LAPORAN PRAKTIKUM STRUKTUR DATA DAN ALGORITMA

MODUL 4
LINKED LIST CIRCULAR DAN NON CIRCULAR

Disusun Oleh :

Muhammad Hamzah Haifan Ma'ruf

231102091

Dosen :

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

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

INSTITUT TEKNOLOGI TELKOM PURWOKERTO

2024

A. **Dasar Teori**

Linked list dapat dibagi menjadi dua jenis, yaitu linked list circular dan non-circular.

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 implementasinya, linked list circular dan non-circular memiliki beberapa perbedaan. Salah satunya adalah pada 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.

B. **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
        {
```

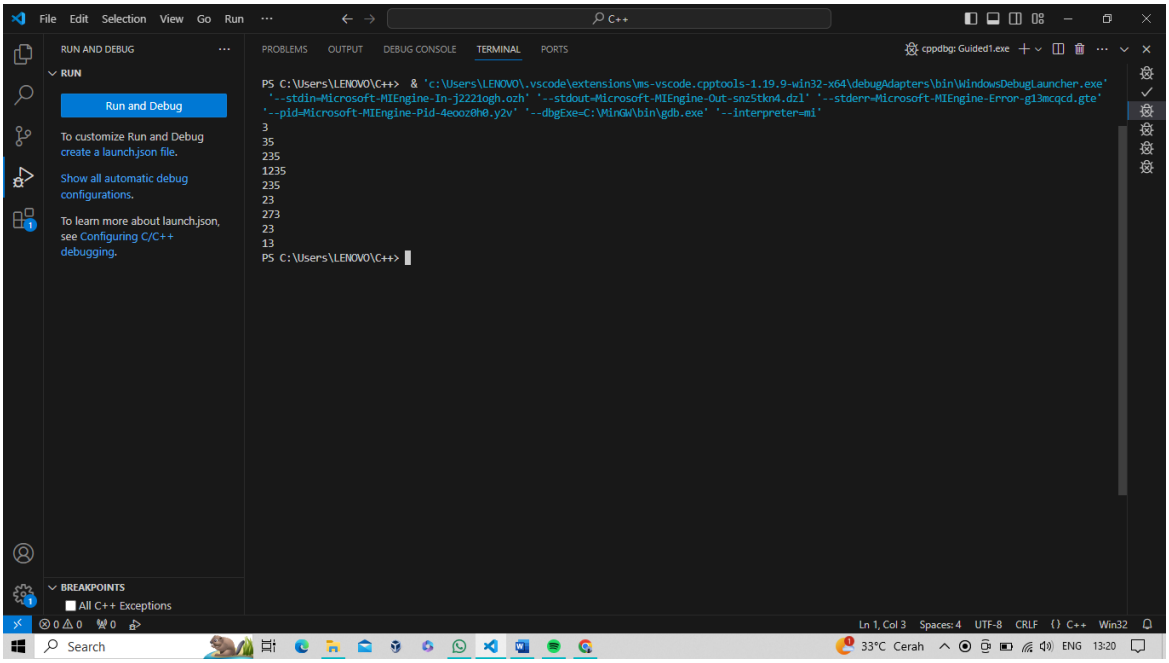
```
            head = tail = NULL;
        }
    }
    else
    {
        cout << "List kosong!" << endl;
    }
}
// Hapus Tengah
void hapusTengah(int posisi)
{
    Node *bantu, *hapus, *sebelum;
    if (posisi < 1 || posisi > hitungList())
    {
        cout << "Posisi di luar jangkauan" << endl;
    }
    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;
}
```

Screenshots Output :



Deskripsi :

Program ini menyediakan berbagai fungsi operasi seperti penambahan elemen di bagian depan, bagian belakang, atau pada posisi tertentu, penghapusan elemen dari bagian depan, belakang, atau posisi tertentu, serta pembaruan nilai elemen. Selain itu, program juga memiliki fungsi untuk menghitung jumlah elemen dalam linked list dan untuk menampilkan isi dari linked list. Dalam main() function, program menunjukkan penggunaan fungsi-fungsi tersebut dengan melakukan berbagai operasi pada linked list seperti penambahan, penghapusan, dan pembaruan elemen, serta menampilkan hasilnya. Setiap langkah operasi pada linked list diikuti dengan output yang menampilkan isi linked list setelah operasi tersebut dilakukan. Dengan demikian, program ini memberikan pemahaman praktis tentang penggunaan linked list dalam pemrograman C++ untuk mengelola dan memanipulasi data.

Guided 2
Source code

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

// 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
```

