LAPORAN PRAKTIKUM STRKTUR DATA DAN PEMROGRAMAN

MODUL 4

LINKED LIST CIRCULAR DAN NON CIRCULAR

Disusun Oleh :

Baharuddin Barkah

Pratama

2311102321

IF-11-A

Dosen Pengampu :

Wahyu Andi Saputra, S. Pd., M. Eng

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

PURWOKERTO

2024

BAB I

TUJUAN PRAKTIKUM

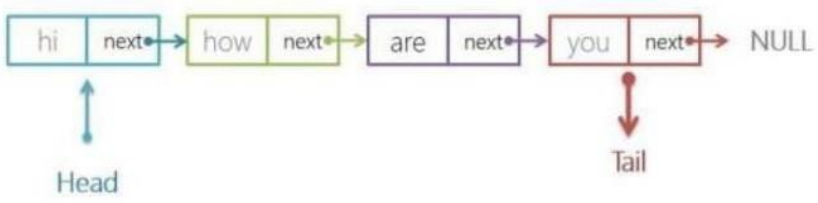
a. Praktikan dapat mengetahui dan memahami linked list circular dan non circular.
b. Praktikan dapat membuat linked list circular dan non circular.
c. Praktikan dapat mengaplikasikan atau menerapkan linked list circular dan non circular pada program yang dibuat.

BAB II

DASAR TEORI

1. Linked List Non Sircular

Linked list non circular merupakan *linked list* dengan node pertama (head) dan node terakhir (tail) yang tidak saling terhubung. Pointer terakhir (tail) pada *Linked List* ini selalu bernilai '***NULL***' sebagai pertanda data terakhir dalam *list*-nya. *Linked list non circular* dapat digambarkan sebagai berikut.



Gambar 1 *Single Linked List Non Circular*

OPERASI PADA LINKED LIST NON CIRCULAR

1. Deklarasi Simpul (Node)

```
Struct node

{

     int data;
     node *next;

};
```

2. Membuat dan Menginisialisasi Pointer Head dan Tail

```
node *head, *tail;
void init()
{

     head = NULL;
     tail = NULL;

};
```

3. Pengecekan Kondisi Linked List

```
bool isEmpety()
{

     if (head == NULL && tail == NULL)
     {
         return true;
     }
```

```
     Else
    {
        return false;
    }

}
```

4. Penambahan simpul (Node)

```
void insertBelakang(string dataUser)
{

     If (isEmpety() == true)
     {

         node *baru = new node;
         baru->data = dataUser;
         head = baru;
         tail = baru;
         baru->next = NULL;

     }

     else
     {

         node *baru = new node;
         baru->data = data User;
         baru->next = NULL;
         tail->next = baru;
         tail = baru;

     }

};
```

5. Penghapusan Simpul (Node)

```
void hapusDepan()
{

     if (isEmpty() == true)
     {
         cout << "List kosong!" << endl;
     }
     else
     {
         node *helper;
         helper  =  head;
         if (head == tail)

         {

             head = NULL;
             tail = NULL;
             delete helper;
         }
```

```
            else
                head = head->next;
                helper->next = NULL;
                delete helper;
        }
    }
}
```

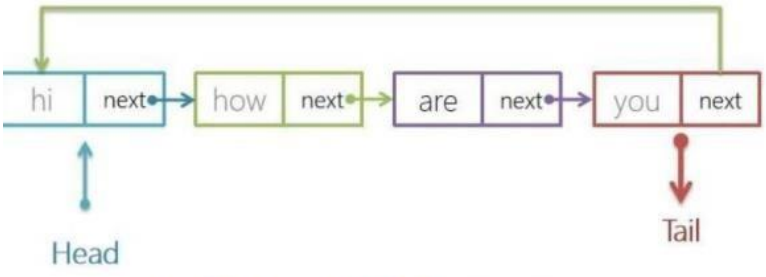
6. Tampil Data Linked List

```
void tampil()
{

    if (isEmpty() == true)
    {
        cout << "List kosong!" << endl;
    }
    Else

    {

      node *helper;
      helper = head;
      while (helper != NULL)
      {
          cout << helper->data << ends;
          helper = helper->next;
      }
    }
}
```

2. Linked List Circular

Linked list circular merupakan *linked list* yang tidak memiliki akhir karena node terakhir (tail) tidak bernilai '***NULL***', tetapi terhubung dengan node pertama (head). Saat menggunakan *linked list circular* kita membutuhkan *dummy node* atau node pengecoh yang biasanya dinamakan dengan node *current* supaya program dapat berhenti menghitung data ketika node *current* mencapai node pertama (head).

