

# **LAPORAN PRAKTIKUM**

## **MODUL 4 LINKED LIST CIRCULAR DAN NON CIRCULAR**



**Disusun oleh:  
Tegar Bangkit Wijaya  
NIM: 2311102027**

**Dosen Pengampu:**  
Wahyu Andi Saputra, S.Pd., M.Eng

**PROGRAM STUDI TEKNIK INFORMATIKA  
FAKULTAS INFORMATIKA  
INSTITUT TEKNOLOGI TELKOM PURWOKERTO  
PURWOKERTO  
2023**

## **BAB I**

### **TUJUAN PRAKTIKUM**

- 1. Mahasiswa dapat memahami linked list circular dan non circular**
- 2. Mahasiswa dapat membuat linked list dan non circular**
- 3. Mahasiswa dapat menerapkan linked list circular dan non circular**

## **BAB II**

### **DASAR TEORI**

#### **1. Linked list non-circular**

Pada linked list non-circular, node terakhir tidak menunjuk pada simpul pertama. Artinya, linked list ini memiliki simpul terakhir yang menunjuk ke null atau tidak menunjuk ke simpul mana pun. Dalam linked list non-circular, traversal atau penelusuran dari awal hingga akhir dapat dilakukan dengan mudah dengan mengikuti alamat setiap node.

#### **2. Linked list circular**

Pada linked list circular, simpul terakhir menunjuk pada simpul pertama. Artinya, linked list ini membentuk sebuah lingkaran, dan traversal dapat dimulai dari mana saja dalam linked list ini. Linked list circular memiliki kelebihan yaitu memungkinkan untuk melakukan traversal dari mana saja dan tidak memerlukan operasi tambahan untuk mengembalikan pointer ke simpul awal.

Dalam penggunaannya Linked list circular dan Linked list non circular memiliki beberapa perbedaan. Salah satunya adalah pada penambahan dan penghapusan node, dimana linked list circular memerlukan operasi penambahan atau penghapusan node, dimana linked list circular memerlukan operasi tambahan untuk mengubah pointer pada simpul terakhir agar menunjuk pada simpul pertama. Sedangkan linked list non circular tidak memerlukan operasi tersebut karena simpul terakhirnya tidak menunjuk ke simpul pertama.

## BAB III

### GUIDED

#### 1. 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;
Node *tail;
// Inisialisasi Node
void init()
{
    head = NULL;
    tail = NULL;
}
// Pengecekan
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;
        tail->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;
        tail->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 diluar 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 << "List 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 << "List kosong!" << endl;
    }
}
// Hapus Tengah
void hapusTengah(int posisi)
{
    Node *bantu, *hapus, *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 << "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)
        {
        }
        else
        {
            cout << "Posisi bukan posisi tengah" << endl;
            bantu = head;
            int nomor = 1;
            while (nomor < posisi)
            {
                bantu = bantu->next;
                nomor++;
            }
            bantu->data = data;
        }
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}

// Ubah Belakang
void ubahBelakang(int data)
{
    if (isEmpty() == 0)
    {
        tail->data = data;
    }
    else
    {
        cout << "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 tampil()
{
    Node *bantu;
    bantu = head;
    if (isEmpty() == false)
    {
        while (bantu != NULL)
        {
            cout << bantu->data << ends;
            bantu = bantu->next;
        }
        cout << endl;
    }
    else
    {
        cout << "List masih kosong!" << endl;
    }
}
}
int main()
{
    init();
    insertDepan(3);
    tampil();
    insertBelakang(5);
}

```

```

    tampil();
    insertDepan(2);
    tampil();
    insertDepan(1);
    tampil();
    hapusDepan();
    tampil();
    hapusBelakang();
    tampil();
    insertTengah(7, 2);
    tampil();
    hapusTengah(2);
    tampil();
    ubahDepan(1);
    tampil();
    ubahBelakang(8);
    tampil();
    ubahTengah(11, 2);
    tampil();
    return 0;
}

```

### Screenshoot program

```

PS C:\praktikum struktur data\Modul4> cd "c:\praktikum struktur data\Modul4\" ; if ($?) { g++ Modul4_guided1.cpp -o Modul4_guided1 } ; if ($?) {
.\Modul4_guided1 }
3
35
235
1235
235
23
273
23
13
18

```

### Deskripsi program

Program ini menangani operasi dasar seperti penambahan dan penghapusan node di awal dan akhir, serta modifikasi data di berbagai posisi.

Setiap node hanya memiliki pointer ke node berikutnya.

Untuk mengubah menjadi circular linked list, diperlukan penyesuaian agar node terakhir menunjuk kembali ke node pertama, membentuk siklus.

Logika penampilan data harus dimodifikasi agar tidak masuk ke dalam perulangan tak terbatas

## 2. Guided 2

```
3. #include <iostream>
4.
5. using namespace std;
6.
7. // Deklarasi Struct Node
8.
9. struct Node
10.{
11.    string data;
12.    Node *next;
13.};
14.
15.Node *head, *tail, *baru, *bantu, *hapus;
16.
17.// Inisialisasi node head & tail
18.void init()
19.{
20.    head = NULL;
21.    tail = head;
22.}
23.
24.// Pengecekan isi list
25.int isEmpty()
26.{
27.    if (head == NULL)
28.    {
29.        return 1; // true
30.    }
31.    else
32.    {
33.        return 0; // false
34.    }
35.}
36.
37.// Buat Node Baru
38.void buatNode(string data)
39.{
40.    baru = new Node;
```

```
41.     baru->data = data;
42.     baru->next = NULL;
43.}
44.
45.// Hitung List
46.int hitungList()
47.{
48.     bantu = head;
49.     int jumlah = 0;
50.     while (bantu != NULL)
51.     {
52.         jumlah++;
53.         bantu = bantu->next;
54.     }
55.     return jumlah;
56.}
57.
58.// Tambah Depan
59.void insertDepan(string data)
60.{
61.     // Buat Node baru
62.     buatNode(data);
63.
64.     if (isEmpty() == 1)
65.     {
66.         head = baru;
67.         tail = head;
68.         baru->next = head;
69.     }
70.     else
71.     {
72.         while (tail->next != head)
73.         {
74.             tail = tail->next;
75.         }
76.         baru->next = head;
77.         head = baru;
78.         tail->next = head;
79.     }
80.}
81.
```

