LAPORAN PRAKTIKUM

PERTEMUAN 5

SINGLE LINKED LIST (BAGIAN KEDUA)

Nama :

Resita Istania Purwanto (2311104037)

Dosen :

Yudha Islami Sulistya

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

I. TP

SOAL 1

Kode:

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

Codeium: Refactor | Explain
struct Node {
    int data;
    Node* next;
};

Codeium: Refactor | Explain | Generate Function Comment | ×
void insert(Node*& head, int value) {
    Node* newNode = new Node();
    newNode->data = value;
    newNode->next = head;
    head = newNode;
}

Codeium: Refactor | Explain | Generate Function Comment | ×
void searchElement(Node* head, int value) {
    Node* current = head;
    int position = 1;

    while (current != nullptr) {
        if (current->data == value) {
            cout << "Elemen ditemukan pada alamat: " << current << " di urutan ke-" << position << endl;
            return;
        }
        current = current->next;
        position++;
    }

    cout << "Elemen tidak ditemukan dalam list." << endl;
}
```

```cpp
Codeium: Refactor | Explain | Generate Function Comment | ×
int main() {
    Node* head = nullptr;
    int values[6];

    // Memasukkan elemen ke dalam list
    cout << "Masukkan 6 elemen integer: ";
    for (int i = 0; i < 6; i++) {
        cin >> values[i];
        insert(head, values[i]);
    }

    // Memasukkan elemen yang ingin dicari
    int searchValue;
    cout << "Masukkan elemen yang ingin dicari: ";
    cin >> searchValue;

    // Mencari elemen
    searchElement(head, searchValue);

    return 0;
}
```

Penjelasan:

- Struct Node: Mewakili simpul dalam Linked List, dengan anggota data untuk menyimpan nilai dan next yang menunjuk ke node berikutnya.
- insert: Menambahkan node baru di awal Linked List. Node baru berisi nilai yang dimasukkan, dan menunjuk ke node sebelumnya.
- searchElement: Mencari nilai tertentu dalam Linked List. Jika ditemukan, akan mencetak alamat dan posisinya; jika tidak, akan mencetak bahwa elemen tidak ditemukan.
- main:Meminta input 6 integer dari user untuk dimasukkan ke Linked List dan Meminta input nilai yang ingin dicari, lalu mencarinya di dalam list menggunakan searchElement.

Output:

```
Masukkan 6 elemen integer: 10 20 30 40 50 60
Masukkan elemen yang ingin dicari: 30
Elemen ditemukan pada alamat: 0x1732a542e60 di urutan ke-4
PS C:\05_Single_Linked_List_Bagian_2\TP>
```

Penjelasan: Urutan posisi 30 muncul sebagai urutan ke-4 bukan ke-3 karena elemen-elemen dimasukkan ke dalam Linked List dari depan. Setiap kali elemen dimasukkan, elemen baru ditempatkan di awal list. Ini menyebabkan urutan elemen menjadi terbalik.

SOAL 2

Kode:

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

Codeium: Refactor | Explain
struct Node {
    int data;
    Node* next;
};

// Fungsi untuk mencetak Linked List
Codeium: Refactor | Explain | ×
void printList(Node* head) {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}

// Fungsi Bubble Sort untuk mengurutkan Linked List
Codeium: Refactor | Explain | ×
void bubbleSort(Node* head) {
    if (head == nullptr) return;  // Jika list kosong, langsung kembali

    bool swapped;
    Node* current;
    Node* last = nullptr;  // Menandai bagian akhir dari list yang sudah terurut

    int iterasi = 1;  // Counter untuk melihat jumlah iterasi

    do {
        swapped = false;
        current = head;

        cout << "Iterasi " << iterasi << ": ";
        // Iterasi melalui Linked List hingga ke node yang belum terurut
        while (current->next != last) {
            if (current->data > current->next->data) {
                // Menukar data antara node saat ini dan node berikutnya
                int temp = current->data;
                current->data = current->next->data;
                current->next->data = temp;
                swapped = true;  // Menandai bahwa pertukaran terjadi
            }
            current = current->next;  // Lanjut ke node berikutnya
        }

        // Cetak list setelah setiap iterasi
        printList(head);

        last = current;  // Update bagian akhir dari list yang sudah terurut
        iterasi++;
    } while (swapped);  // Ulangi sampai tidak ada pertukaran lagi
}

Codeium: Refactor | Explain | Generate Function Comment | ×
int main() {
    // Membuat Linked List tanpa elemen 5: 4 -> 2 -> 3 -> 1
    Node* head = new Node{4, nullptr};
    head->next = new Node{2, nullptr};
    head->next->next = new Node{3, nullptr};
    head->next->next->next = new Node{1, nullptr};

    // Cetak Linked List sebelum diurutkan
```

```
    cout << "List sebelum diurutkan: ";
    printList(head);

    // Mengurutkan List menggunakan Bubble Sort
    bubbleSort(head);

    // Cetak Linked List setelah diurutkan
    cout << "List setelah diurutkan: ";
    printList(head);

    return 0;
}
```