Linked list circular dapat digunakan untuk menyimpan data yang perlu diakses secara berulang, seperti daftar putar lagu, daftar pesan dalam antrian, atau penggunaan memori berulang dalam suatu aplikasi. *Linked list circular* dapat digambarkan sebagai berikut



Gambar 2 *Single Linked List Circular*

1. **Deklarasi Simpul (Node)**

```
struct Node

{
    string data;
    Node *next;

};
```

2. **Membuat dan menginisialisasi Pointer Head dan Tail**

```
Node *head, *tail, *baru, *bantu, *hapus;

void init()
{

     head = NULL;
     tail = head;

}
```

3. **Pengecekan Kondisi Linked List**

```
int isEmpty()
{

    if (head == NULL)
        return 1; // true
    else
        return 0; // false

}
```

4. **Pembuatan Simpul (Node)**

```
void buatNode(string data)
{

     baru = new Node;
     baru->data = data;
     baru->next = NULL;

}
```

5. **Penambahan Simpul (Node)**

```
// Tambah Depan
void insertDepan(string data)
{
```

```
    // Buat Node baru
    buatNode(data);

    if (isEmpty() == 1)
    {

        head = baru;
        tail = head;
        baru->next = head;

    }
    else
    {

        while (tail->next != head)
        {
            tail = tail->next;
        }
        baru->next = head;
        head = baru;
        tail->next = head;
    }
}
```

6. Penghapus Simpul (Node)

```
void hapusBelakang()
{

    if (isEmpty() == 0)
    {
        hapus = head;
        tail = head;
        if (hapus->next == head)
        {
            head = NULL;
            tail = NULL;

            delete hapus;
        }
        else
        {

            while (hapus->next != head)
            {
                hapus = hapus->next;
            }
            while (tail->next != hapus)
            {
                tail = tail->next;
            }
            tail->next = head;
            hapus->next = NULL;

            delete hapus;
        }
}
```

7. Menampilkan Data Linked List

```
void tampil()
{
    if (isEmpty() == 0)
    {
        tail = head;
        do
        {
            cout << tail->data << ends;
            tail = tail->next;
        } while (tail != head);
        cout << endl;
    }
}
```

BAB III
GUIDED

GUIDED 1
SOURCE CODE

```
#include <iostream>

using namespace std;

// PROGRAM SINGLE LINKED LIST NON-CIRCULAR

// Deklarasi struct node
struct Node
{
    int data;
    Node *next;
};

Node *head; // Deklarasi head
Node *tail; // Deklarasi tail

// Inisialisasi Node
void init()
{
    head = NULL;
    tail = NULL;
}

// Pengecekkan apakah linked list kosong
bool isEmpty()
{
    if (head == NULL)
    {
        return true;
    }
    else
    {
        return false;
    }
}

// Tambah depan
void insertDepan(int nilai)
{

    // buat node baru
    Node *baru = new Node();
    baru->data = nilai;
    baru->next = NULL;
    if (isEmpty() == true)
    {
        head = tail = baru;
        head->next = NULL;
    }
    else
    {
        baru->next = head;
        head = baru;
    }
}
```

```
// Tambah belakang
void insertBelakang(int nilai)
{
    // buat node baru
    Node *baru = new Node();
    baru->data = nilai;
    baru->next = NULL;
    if (isEmpty() == true)
    {
        head = tail = baru;
        head->next = NULL;
    }
    else
    {
        tail->next = baru;
        tail = baru;
    }
}


// Hitung jumlah list
int hitungList()
{
    Node *hitung;
    hitung = head;
    int jumlah = 0;
    while (hitung != NULL)
    {
        jumlah++;
        hitung = hitung->next;
    }
    return jumlah;
}

// Tambah tengah
void insertTengah(int data, int posisi)
{
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "Posisi di luar jangkauan" << endl;
    }
    else if (posisi == 1)
    {
        cout << "Posisi bukan posisi tengah" << endl;
    }
    else
    {
        Node *baru, *bantu;
        baru = new Node();
        baru->data = data;

        // tranversing
        bantu = head;
        int nomor = 1;
        while (nomor < posisi - 1)
        {
            bantu = bantu->next;
            nomor++;
        }

        baru->next = bantu->next;
```

