LAPORAN PRAKTIKUM
PERTEMUAN 4
MODUL 4 STD

Nama :

Farrel I’zaz Yuwono (2311104014)

Dosen :

YUDHA ISLAMI SULISTYA

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

TUGAS PENDAHULUAN

1. list.cpp

```
#include "list.h"
void createList(List &L) {
    first(L) = nil;
}
address allocate(infotype x) {
    address P = new elmtList;
    info(P) = x;
    next(P) = nil;
    return P;
}
void insertFirst(List &L, address P) {
    next(P) = first(L);
    first(L) = P;
}
void printInfo(List L) {
    address P = first(L);
    while (P != nil) {
        cout << info(P) << " ";
        P = next(P);
    }
    cout << endl;
}
void insertLast(List &L, address P) {
    if (first(L) == nil) {
        first(L) = P;
    }
```

```
else {
        address last = first(L);
        while (next(last) != nil) {
            last = next(last);
        }
        next(last) = P;
    }
}
```

Main.cpp

```
#include "list.h"
int main() {
    List L;
    createList(L);
    // Isi 3 digit NIM terakhir
    address P1 = allocate(0);  // contoh: NIM terakhir
    address P2 = allocate(1);
    address P3 = allocate(4);
    insertFirst(L, P1);
    insertFirst(L, P2);
    insertFirst(L, P3);
    printInfo(L);
    return 0;
}
```

```
PS D:\kuliah\strukturdat\tugas kuliah\04_Single_Linked_List\TP> & .\'TP.exe'
4 1 0
PS D:\kuliah\strukturdat\tugas kuliah\04_Single_Linked_List\TP>
```

UNGUIDED

1. Program nomor 1 dan 2 saya gabung, karena add dan hapus list ada di program yang sama.

```
#include <iostream>

using namespace std;

struct Node

{

   int data;

   Node *next;

};

Node *head = NULL;


void insertDepan(int nilai)

{

   Node *newNode = new Node();

   newNode->data = nilai;

   newNode->next = head;

   head = newNode;

}


void insertBelakang(int nilai)

{

   Node *newNode = new Node();

   newNode->data = nilai;

   newNode->next = NULL;

   if (head == NULL)

   {

      head = newNode;

   }
```

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


void cetakLinkedList()
{
   Node *temp = head;
   while (temp != NULL)
   {
      cout << temp->data << " -> ";
      temp = temp->next;
   }
   cout << "NULL" << endl;
}


void hapusNode(int nilai) {
   Node* temp = head;
   Node* prev = NULL;


   // Jika node yang dihapus adalah head
   if (temp != NULL && temp->data == nilai) {
      head = temp->next;
      delete temp;
      return;
   }
```

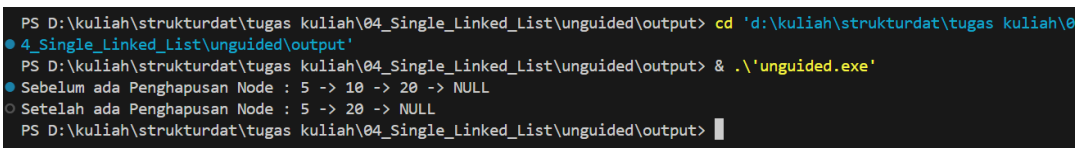
```
    // Mencari node yang akan dihapus
    while (temp != NULL && temp->data != nilai) {
        prev = temp;
        temp = temp->next;
    }

    // Jika node tidak ditemukan
    if (temp == NULL) return;

    // Hapus node
    prev->next = temp->next;
    delete temp;
}

int main()
{
    insertDepan(10);
    insertBelakang(20);
    insertDepan(5);
    cout <<"Sebelum ada Penghapusan Node : ";
    cetakLinkedList(); // Output: 5 -> 10 -> 20 -> NULL
    hapusNode(10);
    cout <<"Setelah ada Penghapusan Node : ";
    cetakLinkedList();
    return 0;
}
```

Output 1 dan 2 :



3.

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

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

Node* head = NULL;

void insertDepan(int nilai) {
   Node* newNode = new Node();
   newNode->data = nilai;
   newNode->next = head;
   head = newNode;
}

void insertBelakang(int nilai) {
   Node* newNode = new Node();
   newNode->data = nilai;
   newNode->next = NULL;
   if (head == NULL) {
      head = newNode;
   } else {
      Node* temp = head;
      while (temp->next != NULL) {
         temp = temp->next;
      }
      temp->next = newNode;
   }
}
```

```
bool cariNode(int nilai) {
    Node* temp = head;
    while (temp != NULL) {
        if (temp->data == nilai) {
            return true;
        }
        temp = temp->next;
    }
    return false;
}

int panjangLinkedList() {
    int panjang = 0;
    Node* temp = head;
    while (temp != NULL) {
        panjang++;
        temp = temp->next;
    }
    return panjang;
}

void cetakLinkedList() {
    Node* temp = head;
    while (temp != NULL) {
        cout << temp->data << " -> ";
        temp = temp->next;
    }
    cout << "NULL" << endl;
}
```

```
bool cariNode(int nilai) {
    Node* teint main() {
    insertDepan(10);
    insertBelakang(20);
    insertDepan(5);
    if (cariNode(20)) {
        cout << "Node dengan nilai 20 ditemukan." << endl;
    } else {
        cout << "Node dengan nilai 20 tidak ditemukan." << endl;
    }
    cout << "Panjang linked list: " << panjangLinkedList() << endl;
    cetakLinkedList(); // Output: 5 -> 10 -> 20 -> NULL
    return 0;
}




mp = head;
    while (temp != NULL) {
        if (temp->data == nilai) {
            return true;
        }
        temp = temp->next;
    }
    return false;
}
```

```
int panjangLinkedList() {
    int panjang = 0;
    Node* temp = head;
    while (temp != NULL) {
        panjang++;
        temp = temp->next;
    }
    return panjang;
}

void cetakLinkedList() {
    Node* temp = head;
    while (temp != NULL) {
        cout << temp->data << " -> ";
        temp = temp->next;
    }
    cout << "NULL" << endl;
}
```

Output :

```
PS D:\kuliah\strukturdat\tugas kuliah\04_Single_Linked_List\unguided\output> cd 'd:\kuliah\strukturdat\tugas kuliah\0
4_Single_Linked_List\unguided\output'
PS D:\kuliah\strukturdat\tugas kuliah\04_Single_Linked_List\unguided\output> & .\'unguided_03.exe'
Node dengan nilai 20 ditemukan.
Panjang linked list: 3
5 -> 10 -> 20 -> NULL
PS D:\kuliah\strukturdat\tugas kuliah\04_Single_Linked_List\unguided\output>
```

Penjelasan Unguided :

1. Membuat Single Linked List: Program ini membuat linked list dengan tiga operasi dasar yaitu menambah node di depan, menambah node di belakang, dan mencetak isi linked list.
2. Menghapus Node pada Linked List: Program ini dapat menghapus node tertentu berdasarkan nilai yang diberikan, serta mencetak ulang isi linked list setelah penghapusan.
3. Mencari dan Menghitung Panjang Linked List: Program ini mencari node dengan nilai tertentu dalam linked list dan menghitung jumlah total node yang ada.