Ketentuan Tugas Pendahuluan

- Untuk soal teori **JAWABAN DIKETIK DENGAN RAPI** dan untuk soal algoritma **SERTAKAN SCREENSHOOT CODINGAN DAN HASIL OUTPUT.**
- Deadline pengumpulan TP Modul 5 adalah **Senin, 14 Oktober 2024 pukul 06.00 WIB.**
- **TIDAK ADA TOLERANSI KETERLAMBATAN, TERLAMBAT ATAU TIDAK MENGUMPULKAN TP ONLINE MAKA DIANGGAP TIDAK MENGERJAKAN.**
- **DILARANG PLAGIAT (PLAGIAT = E).**
- Kerjakan TP dengan jelas agar dapat dimengerti.
- Untuk setiap soal nama fungsi atau prosedur **WAJIB** menyertakan **NIM**, contoh: **insertFirst_130122xxxx**.
- File diupload di LMS menggunakan format **PDF** dengan ketentuan **:** **TP_MODX_NIM_KELAS.pdf**

```
Contoh:
int searchNode_130122xxxx (List L, int X);
```

CP:

- Raihan (089638482851)
- Kayyisa (085105303555)
- Abiya (082127180662)
- Rio (081210978384)

SELAMAT MENGERJAKAN^^

LAPORAN PRAKTIKUM

PERTEMUAN 5

SINGLE LINKED LIST BAGIAN BAGIAN KEDUA

Nama :

Tsaqif Hisyam Saputra (2311104024)

Dosen :

Yudha Islami Sulistya

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

A. Tujuan

- Untuk memahami penggunan linked list dengan pointer operator-operator dalam program.
- Memahami operasi-operasi dasar dalam linked list.
- Membuat program dengan menggunakan linked list dengan prototype yang ada.

B. Tools

Visual Studio Code dengan C++ Extension Pack.

TUGAS PENDAHULUAN

Soal 1

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

// Node structure for the linked list
struct Node {
    int data;
    Node* next;
};

// Function to insert a new node at the end of the list
void insert(Node*& head, int value) {
    Node* newNode = new Node();
    newNode->data = value;
    newNode->next = nullptr;

    if (head == nullptr) {
        head = newNode;
    } else {
        Node* temp = head;
        while (temp->next != nullptr) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}

// Function to search for an element in the list
void searchElement(Node* head, int key) {
    Node* current = head;
    int position = 1;
    bool found = false;

    while (current != nullptr) {
        if (current->data == key) {
            found = true;
            cout << "Elemen ditemukan pada alamat: " << current << " di posisi ke-" << position << endl;
            break;
        }
        current = current->next;
        position++;
    }

    if (!found) {
        cout << "Elemen tidak ditemukan dalam list." << endl;
    }
}

int main() {
    Node* head = nullptr;
    int elements[6];

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

    int searchValue;
    cout << "Masukkan nilai yang ingin dicari: ";
    cin >> searchValue;

    // Search for the element
    searchElement(head, searchValue);

    return 0;
}
```

Dalam kodingan tersebut terdapat

Struktur Node: program yang mendefinisikan struktur **node** yang menyimpan data integer dan pointer ke node berikutnya.

Fungsi insert: Untuk menambahkan elemen ke dalam linked list.

Fungsi searchElement: Menggunakan pendekatan linear search untuk mencari elemen dalam linked list. Jika elemen ditemukan, ia akan menampilkan alamat dan posisi elemen tersebut. Jika tidak, akan menampilkan pesan bahwa elemen tidak ditemukan. Main Function: Meminta pengguna untuk memasukkan 6 elemen integer dan kemudian mencari elemen yang diinput pengguna.

Output:

```
* Executing task: C:/Windows/System32/cmd.exe /d /c .\build\Debug\outDebug.exe