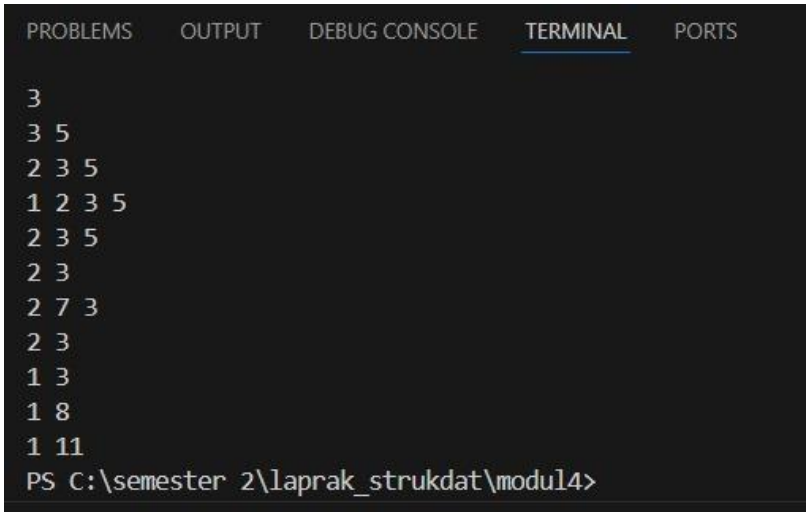
```
            bantu->next = baru;
        }
    }

    // Hapus depan
    void hapusDepan()
    {
        Node *hapus;
        if (isEmpty() == false)
        {
            if (head->next != NULL)
            {
                hapus = head;
                head = head->next;
                delete hapus;
            }
            else
            {
                head = tail = NULL;
            }
        }
        else
        {
            cout << "Linked list masih kosong" << endl;
        }
    }

    // Hapus belakang
    void hapusBelakang()
    {
        Node *hapus;
        Node *bantu;
        if (isEmpty() == false)
        {
            if (head != tail)
            {
                hapus = tail;
                bantu = head;
                while (bantu->next != tail)
                {
                    bantu = bantu->next;
                }
                tail = bantu;
                tail->next = NULL;
                delete hapus;
            }
            else
            {
                head = tail = NULL;
            }
        }
        else
        {
            cout << "Linked list masih kosong" << endl;
        }
    }
    // Hapus tengah
    void hapusTengah(int posisi)
    {
        Node *hapus, *bantu, *sebelum;
        if (posisi < 1 || posisi > hitungList())
```

```
        {
            cout << "Posisi di luar jangkauan" << endl;
        }
        else if (posisi == 1)
        {
            cout << "Posisi bukan posisi tengah" << endl;
        }
        else
        {
            int nomor = 1;
            bantu = head;
            while (nomor <= posisi)
            {
                if (nomor == posisi - 1)
                {
                    sebelum = bantu;
                }
                if (nomor == posisi)
                {
                    hapus = bantu;
                }
                bantu = bantu->next;
                nomor++;
            }
            sebelum->next = bantu;
            delete hapus;
        }
    }

    // ubah depan
    void ubahDepan(int data)
    {
        if (isEmpty() == 0)
        {
            head->data = data;
        }
        else
        {
            cout << "Linked list masih kosong" << endl;
        }
    }


    // ubah tengah
    void ubahTengah(int data, int posisi)
    {
        Node *bantu;
        if (isEmpty() == 0)
        {
            if (posisi < 1 || posisi > hitungList())
            {
                cout << "Posisi di luar jangkauan" << endl;
            }
            else if (posisi == 1)
            {
                cout << "Posisi bukan posisi tengah" << endl;
            }
            else
            {
                int nomor = 1;
                bantu = head;
                while (nomor < posisi)
```

```
                {
                    bantu = bantu->next;
                    nomor++;
                }
                bantu->data = data;
            }
        }
        else
        {
            cout << "Linked list masih kosong" << endl;
        }
    }

    // ubah belakang
    void ubahBelakang(int data)
    {
        if (isEmpty() == 0)
        {
            tail->data = data;
        }
        else
        {
            cout << "Linked list masih kosong" << endl;
        }
    }


    // Hapus list
    void clearList()
    {
        Node *bantu, *hapus;
        bantu = head;
        while (bantu != NULL)
        {
            hapus = bantu;
            bantu = bantu->next;
            delete hapus;
        }
        head = tail = NULL;
        cout << "List berhasil terhapus!" << endl;
    }

    // Tampilkan list
    void tampilList()
    {
        Node *bantu;
        bantu = head;
        if (isEmpty() == false)
        {
            while (bantu != NULL)
            {
                cout << bantu->data << " ";
                bantu = bantu->next;
            }
            cout << endl;
        }
        else
        {
            cout << "Linked list masih kosong" << endl;
        }
    }
```

```
int main()
{
    init();
    insertDepan(3);
    tampilList();
    insertBelakang(5);
    tampilList();
    insertDepan(2);
    tampilList();
    insertDepan(1);
    tampilList();
    hapusDepan();
    tampilList();
    hapusBelakang();
    tampilList();
    insertTengah(7, 2);
    tampilList();
    hapusTengah(2);
    tampilList();
    ubahDepan(1);
    tampilList();
    ubahBelakang(8);
    tampilList();
    ubahTengah(11, 2);
    tampilList();

    return 0;
}
```

