MAIN FUNCTION TO DRIVE BOTH CODE:

```
int main() {
    cout << "\n--- Singly Linked List Operations ---\n";
    addNode(10); addNode(20); addNode(30);
    displayList();
    deleteNode(20);
    displayList();

    cout << "\n--- Doubly Linked List Operations ---\n";
    addDNode(100); addDNode(200); addDNode(300);
    displayDList();
    deleteDNode(200);
    displayDList();

    return 0;
}
```

SINGLY LINKED LIST:

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

struct Node {
    int data;
    Node* next;
};

Node* head = nullptr;

void addNode(int value) {
    Node* newNode = new Node{value, nullptr};
    if (!head) {
        head = newNode;
    } else {
        Node* temp = head;
        while (temp->next) temp = temp->next;
        temp->next = newNode;
    }
}

void displayList() {
    Node* temp = head;
    cout << "Singly Linked List: ";
    while (temp) {
        cout << temp->data << " -> ";
        temp = temp->next;
    }
    cout << "NULL\n";
}

void deleteNode(int value) {
    Node* temp = head;
    Node* prev = nullptr;

    while (temp && temp->data != value) {
        prev = temp;
        temp = temp->next;
    }

    if (!temp) {
```

```
        cout << "Value not found!\n";
        return;
    }

    if (!prev) head = head->next;
    else prev->next = temp->next;

    delete temp;
    cout << "Deleted value: " << value << endl;
}
```

DOUBLY LINKED LIST:

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

struct DNode {
    int data;
    DNode* prev;
    DNode* next;
};

DNode* headD = nullptr;

void addDNode(int value) {
    DNode* newNode = new DNode{value, nullptr, nullptr};
    if (!headD) {
        headD = newNode;
    } else {
        DNode* temp = headD;
        while (temp->next) temp = temp->next;
        temp->next = newNode;
        newNode->prev = temp;
    }
}

void displayDList() {
    DNode* temp = headD;
    cout << "Doubly Linked List: ";
    while (temp) {
        cout << temp->data << " <-> ";
        temp = temp->next;
    }
    cout << "NULL\n";
}

void deleteDNode(int value) {
    DNode* temp = headD;

    while (temp && temp->data != value) {
        temp = temp->next;
    }

    if (!temp) {
        cout << "Value not found!\n";
        return;
    }

    if (temp->prev) temp->prev->next = temp->next;
```

```
    else headD = temp->next;

    if (temp->next) temp->next->prev = temp->prev;

    delete temp;
    cout << "Deleted value: " << value << endl;
}
```