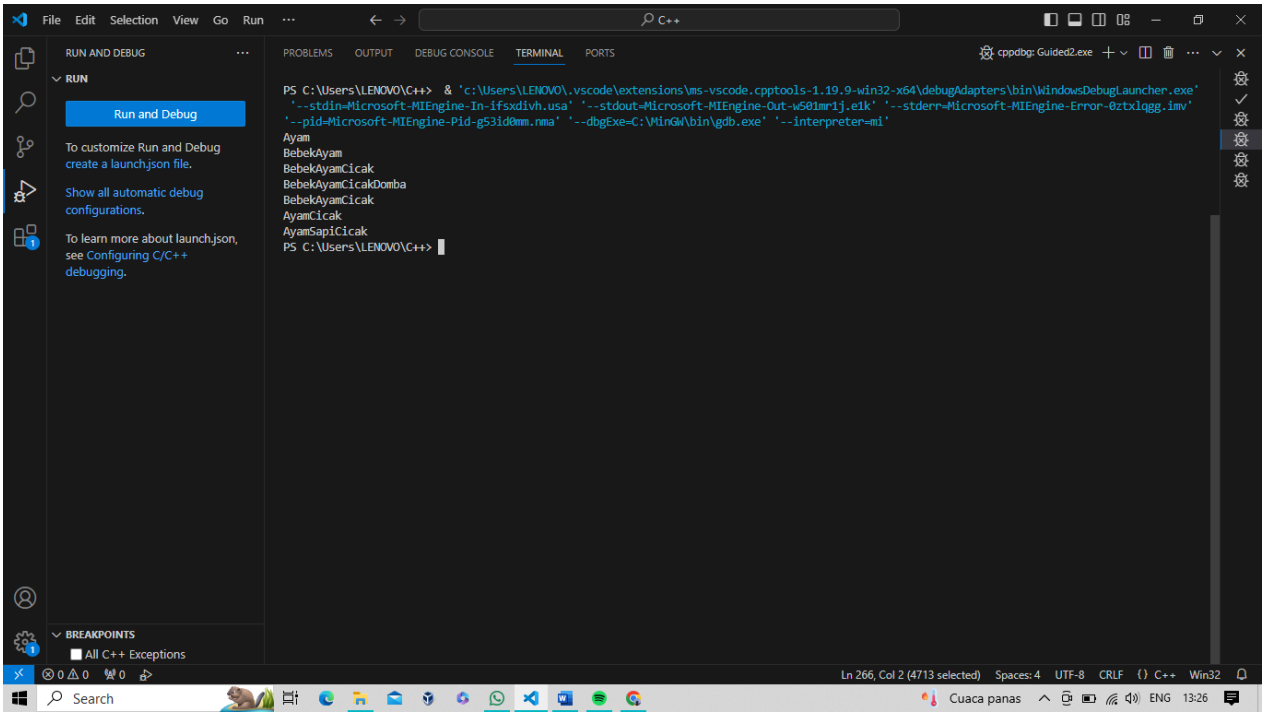
```
void insertTengah(string data, int posisi)
{
    if (isEmpty() == 1)
    {
        head = baru;
        tail = head;
        baru->next = head;
    }
    else
    {
        baru->data = data;
        // transversing
        int nomor = 1;
        bantu = head;
        while (nomor < posisi - 1)
        {
            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;
    }
```

```cpp
}

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;
}
```

Screenshots Output :



Deskripsi :

Linked list ini terdiri dari simpul-simpul yang saling terhubung, dengan setiap simpul memiliki dua bagian yaitu data dan pointer ke simpul berikutnya. Program menyediakan fungsi-fungsi untuk operasi dasar pada linked list, seperti penambahan dan penghapusan elemen di depan, belakang, atau tengah, serta untuk menampilkan isi dari linked list. Fungsi-fungsi ini diaplikasikan dalam fungsi main(), di mana setelah inisialisasi linked list, program melakukan serangkaian operasi untuk menambah, menghapus, dan menampilkan isi dari linked list. Penting untuk dicatat bahwa linked list yang digunakan bersifat circular, yang berarti elemen terakhir terhubung kembali ke elemen pertama, membentuk suatu lingkaran. Ini memengaruhi implementasi operasi penambahan dan penghapusan di bagian belakang linked list yang memperhatikan sifat lingkaran tersebut.

