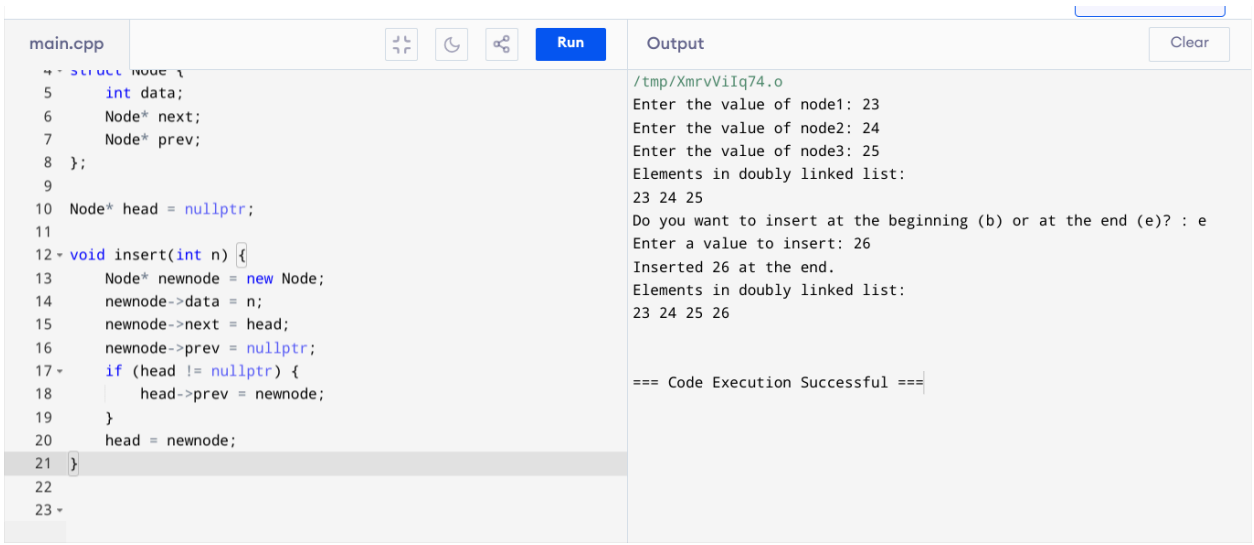
Zainab Shahzad
56108
DSA Lab 5

Task 1:

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

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

Node* head = nullptr;

void insert(int n) {
    Node* newnode = new Node;
    newnode->data = n;
    newnode->next = head;
    newnode->prev = nullptr;
    if (head != nullptr) {
        head->prev = newnode;
    }
    head = newnode;
}

void insertAtEnd(int n) {
    Node* newnode = new Node;
    newnode->data = n;
    newnode->next = nullptr;
    if (head == nullptr) {

        newnode->prev = nullptr;
        head = newnode;
        return;
    }


    Node* temp = head;
    while (temp->next != nullptr) {
        temp = temp->next;
    }
```

```
    temp->next = newnode;
    newnode->prev = temp;
}

void print() {
    cout << "Elements in doubly linked list:\n";
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}

int main() {

    Node node1, node2, node3;
    cout << "Enter the value of node1: ";
    cin >> node1.data;
    node1.prev = nullptr;
    node1.next = &node2;

    cout << "Enter the value of node2: ";
    cin >> node2.data;
    node2.prev = &node1;
    node2.next = &node3;

    cout << "Enter the value of node3: ";
    cin >> node3.data;
    node3.prev = &node2;
    node3.next = nullptr;

    head = &node1;
    print();


        cout << "Do you want to insert at the beginning (b) or at the end (e)? : ";
        char choice;
        cin >> choice;

        cout << "Enter a value to insert: ";
        int value;
        cin >> value;
```

```
        if (choice == 'b') {
            insert(value);
            cout << "Inserted " << value << " at the beginning.\n";
            print();
        } else if (choice == 'e') {
            insertAtEnd(value);
            cout << "Inserted " << value << " at the end.\n";
            print();
        } else {
            cout << "Invalid choice! Please enter 'b' for beginning, 'e' for end.\n";


        print();
    }

    return 0;
}
```



Task 2:

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

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

Node* head = nullptr;

void insertAtEnd(int n) {
    Node* newnode = new Node;
    newnode->data = n;
    newnode->next = nullptr;

    if (head == nullptr) {
        newnode->prev = nullptr;
        head = newnode;
        return;
    }

    Node* temp = head;
    while (temp->next != nullptr) {
        temp = temp->next;
    }

    temp->next = newnode;
    newnode->prev = temp;
}

void deleteAtStart() {
    if (head == nullptr) {
        cout << "List is empty. Nothing to delete.\n";
        return;
    }

    Node* temp = head;
    head = head->next;

    if (head != nullptr) {
        head->prev = nullptr;
    }
```

```
    delete temp;
    cout << "Deleted the node from the beginning.\n";
}

void deleteAtEnd() {
    if (head == nullptr) {
        cout << "List is empty. Nothing to delete.\n";
        return;
    }

    Node* temp = head;
    while (temp->next != nullptr) {
        temp = temp->next;
    }

    if (temp == head) {
        delete head;
        head = nullptr;
    } else {
        temp->prev->next = nullptr;
        delete temp;
    }

    cout << "Deleted the node from the end.\n";
}

void print() {
    cout << "Elements in doubly linked list:\n";
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}

int main() {

    int value;
    for (int i = 1; i <= 3; i++) {
        cout << "Enter the value of node " << i << ": ";
        cin >> value;
        insertAtEnd(value);
    }
```

```
    print();

    while (true) {
        cout << "Do you want to delete from the beginning (b) or from the end (e)?  ";
        char choice;
        cin >> choice;

        if (choice == 'b') {
            deleteAtStart();

        } else if (choice == 'e') {
            deleteAtEnd();

        } else {
            cout << "Invalid choice! Please enter 'b' for beginning or 'e' for end.\n";

        print();
    }


    while (head != nullptr) {
        deleteAtStart();

    }

    return 0;
}
```