```

82. // Tambah Belakang
83. void insertBelakang(string data)
84. {
85.     // Buat Node baru
86.     buatNode(data);
87.
88.     if (isEmpty() == 1)
89.     {
90.         head = baru;
91.         tail = head;
92.         baru->next = head;
93.     }
94.     else
95.     {
96.         while (tail->next != head)
97.         {
98.             tail = tail->next;
99.         }
100.         tail->next = baru;
101.         baru->next = head;
102.     }
103. }
104.
105. // Tambah Tengah
106. void insertTengah(string data, int posisi)
107. {
108.     if (isEmpty() == 1)
109.     {
110.         head = baru;
111.         tail = head;
112.         baru->next = head;
113.     }
114.     else
115.     {
116.         baru->data = data;
117.         // transversing
118.         int nomor = 1;
119.         bantu = head;
120.         while (nomor < posisi - 1)
121.         {
122.             bantu = bantu->next;

```

```

123.         nomor++;
124.     }
125.     baru->next = bantu->next;
126.     bantu->next = baru;
127. }
128. }
129.
130. // Hapus Depan
131. void hapusDepan()
132. {
133.     if (isEmpty() == 0)
134.     {
135.         hapus = head;
136.         tail = head;
137.         if (hapus->next == head)
138.         {
139.             head = NULL;
140.             tail = NULL;
141.             delete hapus;
142.         }
143.         else
144.         {
145.             while (tail->next != hapus)
146.             {
147.                 tail = tail->next;
148.             }
149.             head = head->next;
150.             tail->next = head;
151.             hapus->next = NULL;
152.             delete hapus;
153.         }
154.     }
155.     else
156.     {
157.         cout << "List masih kosong!" << endl;
158.     }
159. }
160.
161. // Hapus Belakang
162. void hapusBelakang()
163. {

```

```

164.         if (isEmpty() == 0)
165.         {
166.             hapus = head;
167.             tail = head;
168.             if (hapus->next == head)
169.             {
170.                 head = NULL;
171.                 tail = NULL;
172.                 delete hapus;
173.             }
174.             else
175.             {
176.                 while (hapus->next != head)
177.                 {
178.                     hapus = hapus->next;
179.                 }
180.                 while (tail->next != hapus)
181.                 {
182.                     tail = tail->next;
183.                 }
184.                 tail->next = head;
185.                 hapus->next = NULL;
186.                 delete hapus;
187.             }
188.         }
189.         else
190.         {
191.             cout << "List masih kosong!" << endl;
192.         }
193.     }
194.
195.     // Hapus Tengah
196.     void hapusTengah(int posisi)
197.     {
198.         if (isEmpty() == 0)
199.         {
200.             // transversing
201.             int nomor = 1;
202.             bantu = head;
203.             while (nomor < posisi - 1)
204.             {

```



```

205.         bantu = bantu->next;
206.         nomor++;
207.     }
208.     hapus = bantu->next;
209.     bantu->next = hapus->next;
210.     delete hapus;
211. }
212. else
213. {
214.     cout << "List masih kosong!" << endl;
215. }
216. }
217.
218. // Hapus List
219. void clearList()
220. {
221.     if (head != NULL)
222.     {
223.         hapus = head->next;
224.         while (hapus != head)
225.         {
226.             bantu = hapus->next;
227.             delete hapus;
228.             hapus = bantu;
229.         }
230.         delete head;
231.         head = NULL;
232.     }
233.     cout << "List berhasil terhapus!" << endl;
234. }
235.
236. // Tampilkan List
237. void tampil()
238. {
239.     if (isEmpty() == 0)
240.     {
241.         tail = head;
242.         do
243.         {
244.             cout << tail->data << ends;
245.             tail = tail->next;

```

```

246.         } while (tail != head);
247.         cout << endl;
248.     }
249.     else
250.     {
251.         cout << "List masih kosong!" << endl;
252.     }
253. }
254.
255. int main()
256. {
257.     init();
258.     insertDepan("Ayam");
259.     tampil();
260.     insertDepan("Bebek");
261.     tampil();
262.     insertBelakang("Cicak");
263.     tampil();
264.     insertBelakang("Domba");
265.     tampil();
266.     hapusBelakang();
267.     tampil();
268.     hapusDepan();
269.     tampil();
270.     insertTengah("Sapi", 2);
271.     tampil();
272.     hapusTengah(2);
273.     tampil();
274.
275.     return 0;
276. }

```

## Screenshot program

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
PS C:\praktikum struktur data\Modul4> cd "c:\praktikum struktur data\Modul4" ; if ($?) { g++ Modul4_guided2.cpp -o Modul4_guided2 } ; if ($?) {
.\Modul4_guided2 }
Ayam
BebekAyam
BebekAyamCicak
BebekAyamCicakDomba
BebekAyamCicak
AyamCicak
AyamSapiCicak
PS C:\praktikum struktur data\Modul4>

```

## **Deskripsi program**

Program di atas adalah implementasi dari struktur data Circular Linked List dalam bahasa pemrograman C++. Circular Linked List adalah jenis linked list di mana elemen terakhir mengacu kembali ke elemen pertama. Program ini dilengkapi dengan fungsi-fungsi untuk menambahkan, menghapus, dan menampilkan elemen dalam Circular Linked List. Fungsi-fungsi seperti `insertDepan()` dan `insertBelakang()` digunakan untuk menambahkan elemen di bagian depan dan belakang Circular Linked List, sementara `hapusDepan()`, `hapusBelakang()`, dan `hapusTengah()` digunakan untuk menghapus elemen dari Circular Linked List.

