Nama : Marvel Sanjaya Setiawan

Nim : 2311104053

Laporan Asesmen Praktikum CLO 1

Program :

```cpp
//Marvel Sanjaya Setiawan
//2311104053

#include "LinkedList.h"
#include "LinkedList.cpp"

int main() {
    LinkedList list;
    list.createNewList();
    int N;
    cout << "Masukkan jumlah mahasiswa: ";
    cin >> N;

    for (int i = 0; i < N; ++i) {
        string nama, nim, kelas;
        float nilaiAsesmen, nilaiPraktikum;
        cout << "Masukkan nama: ";
        cin >> nama;
        cout << "Masukkan NIM: ";
        cin >> nim;
        cout << "Masukkan kelas: ";
        cin >> kelas;
        cout << "Masukkan nilai asesmen: ";
        cin >> nilaiAsesmen;
        cout << "Masukkan nilai praktikum: ";
        cin >> nilaiPraktikum;

        Mahasiswa* newMahasiswa = list.newElement(nama, nim, kelas, nilaiAsesmen, nilaiPraktikum);

        list.insertFirst(newMahasiswa);  // Insert menggunakan mekanisme insertFirst tanpa memeriksa NIM
    }

    cout << "\nData mahasiswa: \n";
    list.printList();

    Mahasiswa* highest = list.findHighestAsesmen();
    if (highest != nullptr) {
        cout << "\nMahasiswa dengan nilai asesmen tertinggi: \n";
        cout << "Nama: " << highest->nama << ", NIM: " << highest->nim << ", Kelas: " << highest->kelas
             << ", Nilai Asesmen: " << highest->nilaiAsesmen << ", Nilai Praktikum: " << highest-
>nilaiPraktikum << endl;
    }

    list.removeDuplicates();
    cout << "\nData mahasiswa setelah menghapus duplikat: \n";
    list.printList();

    return 0;
}
```

```cpp
#include "LinkedList.h"

void LinkedList::createNewList() {
    head = nullptr;
}

void LinkedList::insertFirst(Mahasiswa* newMahasiswa) {
    newMahasiswa->next = head;
    head = newMahasiswa;
}

Mahasiswa* LinkedList::newElement(const string& nama, const string& nim, const string& kelas, float
nilaiAsesmen, float nilaiPraktikum) {
    Mahasiswa* newMahasiswa = new Mahasiswa;
    newMahasiswa->nama = nama;
    newMahasiswa->nim = nim;
    newMahasiswa->kelas = kelas;
    newMahasiswa->nilaiAsesmen = nilaiAsesmen;
    newMahasiswa->nilaiPraktikum = nilaiPraktikum;
    newMahasiswa->next = nullptr;
    return newMahasiswa;
}

bool LinkedList::isEmpty() {
    return head == nullptr;
}

void LinkedList::deleteFirst() {
    if (isEmpty()) {
        cout << "List is empty, nothing to delete." << endl;
        return;
    }
    Mahasiswa* temp = head;
    head = head->next;
    delete temp;
}

int LinkedList::length() {
    int count = 0;
    Mahasiswa* temp = head;
    while (temp != nullptr) {
        count++;
        temp = temp->next;
    }
    return count;
}

Mahasiswa* LinkedList::findElement(const string& nim) {
    Mahasiswa* temp = head;
    while (temp != nullptr) {
        if (temp->nim == nim) {
            return temp;
        }
        temp = temp->next;
    }
    return nullptr;
}

void LinkedList::printList() {
    Mahasiswa* temp = head;
    while (temp != nullptr) {
        cout << "Nama: " << temp->nama << ", NIM: " << temp->nim << ", Kelas: " << temp->kelas
             << ", Nilai Asesmen: " << temp->nilaiAsesmen << ", Nilai Praktikum: " << temp-
>nilaiPraktikum << endl;
        temp = temp->next;
    }
}

Mahasiswa* LinkedList::findHighestAsesmen() {
    if (head == nullptr) return nullptr;
    Mahasiswa* temp = head;
    Mahasiswa* highest = head;
    while (temp != nullptr) {
        if (temp->nilaiAsesmen > highest->nilaiAsesmen) {
            highest = temp;
        }
        temp = temp->next;
    }
    return highest;
}

void LinkedList::removeDuplicates() {
    Mahasiswa* temp = head;
    while (temp != nullptr && temp->next != nullptr) {
        Mahasiswa* runner = temp;
        while (runner->next != nullptr) {
            if (temp->nim == runner->next->nim) {
                Mahasiswa* duplicate = runner->next;
                runner->next = runner->next->next;
                delete duplicate;
            } else {
                runner = runner->next;
            }
        }
        temp = temp->next;
    }
}
```

```h
#ifndef LINKEDLIST_H_INCLUDED
#define LINKEDLIST_H_INCLUDED

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

struct Mahasiswa {
    string nama;
    string nim;
    string kelas;
    float nilaiAsesmen;
    float nilaiPraktikum;
    Mahasiswa* next;
};

class LinkedList {
public:
    LinkedList();
    void createNewList();
    void insertFirst(Mahasiswa* newMahasiswa);
    Mahasiswa* newElement(const string& nama, const string& nim, const string& kelas, float nilaiAsesmen,
float nilaiPraktikum);
    bool isEmpty();
    void deleteFirst();
    int length();
    Mahasiswa* findElement(const string& nim);
    void printList();
    Mahasiswa* findHighestAsesmen();
    void removeDuplicates();
private:
    Mahasiswa* head;
};

LinkedList::LinkedList() {
    head = nullptr;
}

#endif // LINKEDLIST_H_INCLUDED
```

Output:

```
Masukkan jumlah mahasiswa: 2
Masukkan nama: msg
Masukkan NIM: 2311
Masukkan kelas: s1se0702
Masukkan nilai asesmen: 65
Masukkan nilai praktikum: 70
Masukkan nama: msg
Masukkan NIM: 2311
Masukkan kelas: s1se0702
Masukkan nilai asesmen: 80
Masukkan nilai praktikum: 55

Data mahasiswa:
Nama: msg, NIM: 2311, Kelas: s1se0702, Nilai Asesmen: 80, Nilai Praktikum: 55
Nama: msg, NIM: 2311, Kelas: s1se0702, Nilai Asesmen: 65, Nilai Praktikum: 70

Mahasiswa dengan nilai asesmen tertinggi:
Nama: msg, NIM: 2311, Kelas: s1se0702, Nilai Asesmen: 80, Nilai Praktikum: 55

Data mahasiswa setelah menghapus duplikat:
Nama: msg, NIM: 2311, Kelas: s1se0702, Nilai Asesmen: 80, Nilai Praktikum: 55

Process returned 0 (0x0)   execution time : 34.892 s
Press any key to continue.
```