Masukkan 6 elemen integer ke dalam list:
1 2 3 4 5 6
Masukkan nilai yang ingin dicari: 3
Elemen ditemukan pada alamat: 0x1ecdf6d1200 di posisi ke-3
* Terminal will be reused by tasks, press any key to close it.
```

Soal 2

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

// Struktur Node untuk Linked List
struct Node {
    int data;
    Node* next;
};

// Fungsi untuk menambahkan elemen ke akhir list
void insert(Node*& head, int value) {
    Node* newNode = new Node();
    newNode->data = value;
    newNode->next = nullptr;

    if (head == nullptr) {
        head = newNode;
    } else {
        Node* temp = head;
        while (temp->next != nullptr) {
            temp = temp->next;
        }
        temp->next = newNode;
    }
}

// Fungsi untuk menampilkan elemen-elemen list
void displayList(Node* head) {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}

// Prosedur Bubble Sort untuk mengurutkan elemen dalam linked list
void bubbleSort(Node* head) {
    if (head == nullptr) return; // Jika list kosong, tidak ada yang diurutkan

    bool swapped;
    Node* current;
    Node* lastSorted = nullptr;  // Untuk menandai batas bagian yang sudah terurut

    do {
        swapped = false;
        current = head;

        while (current->next != lastSorted) {
            if (current->data > current->next->data) {
                // Tukar data
                swap(current->data, current->next->data);
                swapped = true;
            }
            current = current->next;
        }
        lastSorted = current;  // Update batas bagian yang sudah terurut
    } while (swapped);
}

int main() {
    Node* head = nullptr;
    int elements[5];

    // Meminta input 5 elemen dari pengguna
    cout << "Masukkan 5 elemen integer ke dalam list:" << endl;
    for (int i = 0; i < 5; i++) {
        cin >> elements[i];
        insert(head, elements[i]);
    }

    // Menampilkan list sebelum sorting
    cout << "List sebelum diurutkan: ";
    displayList(head);

    // Mengurutkan list menggunakan bubble sort
    bubbleSort(head);

    // Menampilkan list setelah sorting
    cout << "List setelah diurutkan: ";
    displayList(head);

    return 0;
}
```

Sama seperti soal sebelumnya, Node menyimpan data integer dan pointer ke node berikutnya. Fungsi insert digunakan untuk menambahkan elemen ke dalam linked list. Fungsi diplayList menampilkan elemen-elemen dalam linked list. Lalu prosedur

bubbleSort Mengimplementasikan algoritma bubble sort dengan pointer current yang menulusuri list dan melakukan pertukaran data jika elemen sekarang lebih besar dari elemen berikutnya. Pertukaran hanya dilakukan pada data, bukan pada pointer. Menggunakan variabel swapped untuk memeriksa apakah ada pertukaran pada iterasi tertentu. Proses diulang sampai tidak ada pertukaran yang dilakukan, yang berarti list sudah terurut. Pada main function akan meminta pengguna memasukkan 5 elemen integer ke dalam list, menampilkan list sebelum dan sesudah diurutkan.

Output:

```
Executing task: C:/Windows/System32/cmd.exe /d /c .\build\Debug\outDebug.exe

Masukkan 5 elemen integer ke dalam list:
3 5 1 2 4
List sebelum diurutkan: 3 5 1 2 4
List setelah diurutkan: 1 2 3 4 5
Terminal will be reused by tasks, press any key to close it.
```

Soal 3

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

// Struktur Node untuk Linked List
struct Node {
    int data;
    Node* next;
};

// Fungsi untuk menambahkan elemen ke dalam list secara urut
void insertSorted(Node*& head, int value) {
    Node* newNode = new Node();
    newNode->data = value;
    newNode->next = nullptr;

    // Jika list kosong atau nilai elemen yang baru lebih kecil dari head, masukkan di depan
    if (head == nullptr || head->data >= value) {
        newNode->next = head;
        head = newNode;
    } else {
        // Cari posisi yang sesuai untuk memasukkan elemen baru
        Node* current = head;
        while (current->next != nullptr && current->next->data < value) {
            current = current->next;
        }
        newNode->next = current->next;
        current->next = newNode;
    }
}

// Fungsi untuk menampilkan elemen-elemen list
void displayList(Node* head) {
    Node* temp = head;
    while (temp != nullptr) {
        cout << temp->data << " ";
        temp = temp->next;
    }
    cout << endl;
}

int main() {
    Node* head = nullptr;
    int elements[4];