C. **Unguided 1**

Source code

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

struct Mahasiswa_091 {
    string Nama_091;
    string NIM_091;
    Mahasiswa_091* next;
};

class LinkedList {
private:
    Mahasiswa_091* head;

public:
    LinkedList() {
        head = NULL;
    }

    void tambah_depan(string nama, string nim) {
        Mahasiswa_091* new_mahasiswa = new Mahasiswa_091;
        new_mahasiswa->Nama_091 = nama;
        new_mahasiswa->NIM_091 = nim;
        new_mahasiswa->next = head;
        head = new_mahasiswa;
    }

    void tambah_belakang(string nama, string nim) {
        Mahasiswa_091* new_mahasiswa = new Mahasiswa_091;
        new_mahasiswa->Nama_091 = nama;
        new_mahasiswa->NIM_091 = nim;
        new_mahasiswa->next = NULL;

        if (head == NULL) {
            head = new_mahasiswa;
            return;
        }

        Mahasiswa_091* 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;
        }

        Mahasiswa_091* new_mahasiswa = new Mahasiswa_091;
        new_mahasiswa->Nama_091 = nama;
        new_mahasiswa->NIM_091 = nim;

        Mahasiswa_091* 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;
        }

        Mahasiswa_091* current = head;
        while (current->next->next != NULL) {
            current = current->next;
        }

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

    void hapus_tengah(int posisi) {
        if (posisi <= 1) {
            Mahasiswa_091* temp = head;
            head = head->next;
            delete temp;
            return;
```

```
        }

        Mahasiswa_091* current = head;
        for (int i = 1; i < posisi - 1 && current != NULL; i++) {
            current = current->next;
        }

        if (current != NULL && current->next != NULL) {
            Mahasiswa_091* temp = current->next;
            current->next = temp->next;
            delete temp;
        } else {
            cout << "Posisi tidak valid." << endl;
        }
    }

    void tampilkan() {
        Mahasiswa_091* 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->Nama_091 << current->NIM_091 << 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->Nama_091 = nama_baru;
        head->NIM_091 = nim_baru;
        cout << "Data " << head->Nama_091 << " 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;
    }

    Mahasiswa_091* current = head;
    Mahasiswa_091* previous = NULL;
    while (current->next != NULL) {
        previous = current;
        current = current->next;
    }

    string nama_lama = current->Nama_091;
    current->Nama_091 = nama_baru;
    current->NIM_091 = 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 {
        Mahasiswa_091* current = head;
        for (int i = 1; i < posisi && current != NULL; i++) {
            current = current->next;
        }
        if (current != NULL) {
            string nama_lama = current->Nama_091;
            current->Nama_091 = nama_baru;
            current->NIM_091 = 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) {
```

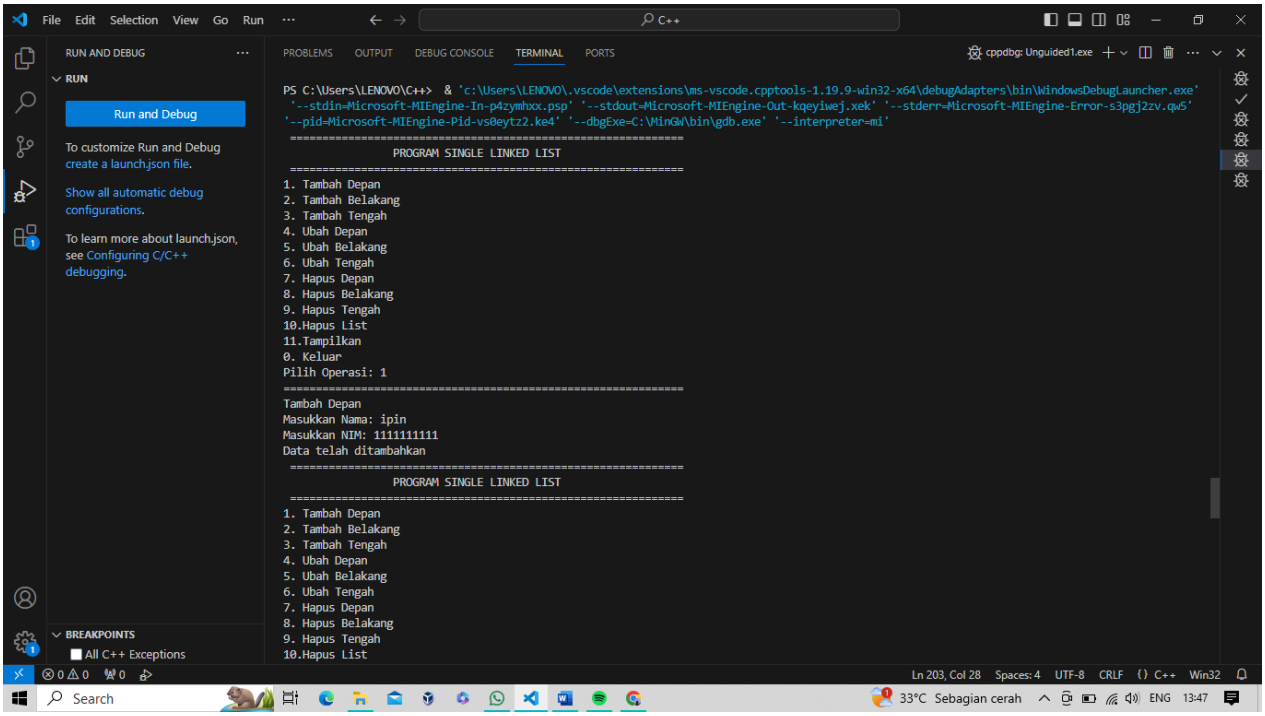
```
            Mahasiswa_091* temp = head;
            head = head->next;
            delete temp;
        }
    }
};