## LATIHAN KELAS - UNGUIDED

### 1. Unguided 1

#### Source code

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

struct Mahasiswa140
{
    string Nama140;
    string NIM140;
    Mahasiswa140 *next;
};

class LinkedList
{
private:
    Mahasiswa140 *head;

public:
    LinkedList()
    {
        head = NULL;
    }

    void tambah_depan(string nama, string nim)
    {
        Mahasiswa140 *new_mahasiswa = new Mahasiswa140;
        new_mahasiswa->Nama140 = nama;
        new_mahasiswa->NIM140 = nim;
        new_mahasiswa->next = head;
        head = new_mahasiswa;
    }

    void tambah_belakang(string nama, string nim)
    {
        Mahasiswa140 *new_mahasiswa = new Mahasiswa140;
```

```

        new_mahasiswa->Nama140 = nama;
        new_mahasiswa->NIM140 = nim;
        new_mahasiswa->next = NULL;
        if (head == NULL)
        {
            head = new_mahasiswa;
            return;
        }
        Mahasiswa140 *current = head;
        while (current->next != NULL)
        {
            current = current->next;
        }
        current->next = new_mahasiswa;
    }

void tambah_tengah(int posisi, string nama, string nim)
{
    if (posisi <= 1)
    {
        tambah_depan(nama, nim);
        return;
    }
    Mahasiswa140 *new_mahasiswa = new Mahasiswa140;
    new_mahasiswa->Nama140 = nama;
    new_mahasiswa->NIM140 = nim;
    Mahasiswa140 *current = head;
    for (int i = 1; i < posisi - 1 && current != NULL; i++)
    {
        current = current->next;
    }
    if (current != NULL)
    {
        new_mahasiswa->next = current->next;
        current->next = new_mahasiswa;
    }
    else
    {
        cout << "Posisi tidak valid." << endl;
    }
}

```

```

void hapus_belakang()
{
    if (head == NULL)
    {
        cout << "Linked list kosong." << endl;
        return;
    }
    if (head->next == NULL)
    {
        delete head;
        head = NULL;
        return;
    }
    Mahasiswa140 *current = head;
    while (current->next->next != NULL)
    {
        current = current->next;
    }
    delete current->next;
    current->next = NULL;
}

void hapus_tengah(int posisi)
{
    if (posisi <= 1)
    {
        Mahasiswa140 *temp = head;
        head = head->next;
        delete temp;
        return;
    }
    Mahasiswa140 *current = head;
    for (int i = 1; i < posisi - 1 && current != NULL; i++)
    {
        current = current->next;
    }
    if (current != NULL && current->next != NULL)
    {
        Mahasiswa140 *temp = current->next;
        current->next = temp->next;
    }
}

```

```

        delete temp;
    }
    else
    {
        cout << "Posisi tidak valid." << endl;
    }
}

void tampilkan()
{
    Mahasiswa140 *current = head;
    cout <<
    "===== " <<
    endl;
    cout << setw(5) << left << "NO." << setw(20) << left << "NAMA"
    << "NIM" << endl;
    int i = 1;
    while (current != NULL)
    {
        cout << setw(5) << left << i << setw(20) << left <<
        current->Nama140 << current->NIM140 << endl;
        current = current->next;
        i++;
    }
    cout <<
    "===== " <<
    endl;
}

void ubah_depan(string nama_baru, string nim_baru)
{
    if (head == NULL)
    {
        cout << "Linked list kosong." << endl;
        return;
    }
    head->Nama140 = nama_baru;
    head->NIM140 = nim_baru;
    cout << "Data " << head->Nama140 << " telah diganti dengan data
    " << nama_baru << endl;
}

```

```

void ubah_belakang(string nama_baru, string nim_baru)
{
    if (head == NULL)
    {
        cout << "Linked list kosong." << endl;
        return;
    }
    Mahasiswa140 *current = head;
    Mahasiswa140 *previous = NULL;
    while (current->next != NULL)
    {
        previous = current;
        current = current->next;
    }
    string nama_lama = current->Nama140;
    current->Nama140 = nama_baru;
    current->NIM140 = nim_baru;
    cout << "Data " << nama_lama << " telah diganti dengan data "
<< nama_baru << endl;
}

void ubah_tengah(int posisi, string nama_baru, string nim_baru)
{
    if (posisi <= 1)
    {
        ubah_depan(nama_baru, nim_baru);
    }
    else
    {
        Mahasiswa140 *current = head;
        for (int i = 1; i < posisi && current != NULL; i++)
        {
            current = current->next;
        }
        if (current != NULL)
        {
            string nama_lama = current->Nama140;
            current->Nama140 = nama_baru;
            current->NIM140 = nim_baru;
        }
    }
}

```



```

        cout << "Data " << nama_lama << " telah diganti dengan
data " << nama_baru << endl;
    }
    else
    {
        cout << "Posisi tidak valid." << endl;
    }
}
}

void hapus_list()
{
    while (head != NULL)
    {
        hapus_depan();
    }
    cout << "Seluruh data mahasiswa telah dihapus." << endl;
}

void hapus_depan()
{
    if (head != NULL)
    {
        Mahasiswa140 *temp = head;
        head = head->next;
        delete temp;
    }
}

};

int main()
{
    LinkedList linked_list;
    int pilihan;
    string nama, nim;
    int posisi;
    do
    {
        cout <<
"===== " <<
endl;

```

```

        cout << " PROGRAM SINGLE LINKED LIST " << endl;
        cout <<
"===== " <<
endl;
        cout << setw(2) << "1. " << setw(17) << left << "Tambah Depan "
<< endl;
        cout << setw(2) << "2. " << setw(17) << left << "Tambah
Belakang " << endl;
        cout << setw(2) << "3. " << setw(17) << left << "Tambah Tengah
" << endl;
        cout << setw(2) << "4. " << setw(17) << left << "Ubah Depan "
<< endl;
        cout << setw(2) << "5. " << setw(17) << left << "Ubah Belakang
" << endl;
        cout << setw(2) << "6. " << setw(17) << left << "Ubah Tengah "
<< endl;
        cout << setw(2) << "7. " << setw(17) << left << "Hapus Depan "
<< endl;
        cout << setw(2) << "8. " << setw(17) << left << "Hapus Belakang
" << endl;
        cout << setw(2) << "9. " << setw(17) << left << "Hapus Tengah "
<< endl;
        cout << setw(2) << "10." << setw(17) << left << "Hapus List "
<< endl;
        cout << setw(2) << "11." << setw(17) << left << "Tampilkan" <<
endl;
        cout << setw(2) << "0. " << setw(17) << left << "Keluar" <<
endl;

        cout << "Pilih Operasi: ";
        cin >> pilihan;
        cout <<
"===== " <<
endl;

        switch (pilihan)
        {
        case 1:
            cout << "Tambah Depan" << endl;
            cout << "Masukkan Nama: ";
            cin >> nama;
            cout << "Masukkan NIM: ";
            cin >> nim;

```