Penjelasan:

- Struct Node menyimpan data dan pointer ke node berikutnya.
- Fungsi printList mencetak elemen-elemen Linked List.
- Fungsi bubbleSort mengurutkan Linked List menggunakan algoritma Bubble Sort
- Di dalam main(), Linked List berisi elemen 4 -> 2 -> 3 -> 1 dibuat, kemudian diurutkan dan dicetak sebelum dan sesudah pengurutan.

Output:

```
List sebelum diurutkan: 4 2 3 1
Iterasi 1: 2 3 1 4
Iterasi 2: 2 1 3 4
Iterasi 3: 1 2 3 4
Iterasi 4: 1 2 3 4
List setelah diurutkan: 1 2 3 4
PS C:\05_Single_Linked_List_Bagian_2\TP>
```

Penjelasan: List berubah dari tidak terurut menjadi terurut dalam tiga iterasi.

SOAL 3

Kode:

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

Codeium: Refactor | Explain
struct Node {
    int data;
    Node* next;
};

Codeium: Refactor | Explain | Generate Function Comment | ×
void insert(Node*& head, int value) {
    Node* newNode = new Node();
    newNode->data = value;
    newNode->next = head;
    head = newNode;
}

Codeium: Refactor | Explain | Generate Function Comment | ×
void insertSorted(Node*& head, int value) {
    Node* newNode = new Node();
    newNode->data = value;

    if (head == nullptr || head->data >= newNode->data) {
        newNode->next = head;
        head = newNode;
    } else {
        Node* current = head;
        while (current->next != nullptr && current->next->data < newNode->data) {
            current = current->next;
        }
        newNode->next = current->next;
        current->next = newNode;
    }
}
```

```
Codeium: Refactor | Explain | Generate Function Comment | ×
void printList(Node* head) {
    while (head != nullptr) {
        cout << head->data << " ";
        head = head->next;
    }
    cout << endl;
}

Codeium: Refactor | Explain | Generate Function Comment | ×
int main() {
    Node* head = nullptr;
    int values[4];

    // Memasukkan 4 elemen ke dalam list secara terurut2
    cout << "Masukkan 4 elemen integer secara terurut: ";
    for (int i = 0; i < 4; i++) {
        cin >> values[i];
        insertSorted(head, values[i]);
    }

    // Menampilkan list sebelum penambahan elemen baru
    cout << "List sebelum penambahan: ";
    printList(head);

    // Memasukkan elemen baru yang harus tetap terurut
    int newValue;
    cout << "Masukkan elemen baru yang ingin ditambahkan: ";
    cin >> newValue;
    insertSorted(head, newValue);

    // Menampilkan list setelah penambahan
    cout << "List setelah penambahan: ";
    printList(head);

    return 0;
}
```

Penjelasan:

- Struct Node: Menyimpan data integer dan pointer ke node berikutnya.
- Fungsi insert: Menambahkan elemen di awal Linked List (tidak digunakan).
- Fungsi insertSorted: Menyisipkan elemen baru secara terurut ke dalam Linked List.
- Fungsi printList: Mencetak elemen-elemen Linked List.
- Fungsi main: Mengambil 4 elemen terurut dari pengguna dan menampilkan Linked List sebelum dan sesudah penambahan elemen baru.

Output:

```
Masukkan 4 elemen integer secara terurut: 2 5 7 8
List sebelum penambahan: 2 5 7 8
Masukkan elemen baru yang ingin ditambahkan: 4
List setelah penambahan: 2 4 5 7 8
```

Penjelasan:

- List sebelum penambahan: Menampilkan 4 elemen yang dimasukkan oleh pengguna secara terurut, misalnya 2 5 7 8.
- List setelah penambahan: Menampilkan Linked List setelah menambahkan elemen baru (misalnya 4) di posisi yang tepat, menjadi 2 4 5 7 8.
- Output menunjukkan bahwa elemen tetap terurut setelah penambahan.

II. UNGUIDED

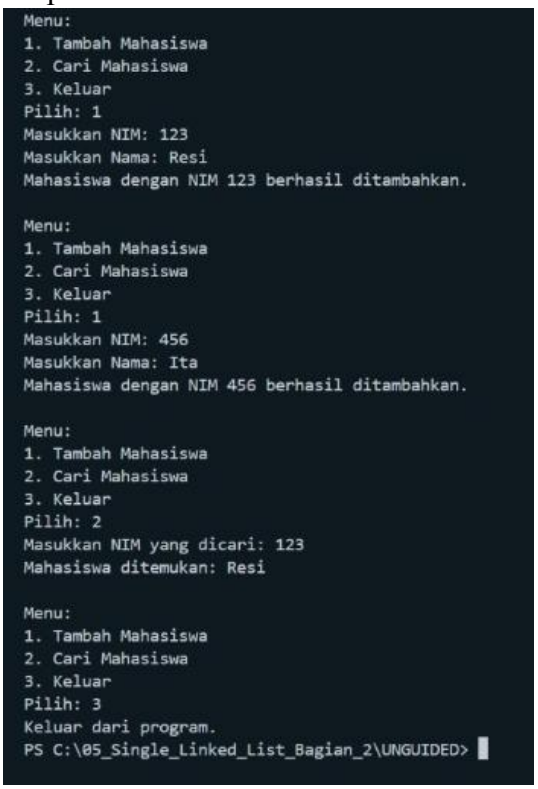
Kode:

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

Codeium: Refactor | Explain
struct Mahasiswa {
    int NIM;
    string nama;
    Mahasiswa* next;
};

Mahasiswa* head = nullptr;

// Fungsi untuk menambahkan mahasiswa ke dalam list
Codeium: Refactor | Explain | ×
void tambahMahasiswa(int nim, string nama) {
    Mahasiswa* baru = new Mahasiswa();
    baru->NIM = nim;
    baru->nama = nama;
    baru->next = head;
    head = baru;
    cout << "Mahasiswa dengan NIM " << nim << " berhasil ditambahkan." << endl;
}

// Fungsi untuk mencari mahasiswa berdasarkan NIM
Codeium: Refactor | Explain | ×
void cariMahasiswa(int nim) {
    Mahasiswa* current = head;
    while (current != nullptr) {
        if (current->NIM == nim) {
            cout << "Mahasiswa ditemukan: " << current->nama << endl;
            return;
        }
        current = current->next;
    }
    cout << "Mahasiswa dengan NIM " << nim << " tidak ditemukan." << endl;
}

Codeium: Refactor | Explain | Generate Function Comment | ×
int main() {
    int pilihan, nim;
    string nama;

    do {
        cout << "\nMenu:\n";
        cout << "1. Tambah Mahasiswa\n";
        cout << "2. Cari Mahasiswa\n";
        cout << "3. Keluar\n";
        cout << "Pilih: ";
        cin >> pilihan;

        switch (pilihan) {
            case 1:
                cout << "Masukkan NIM: ";
                cin >> nim;
                cin.ignore();  // Membersihkan buffer
                cout << "Masukkan Nama: ";
                getline(cin, nama);
                tambahMahasiswa(nim, nama);
                break;
            case 2:
                cout << "Masukkan NIM yang dicari: ";
                cin >> nim;
                cariMahasiswa(nim);
                break;
            case 3:
                cout << "Keluar dari program.\n";
                break;
            default:
                cout << "Pilihan tidak valid.\n";
        }
    } while (pilihan != 3);

    return 0;
}
```

Penjelasan:

- Struct Mahasiswa: Menyimpan data mahasiswa (NIM, nama) dan pointer ke node berikutnya.
- Fungsi tambahMahasiswa: Menambahkan mahasiswa ke Linked List dan menampilkan konfirmasi.
- Fungsi cariMahasiswa: Mencari mahasiswa berdasarkan NIM dan menampilkan nama jika ditemukan.
- Fungsi main: Menampilkan menu untuk menambah atau mencari mahasiswa hingga pengguna memilih keluar.

Output:



Penjelasan:

- User menambah mahasiswa dan mencari berdasarkan NIM.
- Program berlanjut hingga pengguna memilih untuk keluar.