Nama: Adhiansyah M. Pradana F.

Kelas: S1SE-07-02

NIM: 2211104038

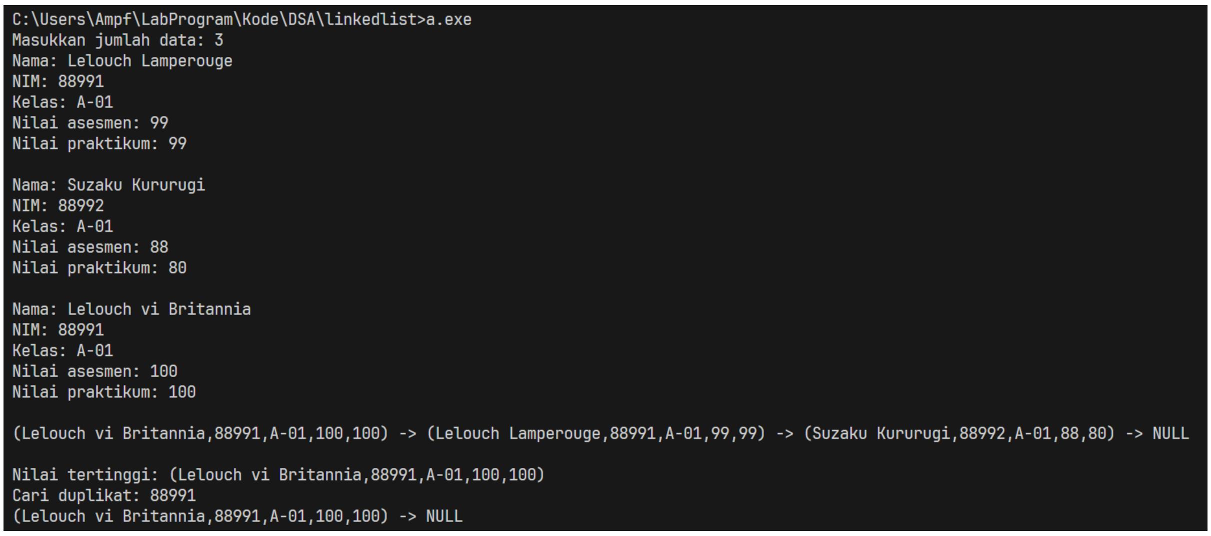
```
/**
 * Asesmen Praktikum CLO 1
 * 21 November 2024
 *
 * Daftar berantai pilihan: Daftar berantai ganda (NIM genap)
 *
 * Adhiansyah Muhammad Pradana Farawowan
 * 2211104038
 * S1SE-07-02
 *
 */

#include <iostream>
#include <string>

struct InformasiMatkul
{
    std::string nama;
    int nim;
    std::string kelas;
    int nilai_asesmen;
    int nilai_praktikum;
};

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

struct DoublyLinkedList
{
    Node *first;
    Node *last;
};

InformasiMatkul *informasiMatkul(std::string nama, int nim, std::string kelas, int
nilai_asesmen, int nilai_praktikum)
{
    struct InformasiMatkul *im = new InformasiMatkul;
    im->nama = nama;
    im->nim = nim;
    im->kelas = kelas;
    im->nilai_asesmen = nilai_asesmen;
    im->nilai_praktikum = nilai_praktikum;

    return im;
}

/* function newElement(data : infotype ) → address */
Node *newElement(InformasiMatkul *data)
{
    struct Node *n = new Node;
    n->data = data;
    n->next = nullptr;
    n->prev = nullptr;

    return n;
}

/* function createNewList() →List */
DoublyLinkedList *createNewList()
{
    struct DoublyLinkedList *dll = new DoublyLinkedList;
    dll->first = nullptr;
    dll->last = nullptr;
```

```cpp
    return dll;
}

/* function isEmpty(a: List) → boolean */
bool isEmpty(DoublyLinkedList *a)
{
    if (a->first != nullptr)
    {
        return false;
    }

    return true;
}

/* procedure insertFirst(in/out a:List, in p:address) */
void insertFirst(DoublyLinkedList *a, Node *p)
{
    if (a->first == nullptr)
    {
        a->first = p;
        a->last = p;

        return;
    }

    p->next = a->first;
    a->first = p;

    return;
}

/* procedure insertAfter(in/out a:List , in x:infotype, in p:address) */
void insertAfter(DoublyLinkedList *a, InformasiMatkul *x, Node *p)
{
    if (a->first == nullptr)
    {
        std::cout << "MAIN.CPP: Linked list is empty, abort." << '\n';
        return;
    }

    Node *current = a->first;
    while (current != nullptr)
    {
        if (current->data == x)
        {
            Node *after_current = current->next;
            p->next = after_current;
            p->prev = current;

            current->next = p;
            after_current->prev = p;

            return;
        }

        current = current->next;
    }

    std::cout << "MAIN.CPP: Related data is not found, abort." << '\n';
    return;
}

/* procedure insertLast(in/out a:List, in p:address) */
void insertLast(DoublyLinkedList *a, Node *p)
{
    if (a->first == nullptr)
    {
        a->first = p;
        a->last = p;
        return;
    }

    a->last->next = p;
    p->prev = a->last;
    a->last = a->last->next;
    return;
```

```
}

/* procedure deleteFirst(in/out a:List, p:address) */
void deleteFirst(DoublyLinkedList *a)
{
    if (a->first == nullptr)
    {
        std::cout << "MAIN.CPP: Linked list is empty, abort." << '\n';
        return;
    }

    if (a->first == a->last)
    {
        Node *p = a->first;
        a->first = nullptr;
        a->last = nullptr;

        delete p;

        return;
    }

    Node *p = a->first;
    a->first = a->first->next;
    a->first->prev = nullptr;

    delete p;

    return;
}

/* procedure deleteLast(in/out a:List, p:address) */
void deleteLast(DoublyLinkedList *a)
{
    if (a->last == nullptr)
    {
        std::cout << "MAIN.CPP: Linked list is empty, abort." << '\n';
        return;
    }

    if (a->first == a->last)
    {
        Node *p = a->first;
        a->first = nullptr;
        a->last = nullptr;

        delete p;

        return;
    }

    Node *p = a->last;
    a->last = a->last->prev;
    a->last->next = nullptr;

    delete p;

    return;
}

/* function length(a: List) → integer */
int length(DoublyLinkedList *a)
{
    int counts = 0;
    Node *current = a->first;
    while (current != nullptr)
    {
        counts = counts + 1;
        current = current->next;
    }

    return counts;
}