```

        linked_list.tambah_depan(nama, nim);
        cout << "Data telah ditambahkan" << endl;
        break;
    case 2:
        cout << "Tambah Belakang" << endl;
        cout << "Masukkan Nama: ";
        cin >> nama;
        cout << "Masukkan NIM: ";
        cin >> nim;
        linked_list.tambah_belakang(nama, nim);
        cout << "Data telah ditambahkan" << endl;
        break;
    case 3:
        cout << "Tambah Tengah" << endl;
        cout << "Masukkan Nama: ";
        cin >> nama;
        cout << "Masukkan NIM: ";
        cin >> nim;
        cout << "Masukkan Posisi: ";
        cin >> posisi;
        linked_list.tambah_tengah(posisi, nama, nim);
        cout << "Data telah ditambahkan" << endl;
        break;
    case 4:
        cout << "Ubah Depan" << endl;
        cout << "Masukkan Nama Baru: ";
        cin >> nama;
        cout << "Masukkan NIM Baru: ";
        cin >> nim;
        linked_list.ubah_depan(nama, nim);
        cout << "Data telah diubah" << endl;
        break;
    case 5:
        cout << "Ubah Belakang" << endl;
        cout << "Masukkan Nama Baru: ";
        cin >> nama;
        cout << "Masukkan NIM Baru: ";
        cin >> nim;
        linked_list.ubah_belakang(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;
        linked_list.ubah_tengah(posisi, nama, nim);
        break;
    case 7:
        cout << "Hapus Depan" << endl;
        linked_list.hapus_depan();
        cout << "Data depan berhasil dihapus." << endl;
        break;
    case 8:
        cout << "Hapus Belakang" << endl;
        linked_list.hapus_belakang();
        cout << "Data belakang berhasil dihapus." << endl;
        break;
    case 9:
        cout << "Hapus Tengah" << endl;
        cout << "Masukkan Posisi: ";
        cin >> posisi;
        linked_list.hapus_tengah(posisi);
        cout << "Data pada posisi " << posisi << " berhasil
dihapus." << endl;
        break;
    case 10:
        cout << "Hapus List" << endl;
        linked_list.hapus_list();
        break;
    case 11:
        cout << "Tampilkan" << endl;
        linked_list.tampilkan();
        break;
    case 0:
        cout << "Keluar" << endl;
        break;
    default:
        cout << "Pilihan tidak valid, silakan coba lagi." << endl;
}

```

```

    } while (pilihan != 0);
    cout <<
    "===== " <<
endl;
    cout << " " << endl;
    cout <<
    "===== " <<
endl;
    return 0;
}

```

## Screenshot program

### Tampilkan menu ;

```

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
0. Keluar

```

### Tambah Depan

```

Pilih Operasi: 1
=====
Tambah Depan
Masukkan Nama: Jawad
Masukkan NIM: 23300001
Data telah ditambahkan

```

### Tambah Tengah

```

Tambah Tengah
Masukkan Nama: Tegar
Masukkan NIM: 2311102027
Masukkan Posisi: 2
Data telah ditambahkan

```

### Tambah belakang

```
Tambah Belakang
Masukkan Nama: Budi
Masukkan NIM: 23300099
Data telah ditambahkan
```

### Ubah belakang

```
Ubah Belakang
Masukkan Nama Baru: karyo
Masukkan NIM Baru: 231111113
Data Budi telah diganti dengan data karyo
```

### Ubah depan

```
Ubah Depan
Masukkan Nama Baru: yudho
Masukkan NIM Baru: 1222233
Data yudho telah diganti dengan data yudho
Data telah diubah
```

### Ubah tengah

```
Ubah Depan
Masukkan Nama Baru: yudho
Masukkan NIM Baru: 1222233
Data yudho telah diganti dengan data yudho
Data telah diubah
```

### Hapus depan

```
Pilih Operasi: 7
=====
=====
Hapus Depan
Data depan berhasil dihapus.
```

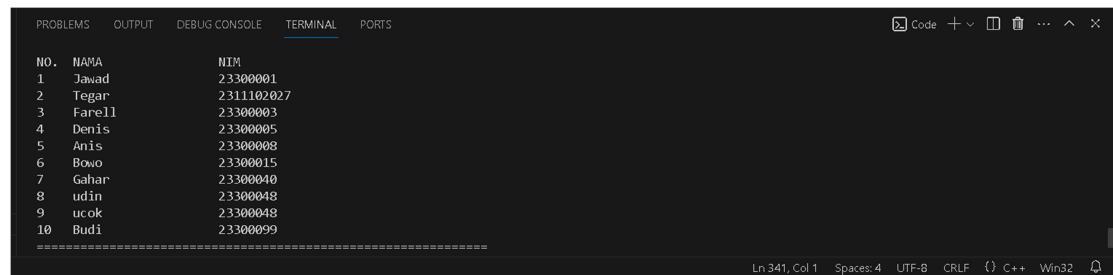
### Hapus belakang

```
Pilih Operasi: 7
=====
=====
Hapus Depan
Data depan berhasil dihapus.
```

## Hapus tengah