SCREENSHOOT OUTPUT



DESKRIPSI PROGRAM

Program ini mengimplementasikan linked list untuk menyimpan nilai integer. Setiap node dalam list memiliki dua elemen: data yang menyimpan nilai integer dan pointer 'next' yang menunjuk ke node berikutnya. Program ini menyediakan fungsi untuk melakukan berbagai operasi pada linked list, seperti:

Menambahkan node: Node baru dapat ditambahkan di depan, di belakang, atau di tengah list.
Menghapus node: Node dapat dihapus dari depan, belakang, atau tengah list.
Mengubah nilai node: Nilai node yang ada di depan, belakang, atau tengah list dapat diubah.

Menghitung jumlah node: Fungsi ini menghitung dan mengembalikan jumlah node dalam list.
Membersihkan list: Fungsi ini menghapus semua node dari list.
Menampilkan list: Fungsi ini menampilkan nilai dari semua node dalam list.

GUIDED 2
SOURCE CODE

```
#include <iostream>
using namespace std;
/// PROGRAM SINGLE LINKED LIST CIRCULAR
// Deklarasi Struct Node
struct Node
{
    string data;
    Node *next;
};
Node *head, *tail, *baru, *bantu, *hapus;
void init()
{
    head = NULL;
    tail = head;
}
// Pengecekan
int isEmpty()
{
    if (head == NULL)
        return 1; // true
    else
        return 0; // false
}
// Buat Node Baru
void buatNode(string data)
{
    baru = new Node;
    baru->data = data;
    baru->next = NULL;
}
// Hitung List
int hitungList()
{
    bantu = head;
    int jumlah = 0;
    while (bantu != NULL)
    {
        jumlah++;
        bantu = bantu->next;
    }
    return jumlah;
}
// Tambah Depan
void insertDepan(string data)
{
    // Buat Node baru
    buatNode(data);
    if (isEmpty() == 1)
    {
        head = baru;
        tail = head;
        baru->next = head;
```

