Assignment No # 02

Submitted To:

Mr. Azib Mehmood

Submitted By:

Sania Ali

Roll No:

22011556-101

Section:

IT (22-B)

Course Title:

Data Structure And Algorithms

Linked List

Program:-

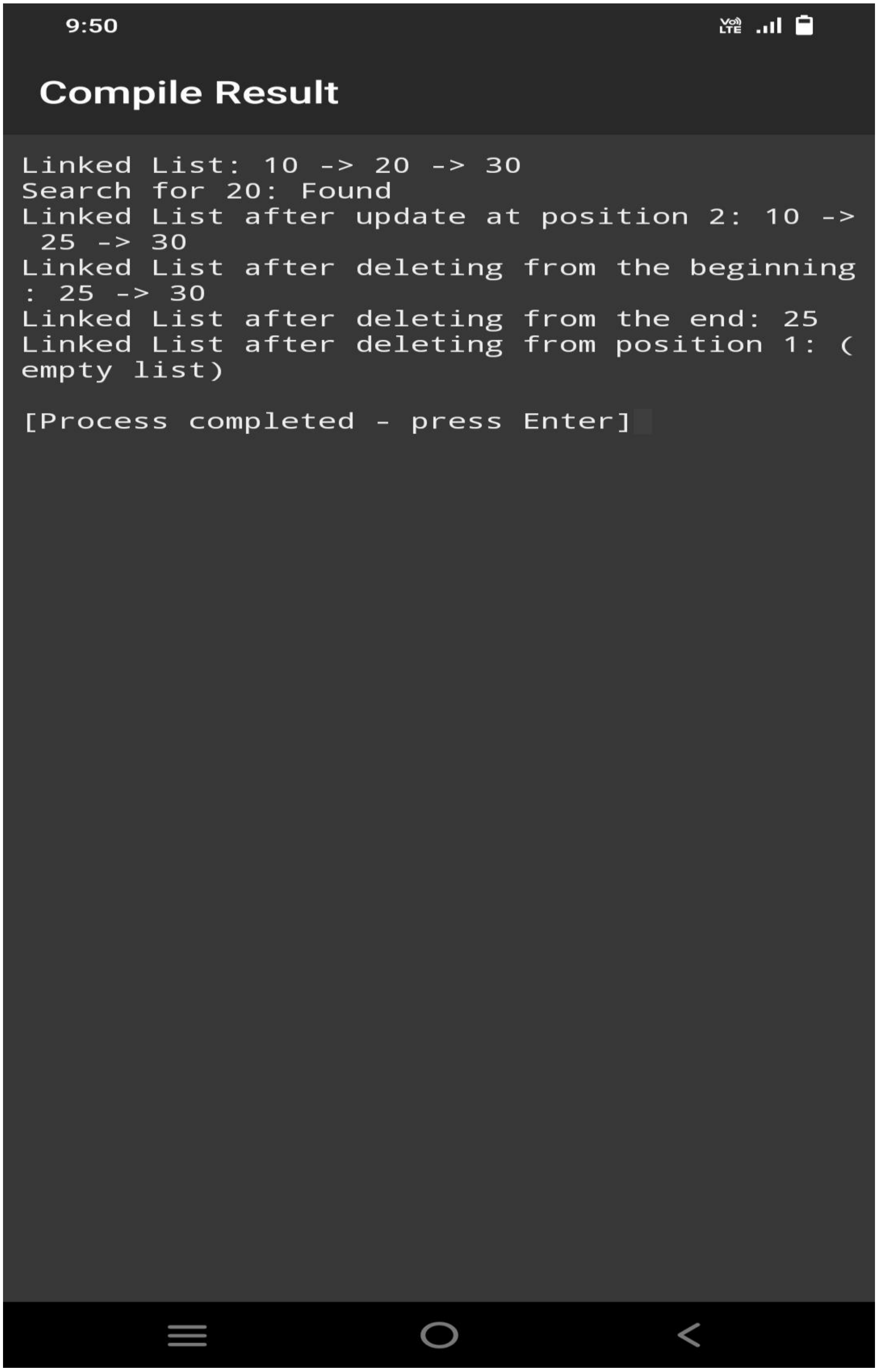
```
#include <iostream>
using namespace std;
struct Node {
    int data;
    Node* next;
};
class LinkedList {
private:
    Node* head;
public:
    // Constructor
    LinkedList() : head(nullptr) {}
    // Function to search for an element in the linked list
    bool search(int key) {
        Node* current = head;
        while (current != nullptr) {
            if (current->data == key) {
                return true; // Element found
            }
            current = current->next;
        }
        return false; // Element not found
    }
    // Function to update the element at any position
    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;
        }
    }
    // Function to insert an element at any position
    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;
            }
        }
    }
    // Function to delete the element from the beginning
    void deleteFromBeginning() {
        if (head != nullptr) {
            Node* temp = head;
```

```
            head = head->next;
            delete temp;
        } else {
            cout << "List is empty. Cannot delete from the
beginning." << endl;
        }
    }
```

// Function to delete the element from the end position

```
    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;
        }
    }
    // Function to delete the element at any position
    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;
    }
  }
};
int main() {
  LinkedList myList;
  myList.insert(1, 10);
  myList.insert(2, 20);
  myList.insert(3, 30);
  cout << "Linked List: ";
  cout << "10 -> 20 -> 30" << endl;
  cout << "Search for 20: " << (myList.search(20) ? "Found" :
"Not found") << endl;
```

```
    myList.update(2, 25);

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

    cout << "10 -> 25 -> 30" << endl;

    myList.deleteFromBeginning();

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

    cout << "25 -> 30" << endl;

    myList.deleteFromEnd();

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

    cout << "25" << endl;

    myList.deleteAtPosition(1);

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

    cout << "(empty list)" << endl;

    return 0;

}
```



9:50

Compile Result

```
Linked List: 10 -> 20 -> 30
Search for 20: Found
Linked List after update at position 2: 10 ->
 25 -> 30
Linked List after deleting from the beginning
: 25 -> 30
Linked List after deleting from the end: 25
Linked List after deleting from position 1: (
empty list)

[Process completed - press Enter]
```