```
Pilih Operasi: 7
=====
=====
Hapus Depan
Data depan berhasil dihapus.
```

## Tampilan



PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL	PORTS
NO.	NAMA	NIM		
1	Jawad	23300001		
2	Tegar	2311102027		
3	Farell	23300003		
4	Denis	23300005		
5	Anis	23300008		
6	Bowo	23300015		
7	Gahar	23300040		
8	udin	23300048		
9	ucok	23300048		
10	Budi	23300099		

## Deskripsi program

Program ini memiliki fitur utama sebagai berikut:

Program yang telah dibuat pada Unguided1 adalah sebuah aplikasi yang menggunakan Linked List Non-Circular untuk menyimpan informasi Nama dan NIM mahasiswa. Aplikasi ini memberikan pengguna kemampuan untuk melakukan berbagai operasi dasar seperti penambahan data (di depan, di belakang, atau di tengah), penghapusan data (di depan, di belakang, atau di tengah), pengubahan data, dan menampilkan isi linked list melalui menu yang disediakan. Setiap operasi yang dipilih akan memanggil metode yang sesuai pada kelas LinkedList. Dalam program tersebut, setiap node dalam linked list menyimpan informasi mengenai nama dan NIM mahasiswa serta pointer yang menunjuk ke node berikutnya. Setiap opsi dalam menu memiliki fungsi spesifik dalam mengelola linked list, seperti penambahan atau penghapusan di berbagai posisi, pengubahan data, dan menampilkan atau

menghapus seluruh data dalam linked list. Pengguna dapat memanipulasi data dalam linked list sesuai dengan kebutuhan mereka melalui aplikasi ini.

## 2. Unguided 2

### Source code

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

struct Mahasiswa140
{
    string Nama140;
    string NIM140;
    Mahasiswa140 *next;
};

class LinkedList
```



```

{
private:
    Mahasiswa140 *head;

public:
    LinkedList()
    {
        head = NULL;
    }

    void tambah_depan(string nama, string nim)
    {
        Mahasiswa140 *new_mahasiswa = new Mahasiswa140;
        new_mahasiswa->Nama140 = nama;
        new_mahasiswa->NIM140 = nim;
        new_mahasiswa->next = head;
        head = new_mahasiswa;
    }

    void tambah_belakang(string nama, string nim)
    {
        Mahasiswa140 *new_mahasiswa = new Mahasiswa140;
        new_mahasiswa->Nama140 = nama;
        new_mahasiswa->NIM140 = nim;
        new_mahasiswa->next = NULL;
        if (head == NULL)
        {
            head = new_mahasiswa;
            return;
        }
        Mahasiswa140 *current = head;
        while (current->next != NULL)
        {
            current = current->next;
        }
        current->next = new_mahasiswa;
    }

    void tambah_tengah(int posisi, string nama, string nim)
    {
        if (posisi <= 1)

```

```

    {
        tambah_depan(nama, nim);
        return;
    }
    Mahasiswa140 *new_mahasiswa = new Mahasiswa140;
    new_mahasiswa->>Nama140 = nama;
    new_mahasiswa->NIM140 = nim;
    Mahasiswa140 *current = head;
    for (int i = 1; i < posisi - 1 && current != NULL; i++)
    {
        current = current->next;
    }
    if (current != NULL)
    {
        new_mahasiswa->next = current->next;
        current->next = new_mahasiswa;
    }
    else
    {
        cout << "Posisi tidak valid." << endl;
    }
}

void hapus_belakang()
{
    if (head == NULL)
    {
        cout << "Linked list kosong." << endl;
        return;
    }
    if (head->next == NULL)
    {
        delete head;
        head = NULL;
        return;
    }
    Mahasiswa140 *current = head;
    while (current->next->next != NULL)
    {
        current = current->next;
    }
}

```

```

        delete current->next;
        current->next = NULL;
    }

void hapus_tengah(int posisi)
{
    if (posisi <= 1)
    {
        Mahasiswa140 *temp = head;
        head = head->next;
        delete temp;
        return;
    }
    Mahasiswa140 *current = head;
    for (int i = 1; i < posisi - 1 && current != NULL; i++)
    {
        current = current->next;
    }
    if (current != NULL && current->next != NULL)
    {
        Mahasiswa140 *temp = current->next;
        current->next = temp->next;
        delete temp;
    }
    else
    {
        cout << "Posisi tidak valid." << endl;
    }
}

void tampilkan()
{
    Mahasiswa140 *current = head;
    cout <<
    "===== " <<
    endl;
    cout << setw(5) << left << "NO." << setw(20) << left <<
    "NAMA" << "NIM" << endl;
    int i = 1;
    while (current != NULL)
    {

```

```

        cout << setw(5) << left << i << setw(20) << left <<
current->Nama140 << current->NIM140 << endl;
        current = current->next;
        i++;
    }
    cout <<
"===== " <<
endl;
}

void ubah_depan(string nama_baru, string nim_baru)
{
    if (head == NULL)
    {
        cout << "Linked list kosong." << endl;
        return;
    }
    head->Nama140 = nama_baru;
    head->NIM140 = nim_baru;
    cout << "Data " << head->Nama140 << " telah diganti dengan
data " << nama_baru << endl;
}

void ubah_belakang(string nama_baru, string nim_baru)
{
    if (head == NULL)
    {
        cout << "Linked list kosong." << endl;
        return;
    }
    Mahasiswa140 *current = head;
    Mahasiswa140 *previous = NULL;
    while (current->next != NULL)
    {
        previous = current;
        current = current->next;
    }
    string nama_lama = current->Nama140;
    current->Nama140 = nama_baru;
    current->NIM140 = nim_baru;
    cout << "Data " << nama_lama << " telah diganti dengan data

```

```

" << nama_baru << endl;
}

void ubah_tengah(int posisi, string nama_baru, string nim_baru)
{
    if (posisi <= 1)
    {
        ubah_depan(nama_baru, nim_baru);
    }
    else
    {
        Mahasiswa140 *current = head;
        for (int i = 1; i < posisi && current != NULL; i++)
        {
            current = current->next;
        }
        if (current != NULL)
        {
            string nama_lama = current->Nama140;
            current->Nama140 = nama_baru;
            current->NIM140 = nim_baru;
            cout << "Data " << nama_lama << " telah diganti
dengan data " << nama_baru << endl;
        }
        else
        {
            cout << "Posisi tidak valid." << endl;
        }
    }
}

void hapus_list()
{
    while (head != NULL)
    {
        hapus_depan();
    }
    cout << "Seluruh data mahasiswa telah dihapus." << endl;
}

void hapus_depan()