    // Meminta input 4 elemen dari pengguna untuk membuat list terurut
    cout << "Masukkan 4 elemen integer yang sudah terurut ke dalam list:" << endl;
    for (int i = 0; i < 4; i++) {
        cin >> elements[i];
        insertSorted(head, elements[i]);
    }

    // Menampilkan list sebelum menambahkan elemen baru
    cout << "List sebelum elemen baru dimasukkan: ";
    displayList(head);

    // Meminta elemen baru yang akan dimasukkan secara terurut
    int newValue;
    cout << "Masukkan elemen baru yang ingin dimasukkan: ";
    cin >> newValue;

    // Menambahkan elemen baru ke dalam list yang sudah terurut
    insertSorted(head, newValue);

    // Menampilkan list setelah elemen baru dimasukkan
    cout << "List setelah elemen baru dimasukkan: ";
    displayList(head);

    return 0;
}
```

Struktur node sama seperti pada program-program sebelumnya, setiap node menyimpan data dan pointer ke node berikutnya. Fungsi insertSorted menambahkan elemen baru ke linked list yang sudah terurut. Algoritma bekerja dengan mencari posisi yang sesuai di mana elemen baru lebih kecil dari elemen berikutnya, dan melakukan penyisispan. Jika linked list kosong atau elemen baru lebih kecil atau sama dengan elemen pertama, elemen baru disisipkan di depan. Jika elemen baru lebih besar, algoritma menelusuri list dan menemukan posisi yang sesuai untuk penyisipan. Fungsi displayList digunakan untuk menampilkan semua elemen di dalam linked list. Main Function yaitu program untuk meminta pengguna untuk memasukkan 4 elemen yang sudah terurut, kemudian pengguna diminta untuk memasukkan elemen baru, dan program menampilkan list sebelum dan sesudah penambahan elemen baru.

UNGUIDED

```cpp
#include <iostream>
#include <string>

using namespace std;

// Struktur node untuk mahasiswa
struct Node {
    int NIM;          // Menyimpan NIM mahasiswa
    string nama;      // Menyimpan nama mahasiswa
    Node* next;       // Pointer ke node berikutnya
};

// Struktur linked list
struct LinkedList {
    Node* head;       // Pointer ke node pertama (head)

    LinkedList() {
        head = nullptr;  // Inisialisasi head dengan null
    }

    // Fungsi untuk membuat node baru dengan data NIM dan Nama
    Node* createNode(int nim, string nama) {
        Node* newNode = new Node();
        newNode->NIM = nim;
        newNode->nama = nama;
        newNode->next = nullptr;
        return newNode;
    }

    // Fungsi untuk menambahkan node di awal list
    void insertFirst(int nim, string nama) {
        Node* newNode = createNode(nim, nama);
        newNode->next = head;
        head = newNode;
    }

    // Fungsi untuk mencari mahasiswa berdasarkan NIM
    Node* searchByNIM(int nim) {
        Node* temp = head;
        while (temp != nullptr) {
            if (temp->NIM == nim) {
                return temp;
            }
            temp = temp->next;
        }
        return nullptr;
    }

    // Fungsi untuk menampilkan semua data mahasiswa
    void displayList() {
        Node* temp = head;
        while (temp != nullptr) {
            cout << "NIM: " << temp->NIM << ", Nama: " << temp->nama << endl;
            temp = temp->next;
        }
    }
};

int main() {
    LinkedList list;

    // Menambahkan data mahasiswa ke dalam linked list
    list.insertFirst(12345, "Alexander Morrison");
    list.insertFirst(67890, "Olivia Hernandez");
    list.insertFirst(11121, "Daniel Fitzgerald");
    list.insertFirst(54321, "Sophia Whitmore");

    // Menampilkan data mahasiswa yang ada di linked list
    cout << "Daftar Mahasiswa:" << endl;
    list.displayList();

    // Mencari mahasiswa berdasarkan NIM
    int nim;
    cout << "\nMasukkan NIM yang ingin dicari: ";
    cin >> nim;