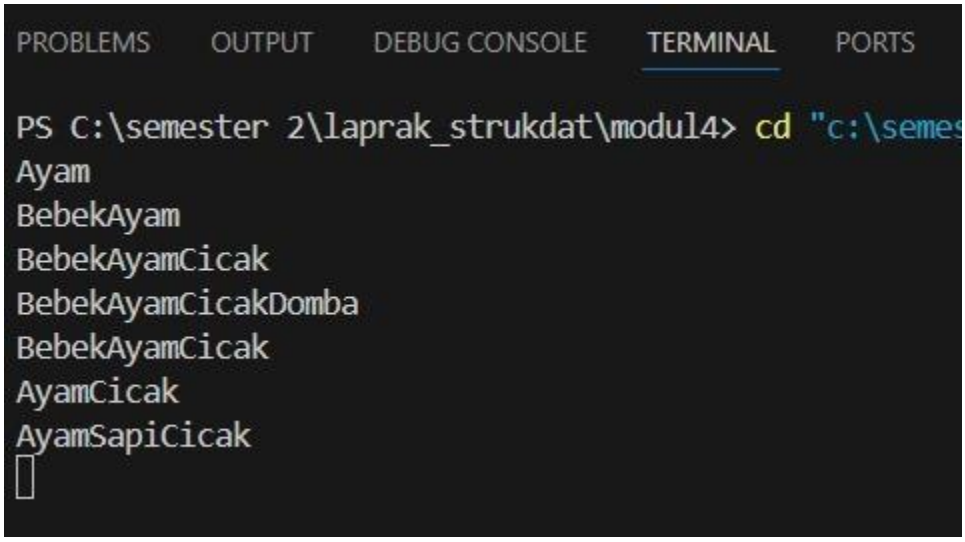
```
        }
        else
        {
            while (tail->next != head)
            {
                tail = tail->next;
            }
            baru->next = head;
            head = baru;
            tail->next = head;
        }
    }
    // Tambah Belakang
    void insertBelakang(string data)
    {
        // Buat Node baru
        buatNode(data);
        if (isEmpty() == 1)
        {
            head = baru;
            tail = head;
            baru->next = head;
        }
        else
        {
            while (tail->next != head)
            {
                tail = tail->next;
            }
            tail->next = baru;
            baru->next = head;
        }
    }
    // Tambah Tengah
    void insertTengah(string data, int posisi)
    {
        if (isEmpty() == 1)
        {
            head = baru;
            tail = head;
            baru->next = head;
        }
        else
        {
            baru->data = data;
            // transversing
            int nomor = 1;
            bantu = head;
            while (nomor < posisi - 1)
            {
                bantu = bantu->next;
                nomor++;
            }
            baru->next = bantu->next;
            bantu->next = baru;
        }
    }
    // Hapus Depan
    void hapusDepan()
    {
        if (isEmpty() == 0)
```

```
        {
            hapus = head;
            tail = head;
            if (hapus->next == head)
            {
                head = NULL;
                tail = NULL;
                delete hapus;
            }
            else
            {

                while (tail->next != hapus)
                {
                    tail = tail->next;
                }
                head = head->next;
                tail->next = head;
                hapus->next = NULL;
                delete hapus;
            }
        }
        else
        {

            cout << "List masih kosong!" << endl;
        }
    }
    // Hapus Belakang
    void hapusBelakang()
    {
        if (isEmpty() == 0)
        {
            hapus = head;
            tail = head;
            if (hapus->next == head)
            {
                head = NULL;
                tail = NULL;
                delete hapus;
            }
            else
            {

                while (hapus->next != head)
                {
                    hapus = hapus->next;
                }
                while (tail->next != hapus)
                {
                    tail = tail->next;
                }
                tail->next = head;
                hapus->next = NULL;
                delete hapus;
            }
        }
        else
        {

            cout << "List masih kosong!" << endl;
        }
    }
```

```
// Hapus Tengah
void hapusTengah(int posisi)
{
    if (isEmpty() == 0)
    {
        // transversing
        int nomor = 1;
        bantu = head;
        while (nomor < posisi - 1)
        {
            bantu = bantu->next;
            nomor++;
        }
        hapus = bantu->next;
        bantu->next = hapus->next;
        delete hapus;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

// Hapus List
void clearList()
{
    if (head != NULL)
    {
        hapus = head->next;
        while (hapus != head)
        {
            bantu = hapus->next;
            delete hapus;
            hapus = bantu;
        }
        delete head;
        head = NULL;
    }
    cout << "List berhasil terhapus!" << endl;
}
// Tampilkan List
void tampil()
{
    if (isEmpty() == 0)
    {
        tail = head;
        do
        {
            cout << tail->data << ends;
            tail = tail->next;
        } while (tail != head);
        cout << endl;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

int main()
{
    init();
    insertDepan("Ayam");
```

```
            tampil();
            insertDepan("Bebek");
            tampil();
            insertBelakang("Cicak");
            tampil();
            insertBelakang("Domba");
            tampil();
            hapusBelakang();
            tampil();
            hapusDepan();
            tampil();
            insertTengah("Sapi", 2);
            tampil();
            hapusTengah(2);
            tampil();
            return 0;
}
```

