head == NULL) {
        cout << "List is empty\n";
        return;
    }
    Node* temp = head;
    head = head->next;
    delete temp;
}

// Delete the last node
void deleteAtEnd() {
    if (head == NULL) {
        cout << "List is empty\n";
        return;
    }
    if (head->next == NULL) {
        delete head;
        head = NULL;
        return;
    }
    Node* temp = head;
    while (temp->next->next != NULL) {
        temp = temp->next;
    }
```

```
    delete temp->next;
    temp->next = NULL;
}

// Delete node at a specific index
void deleteAtIndex(int index) {
    if (head == NULL) {
        cout << "List is empty\n";
        return;
    }
    if (index == 0) {
        deleteAtBeginning();
        return;
    }
    Node* temp = head;
    for (int i = 0; i < index - 1 && temp != NULL; i++) {
        temp = temp->next;
    }
    if (temp == NULL || temp->next == NULL) {
        cout << "Index out of range\n";
        return;
    }
    Node* toDelete = temp->next;
    temp->next = toDelete->next;
    delete toDelete;
}

// Delete first node with given value
void deleteByValue(int val) {
    if (head == NULL) {
        cout << "List is empty\n";
        return;
    }
    if (head->data == val) {
        deleteAtBeginning();
        return;
    }
    Node* temp = head;
    while (temp->next != NULL && temp->next->data != val) {
        temp = temp->next;
    }
    if (temp->next == NULL) {
        cout << "Value not found\n";
        return;
    }
    Node* toDelete = temp->next;
    temp->next = toDelete->next;
    delete toDelete;
```

```
    }

    // Display the list
    void display() {
        Node* temp = head;
        while (temp != NULL) {
            cout << temp->data << " ";
            temp = temp->next;
        }
        cout << endl;
    }
};

int main() {
    SinglyLinkedList list;

    list.insertAtBeginning(10);
    list.insertAtBeginning(20);
    list.display(); // 20 10

    list.insertAtEnd(30);
    list.insertAtEnd(40);
    list.display(); // 20 10 30 40

    list.insertAtIndex(2, 25);
    list.display(); // 20 10 25 30 40

    list.deleteAtBeginning();
    list.display(); // 10 25 30 40

    list.deleteAtEnd();
    list.display(); // 10 25 30

    list.deleteAtIndex(1);
    list.display(); // 10 30

    list.deleteByValue(30);
    list.display(); // 10

    return 0;
}
```

2. Delete a Node in Singly Linked List
○ Platform: GeeksforGeeks
○ Link:
https://www.geeksforgeeks.org/problems/delete-a-node-in-single-linked-list/1



3. Find Middle

○ Platform: LeetCode

○ Link:https://leetcode.com/problems/middle-of-the-linked-list/description/