**Assignment No. 2**

**Submitted to:**

Mr. Azib Mehmood

**Submitted by:**

Maria Inam

**Roll No:**

22011556-100

**Section :**

22-B

**Program**

#include <iostream>

using namespace std;

struct Node {

int data;

Node\* next;

};

class LinkedList {

private:

Node\* head;

public:

LinkedList() : head(nullptr) {}

bool search(int key) {

Node\* current = head;

while (current != nullptr) {

if (current->data == key) {

return true;

}

current = current->next;

}

return false;

}

void update(int position, int newValue) {

Node\* current = head;

for (int i = 1; i < position && current != nullptr; ++i) {

current = current->next;

}

if (current != nullptr) {

current->data = newValue;

} else {

cout << "Position out of bounds." << endl;

}

}

void insert(int position, int value) {

Node\* newNode = new Node{value, nullptr};

if (position == 1) {

newNode->next = head;

head = newNode;

} else {

Node\* current = head;

for (int i = 2; i < position && current != nullptr; ++i) {

current = current->next;

}

if (current != nullptr) {

newNode->next = current->next;

current->next = newNode;

} else {

cout << "Position out of bounds." << endl;

}

}

}

void deleteFromBeginning() {

if (head != nullptr) {

Node\* temp = head;

head = head->next;

delete temp;

} else {

cout << "List is empty. Cannot delete from the beginning." << endl;

}

}

void deleteFromEnd() {

if (head != nullptr) {

if (head->next == nullptr) {

delete head;

head = nullptr;

} else {

Node\* current = head;

while (current->next->next != nullptr) {

current = current->next;

}

delete current->next;

current->next = nullptr;

}

} else {

cout << "List is empty. Cannot delete from the end." << endl;

}

}

void deleteAtPosition(int position) {

if (head != nullptr) {

if (position == 1) {

Node\* temp = head;

head = head->next;

delete temp;

} else {

Node\* current = head;

for (int i = 2; i < position && current->next != nullptr; ++i) {

current = current->next;

}

if (current->next != nullptr) {

Node\* temp = current->next;

current->next = current->next->next;

delete temp;

} else {

cout << "Position out of bounds." << endl;

}

}

} else {

cout << "List is empty. Cannot delete from the specified position." << endl;

}

}

void reverse() {

Node\* prev = nullptr;

Node\* current = head;

Node\* next = nullptr;

while (current != nullptr) {

next = current->next;

current->next = prev;

prev = current;

current = next;

}

head = prev;

}

void display() {

Node\* temp = head;

while (temp != nullptr) {

cout << temp->data << " ";

temp = temp->next;

}

cout << endl;

}

};

int main() {

LinkedList myList;

myList.insert(1, 10);

myList.insert(2, 20);

myList.insert(3, 30);

cout << "Linked List: ";

myList.display();

cout << "Search for 20: " << (myList.search(20) ? "Found" : "Not found") << endl;

myList.update(2, 25);

cout << "Linked List after update at position 2: ";

myList.display();

myList.deleteFromBeginning();

cout << "Linked List after deleting from the beginning: ";

myList.display();

myList.deleteFromEnd();

cout << "Linked List after deleting from the end: ";

myList.display();

myList.deleteAtPosition(1);

cout << "Linked List after deleting from position 1: ";

myList.display();

myList.insert(1, 40);

myList.insert(2, 50);

myList.insert(3, 60);

cout << "Linked List before reverse: ";

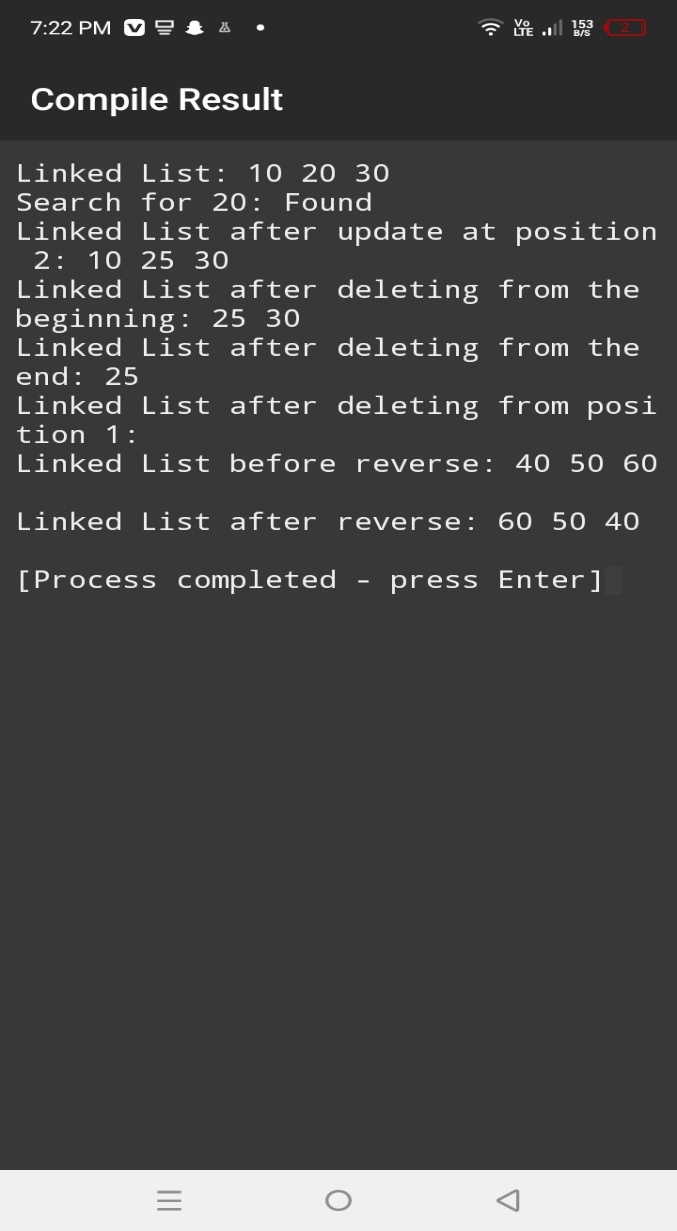
myList.display();

myList.reverse();

cout << "Linked List after reverse: ";

myList.display();

return 0;

}