/* function findElement(a: List, x: infotype) → address */
Node *findElement(DoublyLinkedList *a, InformasiMatkul *x)
```

```
{
    Node *found = a->first;
    while (found != nullptr)
    {
        if (found->data == x)
        {
            return found;
        }

        found = found->next;
    }

    return nullptr;
}

/* procedure printList(a: List) */
void printList(DoublyLinkedList *a)
{
    if (a->first == nullptr)
    {
        std::cout << "MAIN.CPP: Linked list is empty, abort." << '\n';
        return;
    }

    Node *current = a->first;
    while (current != nullptr)
    {
        std::cout << '('
                  << current->data->nama << ','
                  << current->data->nim << ','
                  << current->data->kelas << ','
                  << current->data->nilai_asesmen << ','
                  << current->data->nilai_praktikum << ')'
                  << " -> ";

        current = current->next;
    }

    std::cout << "NULL" << '\n';
    return;
}

void soal_pertama(DoublyLinkedList *data_std)
{
    std::string nama;
    int nim;
    std::string kelas;
    int nilai_asesmen;
    int nilai_praktikum;

    int jumlah_data;
    std::cout << "Masukkan jumlah data: ";
    std::cin >> jumlah_data;
    std::cin.ignore();

    int pos = 0;
    while (pos != jumlah_data)
    {
        std::cout << "Nama: ";
        std::getline(std::cin, nama);

        std::cout << "NIM: ";
        std::cin >> nim;

        std::cout << "Kelas: ";
        std::cin >> kelas;

        std::cout << "Nilai asesmen: ";
        std::cin >> nilai_asesmen;

        std::cout << "Nilai praktikum: ";
        std::cin >> nilai_praktikum;
        std::cin.ignore();

        InformasiMatkul *info = informasiMatkul(nama, nim, kelas, nilai_asesmen,
nilai_praktikum);
```

```cpp
        Node *node_info = newElement(info);

        if (nim % 2 == 0)
        {
            insertLast(data_std, node_info);
        }
        else
        {
            insertFirst(data_std, node_info);
        }

        pos = pos + 1;
        std::cout << '\n';
    }

    printList(data_std);
    std::cout << '\n';
}

void soal_kedua(DoublyLinkedList *data_std) {
    Node *current = data_std->first;
    Node *asesmen_tertinggi = data_std->first;
    while (current != nullptr)
    {
        if (current->data->nilai_asesmen > asesmen_tertinggi->data->nilai_asesmen) {
            asesmen_tertinggi = current;
        }

        current = current->next;
    }
    std::cout << "Nilai tertinggi: ";
    std::cout << '('
              << asesmen_tertinggi->data->nama << ','
              << asesmen_tertinggi->data->nim << ','
              << asesmen_tertinggi->data->kelas << ','
              << asesmen_tertinggi->data->nilai_asesmen << ','
              << asesmen_tertinggi->data->nilai_praktikum << ')';

    std::cout << '\n';
}

void soal_ketiga(DoublyLinkedList *data_std) {
    int find_duplicate;
    std::cout << "Cari duplikat: ";
    std::cin >> find_duplicate;
    std::cin.ignore();

    Node *current = data_std->first;
    while (current != nullptr)
    {
        if (current->data->nim == find_duplicate) {
            current->next = nullptr;
            current->prev = nullptr;
        }

        current = current->next;
    }

    printList(data_std);
}

int main()
{
    DoublyLinkedList *data_std;
    data_std = createNewList();

    soal_pertama(data_std);
    soal_kedua(data_std);
    soal_ketiga(data_std);

    return 0;
}
```


```
C:\Users\Ampf\LabProgram\Kode\DSA\linkedlist>a.exe
Masukkan jumlah data: 3
Nama: Lelouch Lamperouge
NIM: 88991
Kelas: A-01
Nilai asesmen: 99
Nilai praktikum: 99

Nama: Suzaku Kururugi
NIM: 88992
Kelas: A-01
Nilai asesmen: 88
Nilai praktikum: 80

Nama: Lelouch vi Britannia
NIM: 88991
Kelas: A-01
Nilai asesmen: 100
Nilai praktikum: 100

(Lelouch vi Britannia,88991,A-01,100,100) -> (Lelouch Lamperouge,88991,A-01,99,99) -> (Suzaku Kururugi,88992,A-01,88,80) -> NULL

Nilai tertinggi: (Lelouch vi Britannia,88991,A-01,100,100)
Cari duplikat: 88991
(Lelouch vi Britannia,88991,A-01,100,100) -> NULL
```