SCREENSHOOT PROGRAM



DESKRIPSI PROGRAM

Inisialisasi: Fungsi init() digunakan untuk menginisialisasi linked list dengan menset head dan tail ke NULL.
Pengecekan: Fungsi isEmpty() digunakan untuk mengecek apakah linked list kosong.
Membuat Node Baru: Fungsi buatNode() digunakan untuk membuat node baru dengan data yang diberikan dan pointer next yang diset ke NULL.
Menghitung List: Fungsi hitungList() digunakan untuk menghitung jumlah node dalam linked list.
Menambah Node:
insertDepan() : Menambahkan node baru di depan linked list.
insertBelakang() : Menambahkan node baru di belakang linked list.
insertTengah() : Menambahkan node baru di posisi tertentu dalam linked list.
Menghapus Node:
hapusDepan() : Menghapus node pertama dari linked list.
hapusBelakang() : Menghapus node terakhir dari linked list.
hapusTengah() : Menghapus node pada posisi tertentu dalam linked list.
Membersihkan List: Fungsi clearList() digunakan untuk menghapus semua node dari linked list.

BAB IV
UNGUIDED

UNGUIDED 1
SOURCE CODE

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

struct Node {
    string nama;
    string nim;
    Node* next;


};

class LinkedList {
private:
    Node* head;

public:
    LinkedList()  {
        head = nullptr;
    }

    void tambahDepan(string nama, string nim) {
        Node* newNode = new Node;
        newNode->nama = nama;
        newNode->nim = nim;
        newNode->next = head;
        head = newNode;
        cout << "Data telah ditambahkan" << endl;
    }

    void tambahBelakang(string nama, string nim) {
        Node* newNode = new Node;
        newNode->nama = nama;
        newNode->nim = nim;
        newNode->next = nullptr;
        if (head == nullptr) {
            head = newNode;
            return;
        }
        Node* temp = head;
        while (temp->next != nullptr) {
            temp = temp->next;
        }
        temp->next = newNode;
        cout << "Data telah ditambahkan" << endl;
    }

    void tambahTengah(string nama, string nim, int posisi) {
        if (posisi <= 0) {
            cout << "Posisi tidak valid" << endl;
            return;
        }
        Node* newNode = new Node;
        newNode->nama = nama;
        newNode->nim = nim;
        Node* temp = head;
        for (int i = 0; i < posisi - 1; i++) {
            if (temp == nullptr) {
```

```
                    cout << "Posisi tidak valid" << endl;
                    return;
                }
                temp = temp->next;
            }
            if (temp == nullptr) {
                cout << "Posisi tidak valid" << endl;
                return;
            }
            newNode->next = temp->next;
            temp->next = newNode;
            cout << "Data telah ditambahkan" << endl;
        }

        void hapusDepan() {
            if (head == nullptr) {
                cout << "Linked list kosong" << endl;
                return;
            }
            Node* temp = head;
            head = head->next;
            delete temp;
            cout << "Data berhasil dihapus" << endl;
        }

        void hapusBelakang() {
            if (head == nullptr) {
                cout << "Linked list kosong" << endl;
                return;
            }
            if (head->next == nullptr) {
                delete head;
                head = nullptr;
                cout << "Data berhasil dihapus" << endl;
                return;
            }
            Node* temp = head;
            while (temp->next->next != nullptr) {
                temp = temp->next;
            }
            delete temp->next;
            temp->next = nullptr;
            cout << "Data berhasil dihapus" << endl;
        }

        void hapusTengah(int posisi) {
            if (posisi <= 0 || head == nullptr) {
                cout << "Linked list kosong atau posisi tidak valid" <<
    endl;
                return;
            }
            if (posisi == 1) {
                hapusDepan();
                return;
            }
            Node* temp = head;
            for (int i = 0; i < posisi - 2; i++) {
                if (temp->next == nullptr) {
                    cout << "Posisi tidak valid" << endl;
                    return;
                }
```

```
                temp = temp->next;
            }
            if (temp->next == nullptr) {
                cout << "Posisi tidak valid" << endl;
                return;
            }
            Node* nodeToDelete = temp->next;
            temp->next = temp->next->next;
            delete nodeToDelete;
            cout << "Data berhasil dihapus" << endl;
        }

         void ubahDepan(string namaBaru, string nimBaru) {
            if (head == nullptr) {
                cout << "Linked list kosong" << endl;
                return;
            }
            head->nama = namaBaru;
            head->nim = nimBaru;
            cout << "Data berhasil diubah" << endl;
        }