```

```

    {
        if (head != NULL)
        {
            Mahasiswa140 *temp = head;
            head = head->next;
            delete temp;
        }
    }
};

int main()
{
    LinkedList linked_list;
    int pilihan;
    string nama, nim;
    int posisi;
    do
    {
        cout <<
"===== " <<
endl;
        cout << " PROGRAM SINGLE LINKED LIST " << endl;
        cout <<
"===== " <<
endl;
        cout << setw(2) << "1. " << setw(17) << left << "Tambah
Depan " << endl;
        cout << setw(2) << "2. " << setw(17) << left << "Tambah
Belakang " << endl;
        cout << setw(2) << "3. " << setw(17) << left << "Tambah
Tengah " << endl;
        cout << setw(2) << "4. " << setw(17) << left << "Ubah Depan
" << endl;
        cout << setw(2) << "5. " << setw(17) << left << "Ubah
Belakang " << endl;
        cout << setw(2) << "6. " << setw(17) << left << "Ubah
Tengah " << endl;
        cout << setw(2) << "7. " << setw(17) << left << "Hapus
Depan " << endl;
        cout << setw(2) << "8. " << setw(17) << left << "Hapus
Belakang " << endl;

```

```

        cout << setw(2) << "9. " << setw(17) << left << "Hapus
Tengah " << endl;
        cout << setw(2) << "10." << setw(17) << left << "Hapus List
" << endl;
        cout << setw(2) << "11." << setw(17) << left << "Tampilkan"
<< endl;
        cout << setw(2) << "0. " << setw(17) << left << "Keluar" <<
endl;

        cout << "Pilih Operasi: ";
        cin >> pilihan;
        cout <<
"===== " <<
endl;

        switch (pilihan)
        {
        case 1:
            cout << "Tambah Depan" << endl;
            cout << "Masukkan Nama: ";
            cin >> nama;
            cout << "Masukkan NIM: ";
            cin >> nim;
            linked_list.tambah_depan(nama, nim);
            cout << "Data telah ditambahkan" << endl;
            break;
        case 2:
            cout << "Tambah Belakang" << endl;
            cout << "Masukkan Nama: ";
            cin >> nama;
            cout << "Masukkan NIM: ";
            cin >> nim;
            linked_list.tambah_belakang(nama, nim);
            cout << "Data telah ditambahkan" << endl;
            break;
        case 3:
            cout << "Tambah Tengah" << endl;
            cout << "Masukkan Nama: ";
            cin >> nama;
            cout << "Masukkan NIM: ";
            cin >> nim;
            cout << "Masukkan Posisi: ";
            cin >> posisi;

```

```

        linked_list.tambah_tengah(posisi, nama, nim);
        cout << "Data telah ditambahkan" << endl;
        break;
    case 4:
        cout << "Ubah Depan" << endl;
        cout << "Masukkan Nama Baru: ";
        cin >> nama;
        cout << "Masukkan NIM Baru: ";
        cin >> nim;
        linked_list.ubah_depan(nama, nim);
        cout << "Data telah diubah" << endl;
        break;
    case 5:
        cout << "Ubah Belakang" << endl;
        cout << "Masukkan Nama Baru: ";
        cin >> nama;
        cout << "Masukkan NIM Baru: ";
        cin >> nim;
        linked_list.ubah_belakang(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;
        linked_list.ubah_tengah(posisi, nama, nim);
        break;
    case 7:
        cout << "Hapus Depan" << endl;
        linked_list.hapus_depan();
        cout << "Data depan berhasil dihapus." << endl;
        break;
    case 8:
        cout << "Hapus Belakang" << endl;
        linked_list.hapus_belakang();
        cout << "Data belakang berhasil dihapus." << endl;
        break;
    case 9:

```



```

        cout << "Hapus Tengah" << endl;
        cout << "Masukkan Posisi: ";
        cin >> posisi;
        linked_list.hapus_tengah(posisi);
        cout << "Data pada posisi " << posisi << " berhasil
dihapus." << endl;
        break;
    case 10:
        cout << "Hapus List" << endl;
        linked_list.hapus_list();
        break;
    case 11:
        cout << "Tampilkan" << endl;
        linked_list.tampilkan();
        break;
    case 0:
        cout << "Keluar" << endl;
        break;
    default:
        cout << "Pilihan tidak valid, silakan coba lagi." <<
endl;
    }
} while (pilihan != 0);
cout <<
"===== " <<
endl;
    cout << "Tegar Bangkit Wijaya" << endl;
    cout <<
"===== " <<
endl;
    return 0;
}

```

**Screenshoot program**

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
NO. NAME NIM
1 Jawad 23300001
2 Tegar 2311102027
3 Farell 23300003
4 Denis 23300005
5 Anis 23300008
6 Bowo 23300015
7 Gahar 23300040
8 udin 23300048
9 ucok 23300048
10 Budi 23300099
=====
Ln 341, Col 1 Spaces: 4 UTF-8 CRLF { } C++ Win32
```

## Deskripsi program

Buat menu terlebih dahulu lalu masukkan data sesuai soal yang di minta. Pertama, data dengan nama Jawad dan NIM 23300001 dimasukan menggunakan operasi insert depan kemudian data lain nya di masukan dengan insert tengah dan masukan posisi lalu data terakhir Budi masukan dengan insert belakang Setelah semua data dimasukkan tampilkan data yg telah di masukan

## 3. Unguided 3

### Source code

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

struct Mahasiswa140
{
    string Nama140;
    string NIM140;
    Mahasiswa140 *next;
};

class LinkedList
{
private:
    Mahasiswa140 *head;

public:
    LinkedList()
    {
```

```

        head = NULL;
    }

    void tambah_depan(string nama, string nim)
    {
        Mahasiswa140 *new_mahasiswa = new Mahasiswa140;
        new_mahasiswa->Nama140 = nama;
        new_mahasiswa->NIM140 = nim;
        new_mahasiswa->next = head;
        head = new_mahasiswa;
    }

    void tambah_belakang(string nama, string nim)
    {
        Mahasiswa140 *new_mahasiswa = new Mahasiswa140;
        new_mahasiswa->Nama140 = nama;
        new_mahasiswa->NIM140 = nim;
        new_mahasiswa->next = NULL;
        if (head == NULL)
        {
            head = new_mahasiswa;
            return;
        }
        Mahasiswa140 *current = head;
        while (current->next != NULL)
        {
            current = current->next;
        }
        current->next = new_mahasiswa;
    }

    void tambah_tengah(int posisi, string nama, string nim)
    {
        if (posisi <= 1)
        {
            tambah_depan(nama, nim);
            return;
        }
        Mahasiswa140 *new_mahasiswa = new Mahasiswa140;
        new_mahasiswa->Nama140 = nama;
        new_mahasiswa->NIM140 = nim;
    }