    Node* result = list.searchByNIM(nim);
    if (result != nullptr) {
        cout << "Mahasiswa ditemukan! Nama: " << result->nama << endl;
    } else {
        cout << "Mahasiswa dengan NIM " << nim << " tidak ditemukan." << endl;
    }

    return 0;
}
```

Dari kodingan di atas dapat dilihat mengimplementasi Linked List yang menyimpan data mahasiswa, yang terdiri dari NIM (Nomor Induk Mahasiwa) dan nama. Struktur **Node** merepresentasikan setiap node dalam linked list dengan atribut **NIM**, **nama**, dan pointer **next** untuk menunjuk ke node berikutnya. Struktur **LinkedList** memiliki pointer **head** yang menunjuk ke node pertama dalam list. Kode ini memiliki fungsi **createNode()** untuk membuat node baru, **insertFirst()** untuk menambahkan node di awal list, **searchByNIM()** untuk mencari node di berdasarkan NIM, dan **displayList()** untuk menampilkan semua data mahasisw dalam linked list. Di dalam fungsi **main()**, beberapa data mahasiswa ditambahkan ke dalam list dengan **insertFirst()**, kemudian data ditampilkan dengan **displayList()**. Setelah itu, program meminta input NIM dari pengguna untuk mencari data mahasiswa terkait, dan menampilkan hasilnya, apakah ditemukan atau tidak.

OUTPUT:

Jika orang ditemukan

```
 *  Executing task: C:/Windows/System32/cmd.exe /d /c .\build\Debug\outDebug.exe

Daftar Mahasiswa:
NIM: 54321, Nama: Sophia Whitmore
NIM: 11121, Nama: Daniel Fitzgerald
NIM: 67890, Nama: Olivia Hernandez
NIM: 12345, Nama: Alexander Morrison

Masukkan NIM yang ingin dicari: 12345
Mahasiswa ditemukan! Nama: Alexander Morrison
 *  Terminal will be reused by tasks, press any key to close it.
```

Jika orang tidak ditemukan

```
 *  Executing task: C:/Windows/System32/cmd.exe /d /c .\build\Debug\outDebug.exe

Daftar Mahasiswa:
NIM: 54321, Nama: Sophia Whitmore
NIM: 11121, Nama: Daniel Fitzgerald
NIM: 67890, Nama: Olivia Hernandez
NIM: 12345, Nama: Alexander Morrison

Masukkan NIM yang ingin dicari: 666
Mahasiswa dengan NIM 666 tidak ditemukan.
 *  Terminal will be reused by tasks, press any key to close it.
```

Guided

```cpp
#include "singlelist.cpp"

int main(){
    List L;
    Elemen* P1, * P2, * P3, * P4, * P5 = NULL;
    createList(L);

    P1 = alokasi(2);
    insertFirst(L, P1);

    P2 = alokasi (0);
    insertFirst(L, P2);

    P3 = alokasi (8);
    insertFirst(L, P3);

    P4 = alokasi(12);
    insertFirst(L, P4);

    P5 = alokasi(9);
    insertFirst(L, P5);

    printInfo(L);

    Elemen* found = findElm(L, 8);
    if (found != NULL){
        cout << found->data << " ditemukan dalam list" << endl;
    } else {
        cout << "Elemen tidak ditemukan" << endl;
    }

    int total = sumAllElements(L);
    cout << "Total info dari kelima elemen adalah " << total << endl;

    return 0;
}
```

```cpp
#include <iostream>

using namespace std;

struct Elemen{
    int data;
    Elemen* next;
};

struct List{
    Elemen* first;
};

void createList(List& L){
    L.first = NULL;
}

Elemen* alokasi(int x){
    Elemen* P = new Elemen;
    if (P != NULL){
        P->data = x;
        P->next = NULL;
    }
    return P;
}

void insertFirst(List& L, Elemen* P){
    P->next = L.first;
    L.first = P;
}

void printInfo(List L){
    Elemen* P = L.first;
    while (P != NULL) {
        cout << P->data << " ";
        P = P->next;
    }
    cout << endl;
}

Elemen* findElm(List L, int x){
    Elemen* P = L.first;
    while (P != NULL) {
        if (P->data == x){
            return P;
        }
        P = P->next;
    }
    return NULL;
}

int sumAllElements(List L){
    int total = 0;
    Elemen* P = L.first;
    while (P != NULL){
        total += P->data;
        P = P->next;
    }
    return total;
}
```