        void ubahBelakang(string namaBaru, string nimBaru) {
            if (head == nullptr) {
                cout << "Linked list kosong" << endl;
                return;
            }
            Node* temp = head;
            while (temp->next != nullptr) {
                temp = temp->next;
            }
            temp->nama = namaBaru;
            temp->nim = nimBaru;
            cout << "Data berhasil diubah" << endl;
        }

        void ubahTengah(string namaBaru, string nimBaru, int posisi) {
            if (posisi <= 0 || head == nullptr) {
                cout << "Linked list kosong atau posisi tidak valid" <<
    endl;
                return;
            }
            Node* temp = head;
            for (int i = 0; i < posisi - 1; i++) {
                if (temp == nullptr) {
                    cout << "Posisi tidak valid" << endl;
                    return;
                }
                temp = temp->next;
            }
            if (temp == nullptr) {
                cout << "Posisi tidak valid" << endl;
                return;
            }
            temp->nama = namaBaru;
            temp->nim = nimBaru;
            cout << "Data berhasil diubah" << endl;
        }

        void hapusList() {
            Node* current = head;
```

```
            Node* next;
            while (current != nullptr) {
                next = current->next;
                delete current;
                current = next;
            }
            head = nullptr;
            cout << "Linked list berhasil dihapus" << endl;
        }

        void tampilkanData() {
            Node* temp = head;
            cout << "DATA MAHASISWA" << endl;
            cout << "NAMA\tNIM" << endl;
            while (temp != nullptr) {
                cout << temp->nama << "\t" << temp->nim << endl;
                temp = temp->next;
            }
        }
    };

    int main() {
        LinkedList linkedList;
        int choice;
        string nama, nim;
        int posisi;

        do {
            cout << "PROGRAM SINGLE LINKED LIST NON-CIRCULAR" << endl;
            cout << "1. Tambah Depan" << endl;
            cout << "2. Tambah Belakang" << endl;
            cout << "3. Tambah Tengah" << endl;
            cout << "4. Ubah Depan" << endl;
            cout << "5. Ubah Belakang" << endl;
            cout << "6. Ubah Tengah" << endl;
            cout << "7. Hapus Depan" << endl;
            cout << "8. Hapus Belakang" << endl;
            cout << "9. Hapus Tengah" << endl;
            cout << "10. Hapus List" << endl; // Menu untuk hapus list
            cout << "11. Tampilkan Data" << endl;
            cout << "12. Keluar" << endl;
            cout << "Pilih Operasi : ";
            cin >> choice;

            switch (choice) {
                case 1:
                    cout << "-Tambah Depan-" << endl;
                    cout << "Masukkan Nama : ";
                    cin >> nama;
                    cout << "Masukkan NIM : ";
                    cin >> nim;
                    linkedList.tambahDepan(nama, nim);
                    break;
                case 2:
                    cout << "-Tambah Belakang-" << endl;
                    cout << "Masukkan Nama : ";
                    cin >> nama;
                    cout << "Masukkan NIM : ";
                    cin >> nim;
                    linkedList.tambahBelakang(nama, nim);
                    break;
```

```
                case 3:
                    cout << "-Tambah Tengah-" << endl;
                    cout << "Masukkan Nama : ";
                    cin >> nama;
                    cout << "Masukkan NIM : ";
                    cin >> nim;
                    cout << "Masukkan Posisi : ";
                    cin >> posisi;
                    linkedList.tambahTengah(nama, nim, posisi);
                    break;
                case 4:
                    cout << "-Ubah Depan-" << endl;
                    cout << "Masukkan Nama Baru : ";
                    cin >> nama;
                    cout << "Masukkan NIM Baru : ";
                    cin >> nim;
                    linkedList.ubahDepan(nama, nim);
                    break;
                case 5:
                    cout << "-Ubah Belakang-" << endl;
                    cout << "Masukkan Nama Baru : ";
                    cin >> nama;
                    cout << "Masukkan NIM Baru : ";
                    cin >> nim;
                    linkedList.ubahBelakang(nama, nim);
                    break;
                case 6:
                    cout << "-Ubah Tengah-" << endl;
                    cout << "Masukkan Nama Baru : ";
                    cin >> nama;
                    cout << "Masukkan NIM Baru : ";
                    cin >> nim;
                    cout << "Masukkan Posisi : ";
                    cin >> posisi;
                    linkedList.ubahTengah(nama, nim, posisi);
                    break;
                case 7:
                    linkedList.hapusDepan();
                    break;
                case 8:
                    linkedList.hapusBelakang();
                    break;
                case 9:
                    cout << "-Hapus Tengah-" << endl;
                    cout << "Masukkan Posisi : ";
                    cin >> posisi;
                    linkedList.hapusTengah(posisi);
                    break;
                case 10:
                    linkedList.hapusList(); // Hapus List
                    break;
                case 11:
                    linkedList.tampilkanData();
                    break;
                case 12:
                    cout << "Program selesai." << endl;
                    break;
                default:
                    cout << "Pilihan tidak valid." << endl;
            }
        } while (choice != 12);
```