```

```

Mahasiswa140 *current = head;
for (int i = 1; i < posisi - 1 && current != NULL; i++)
{
    current = current->next;
}
if (current != NULL)
{
    new_mahasiswa->next = current->next;
    current->next = new_mahasiswa;
}
else
{
    cout << "Posisi tidak valid." << endl;
}
}

void hapus_belakang()
{
    if (head == NULL)
    {
        cout << "Linked list kosong." << endl;
        return;
    }
    if (head->next == NULL)
    {
        delete head;
        head = NULL;
        return;
    }
    Mahasiswa140 *current = head;
    while (current->next->next != NULL)
    {
        current = current->next;
    }
    delete current->next;
    current->next = NULL;
}

void hapus_tengah(int posisi)
{
    if (posisi <= 1)

```

```

    {
        Mahasiswa140 *temp = head;
        head = head->next;
        delete temp;
        return;
    }
    Mahasiswa140 *current = head;
    for (int i = 1; i < posisi - 1 && current != NULL; i++)
    {
        current = current->next;
    }
    if (current != NULL && current->next != NULL)
    {
        Mahasiswa140 *temp = current->next;
        current->next = temp->next;
        delete temp;
    }
    else
    {
        cout << "Posisi tidak valid." << endl;
    }
}

void tampilkan()
{
    Mahasiswa140 *current = head;
    cout <<
    "===== " <<
    endl;
    cout << setw(5) << left << "NO." << setw(20) << left << "NAMA"
    << "NIM" << endl;
    int i = 1;
    while (current != NULL)
    {
        cout << setw(5) << left << i << setw(20) << left <<
        current->Nama140 << current->NIM140 << endl;
        current = current->next;
        i++;
    }
}

```

```

        cout <<
        "===== " <<
endl;
    }

    void ubah_depan(string nama_baru, string nim_baru)
    {
        if (head == NULL)
        {
            cout << "Linked list kosong." << endl;
            return;
        }
        head->Nama140 = nama_baru;
        head->NIM140 = nim_baru;
        cout << "Data " << head->Nama140 << " telah diganti dengan data "
        << nama_baru << endl;
    }

    void ubah_belakang(string nama_baru, string nim_baru)
    {
        if (head == NULL)
        {
            cout << "Linked list kosong." << endl;
            return;
        }
        Mahasiswa140 *current = head;
        Mahasiswa140 *previous = NULL;
        while (current->next != NULL)
        {
            previous = current;
            current = current->next;
        }
        string nama_lama = current->Nama140;
        current->Nama140 = nama_baru;
        current->NIM140 = nim_baru;
        cout << "Data " << nama_lama << " telah diganti dengan data "
        << nama_baru << endl;
    }

    void ubah_tengah(int posisi, string nama_baru, string nim_baru)
    {

```

```

        if (posisi <= 1)
        {
            ubah_depan(nama_baru, nim_baru);
        }
        else
        {
            Mahasiswa140 *current = head;
            for (int i = 1; i < posisi && current != NULL; i++)
            {
                current = current->next;
            }
            if (current != NULL)
            {
                string nama_lama = current->Nama140;
                current->Nama140 = nama_baru;
                current->NIM140 = nim_baru;
                cout << "Data " << nama_lama << " telah diganti dengan
data " << nama_baru << endl;
            }
            else
            {
                cout << "Posisi tidak valid." << endl;
            }
        }
    }

void hapus_list()
{
    while (head != NULL)
    {
        hapus_depan();
    }
    cout << "Seluruh data mahasiswa telah dihapus." << endl;
}

void hapus_depan()
{
    if (head != NULL)
    {
        Mahasiswa140 *temp = head;
        head = head->next;
    }
}

```

```

        delete temp;
    }
}
};

int main()
{
    LinkedList linked_list;
    int pilihan;
    string nama, nim;
    int posisi;
    do
    {
        cout <<
"===== " <<
endl;
        cout << " PROGRAM SINGLE LINKED LIST " << endl;
        cout <<
"===== " <<
endl;
        cout << setw(2) << "1. " << setw(17) << left << "Tambah Depan "
<< endl;
        cout << setw(2) << "2. " << setw(17) << left << "Tambah
Belakang " << endl;
        cout << setw(2) << "3. " << setw(17) << left << "Tambah Tengah
" << endl;
        cout << setw(2) << "4. " << setw(17) << left << "Ubah Depan "
<< endl;
        cout << setw(2) << "5. " << setw(17) << left << "Ubah Belakang
" << endl;
        cout << setw(2) << "6. " << setw(17) << left << "Ubah Tengah "
<< endl;
        cout << setw(2) << "7. " << setw(17) << left << "Hapus Depan "
<< endl;
        cout << setw(2) << "8. " << setw(17) << left << "Hapus Belakang
" << endl;
        cout << setw(2) << "9. " << setw(17) << left << "Hapus Tengah "
<< endl;
        cout << setw(2) << "10." << setw(17) << left << "Hapus List "
<< endl;