```
        return 0;
}
```

SCREENSHOOT CODE

Tampilan Operasi

Tampil Data

```
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan
5. Ubah Belakang
6. Ubah Tengah
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10. Hapus List
11. Tampilkan Data
12. Keluar
Pilih Operasi : 1
-Tambah Depan-
Masukkan Nama : BaharuddinBarkahPratama
Masukkan NIM : 2311102321
Data telah ditambahkan
```

- Setelah membuat menu tersebut, masukkan data sesuai urutan berikut, lalu tampilkan data yang telah dimasukkan. (Gunakan insert depan, belakang atau tengah)

```
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan
5. Ubah Belakang
6. Ubah Tengah
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10. Hapus List
11. Tampilkan Data
12. Keluar
Pilih Operasi : 11
DATA MAHASISWA
NAMA    NIM
Lisa    2311102121
Jokowi  2311109998
Rocky   2311102143
Ryan    2311108976
Tommy   2311103434
Naruto  2311103332
BaharuddinBarkahPratama 2311102321
```

Lakukan perintah berikut:

a. Tambahkan data berikut diantara Tommy dan Naruto:
Bobby 2311109999

```
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan
5. Ubah Belakang
6. Ubah Tengah
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10. Hapus List
11. Tampilkan Data
12. Keluar
Pilih Operasi : 3
-Tambah Tengah-
Masukkan Nama : Bobby
Masukkan NIM : 2311109999
Masukkan Posisi : 4
Data telah ditambahkan
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan
5. Ubah Belakang
6. Ubah Tengah
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10. Hapus List
11. Tampilkan Data
12. Keluar
Pilih Operasi : 11
DATA MAHASISWA
NAMA    NIM
Lisa    2311102121
Jokowi  2311109998
Rocky   2311102143
Tommy   2311103434
Bobby   2311109999
Naruto  2311103332
BaharuddinBarkahPratama 2311102321
```

b. Hapus Data Lisa

```
DATA MAHASISWA
NAMA    NIM
Lisa    2311102121
Jokowi  2311109998
Rocky   2311102143
Tommy   2311103434
Bobby   2311109999
Naruto  2311103332
hyoki   23111109932
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan
5. Ubah Belakang
6. Ubah Tengah
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10. Hapus List
11. Tampilkan Data
12. Keluar
Pilih Operasi : 9
-Hapus Tengah-
Masukkan Posisi : 1
Data berhasil dihapus
```

```
PROGRAM SINGLE LINKED LIST NON-CIRCULAR
1. Tambah Depan
2. Tambah Belakang
3. Tambah Tengah
4. Ubah Depan
5. Ubah Belakang
6. Ubah Tengah
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10. Hapus List
11. Tampilkan Data
12. Keluar
Pilih Operasi : 11
DATA MAHASISWA
NAMA    NIM
Jokowi  2311109998
Rocky   2311102143
Tommy   2311103434
Bobby   2311109999
Naruto  2311103332
hyoki   23111109932
```

DESKRIPSI PROGRAM

program memungkinkanini pengguna untuk melakukan banyak operasi pada daftar tertaut , seperti augmentasi datadaftar padaseperti augmentasi data pada node, augmentasi data pada belakang, data tengah, dan data depan, belakang, dan data tengah, augmentasi data pada node, dan pembesaran data pada semua daftar tertaut .node, augmentasi data di belakang, data tengah, dan data depan, belakang, danaugmentasi data pada node,pembesaran pada semua daftar tertaut . Setiap operasi dapat diakses oleh penggunaoleh pengguna melalui menu yang ditampilkan secara berulang , dan program akan terus berjalan hingga pengguna memilih untuk keluar .melaluimenu yang ditampilkan dalam satu lingkaran , dan program akan terus berjalan hingga pengguna memilih untuk keluar .