```



```

        cout << setw(2) << "11." << setw(17) << left << "Tampilkan" <<
endl;
        cout << setw(2) << "0. " << setw(17) << left << "Keluar" <<
endl;

        cout << "Pilih Operasi: ";
        cin >> pilihan;
        cout <<
"===== " <<
endl;

        switch (pilihan)
        {
        case 1:
            cout << "Tambah Depan" << endl;
            cout << "Masukkan Nama: ";
            cin >> nama;
            cout << "Masukkan NIM: ";
            cin >> nim;
            linked_list.tambah_depan(nama, nim);
            cout << "Data telah ditambahkan" << endl;
            break;
        case 2:
            cout << "Tambah Belakang" << endl;
            cout << "Masukkan Nama: ";
            cin >> nama;
            cout << "Masukkan NIM: ";
            cin >> nim;
            linked_list.tambah_belakang(nama, nim);
            cout << "Data telah ditambahkan" << endl;
            break;
        case 3:
            cout << "Tambah Tengah" << endl;
            cout << "Masukkan Nama: ";
            cin >> nama;
            cout << "Masukkan NIM: ";
            cin >> nim;
            cout << "Masukkan Posisi: ";
            cin >> posisi;
            linked_list.tambah_tengah(posisi, nama, nim);
            cout << "Data telah ditambahkan" << endl;
            break;
        case 4:

```

```

        cout << "Ubah Depan" << endl;
        cout << "Masukkan Nama Baru: ";
        cin >> nama;
        cout << "Masukkan NIM Baru: ";
        cin >> nim;
        linked_list.ubah_depan(nama, nim);
        cout << "Data telah diubah" << endl;
        break;
    case 5:
        cout << "Ubah Belakang" << endl;
        cout << "Masukkan Nama Baru: ";
        cin >> nama;
        cout << "Masukkan NIM Baru: ";
        cin >> nim;
        linked_list.ubah_belakang(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;
        linked_list.ubah_tengah(posisi, nama, nim);
        break;
    case 7:
        cout << "Hapus Depan" << endl;
        linked_list.hapus_depan();
        cout << "Data depan berhasil dihapus." << endl;
        break;
    case 8:
        cout << "Hapus Belakang" << endl;
        linked_list.hapus_belakang();
        cout << "Data belakang berhasil dihapus." << endl;
        break;
    case 9:
        cout << "Hapus Tengah" << endl;
        cout << "Masukkan Posisi: ";
        cin >> posisi;
        linked_list.hapus_tengah(posisi);

```

```

        cout << "Data pada posisi " << posisi << " berhasil
dihapus." << endl;
        break;
    case 10:
        cout << "Hapus List" << endl;
        linked_list.hapus_list();
        break;
    case 11:
        cout << "Tampilkan" << endl;
        linked_list.tampilkan();
        break;
    case 0:
        cout << "Keluar" << endl;
        break;
    default:
        cout << "Pilihan tidak valid, silakan coba lagi." << endl;
    }
} while (pilihan != 0);
cout <<
"===== " <<
endl;
cout << "" << endl;
cout <<
"===== " <<
endl;
return 0;
}

```

**a.** Tambahkan data Wati 2330004 diantara Farell dan Dennis

NO.	NAMA	NIM
1	Jawad	23300001
2	Tegar	2311102027
3	Farrel	23300003
4	Wati	23300004
5	Denis	23300005
6	Anis	23300008
7	Bowo	23300015
8	Gahar	23300040

- b. Hapus data denis

Pilih Operasi: 9

=====

Hapus Tengah

Masukkan Posisi: 5

Data pada posisi 5 berhasil dihapus.

- c. Tambahkan data Owi 2330000 di awal

Tambah Depan

Masukkan Nama: Owi

Masukkan NIM: 2330000

Data telah ditambahkan

- d. Tambahkan data David 23300100 di akhir

Tambah Belakang

Masukkan Nama: David

Masukkan NIM: 23300100

Data telah ditambahkan

- e. Ubah data Udin menjadi Idin 23300045

1	Owi	2330000
2	Jawad	23300001
3	Tegar	2311102027
4	Farrel	23300003
5	Wati	23300004
6	Anis	23300008
7	Bowo	23300015
8	Gahar	23300040
9	Idin	23300045
10	Ucok	23300050
11	Budi	23300099

- f. Ubah data terakhir menjadi Lucy 23300101

Ubah Belakang

Masukkan Nama Baru: Lucy

Masukkan NIM Baru: 23300101

Data David telah diganti dengan data Lucy

- g. Hapus data awal

Hapus Depan

Data depan berhasil dihapus.

- h. Ubah data awal menjadi Bagus 2330002

Ubah Depan

Masukkan Nama Baru: Bagus 2330002

Masukkan NIM Baru: Data Bagus telah diganti dengan data Bagus

Data telah diubah

- i. Hapus data terakhir

```
Hapus Belakang  
Data belakang berhasil dihapus.
```

- j. Tampilkan seluruh data

	PROBLEMS	OUTPUT	DEBUG CONSOLE	TERMINAL	PORTS
1	Bagas		2330002		
2	Jawad		23300001		
3	Tegar		2311102027		
4	Farrel		23300003		
5	Wati		2330004		
6	Anis		23300008		
7	Bowo		23300015		
8	Gahar		23300040		
9	Idin		23300045		
10	Ucok		23300050		
11	Budi		23300099		
=====					

## **BAB IV**

### **KESIMPULAN**

Implementasi Linked List dalam bahasa pemrograman C++ yang terdapat dalam materi guided maupun unguided menyajikan fitur dasar seperti penambahan, penghapusan, dan pengubahan data dalam struktur data tersebut. Dilengkapi dengan menu interaktif, program-program ini memungkinkan pengguna untuk melakukan berbagai operasi terhadap Linked List, seperti menambahkan, mengubah, atau menghapus data, serta menampilkan isi dari Linked List. Dengan demikian, program-program ini dapat dijadikan sebagai contoh implementasi praktis Linked List dalam bahasa pemrograman C++ dan sebagai alat yang berguna dalam pengelolaan serta manipulasi data menggunakan struktur data Linked List.