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;
```

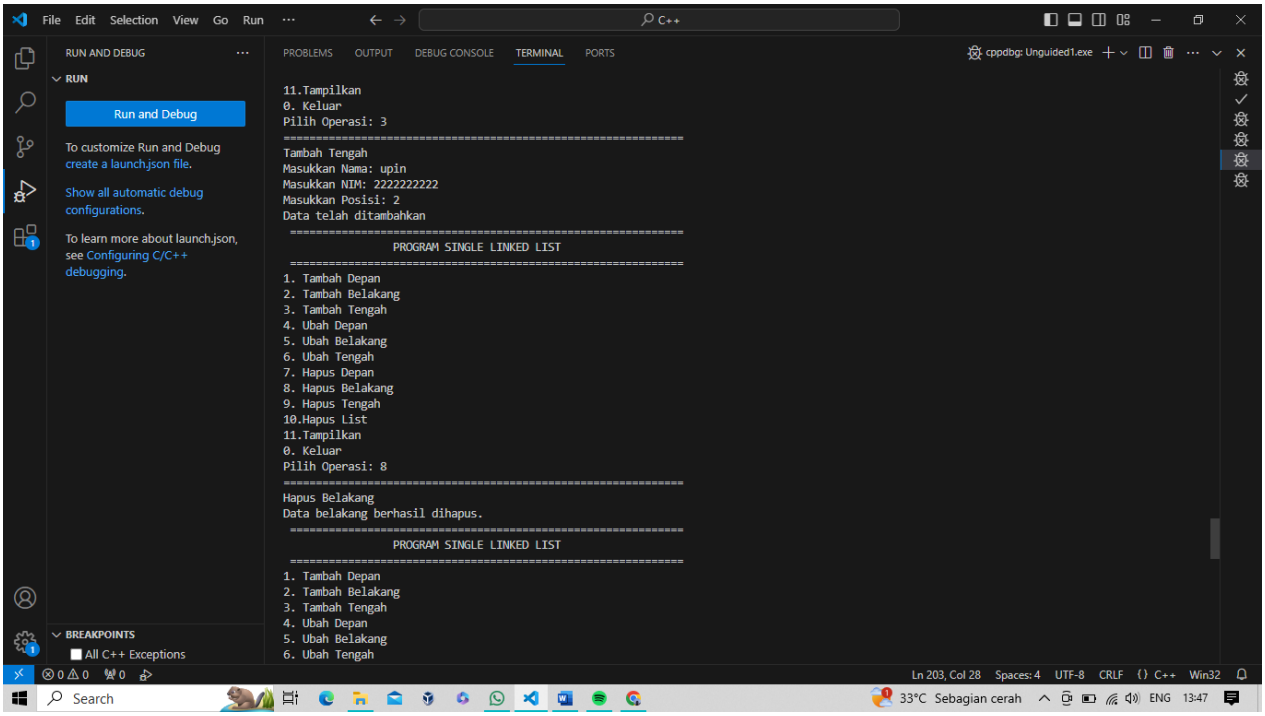
```
        cout <<
"================================================================="
<< endl;

        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;
```

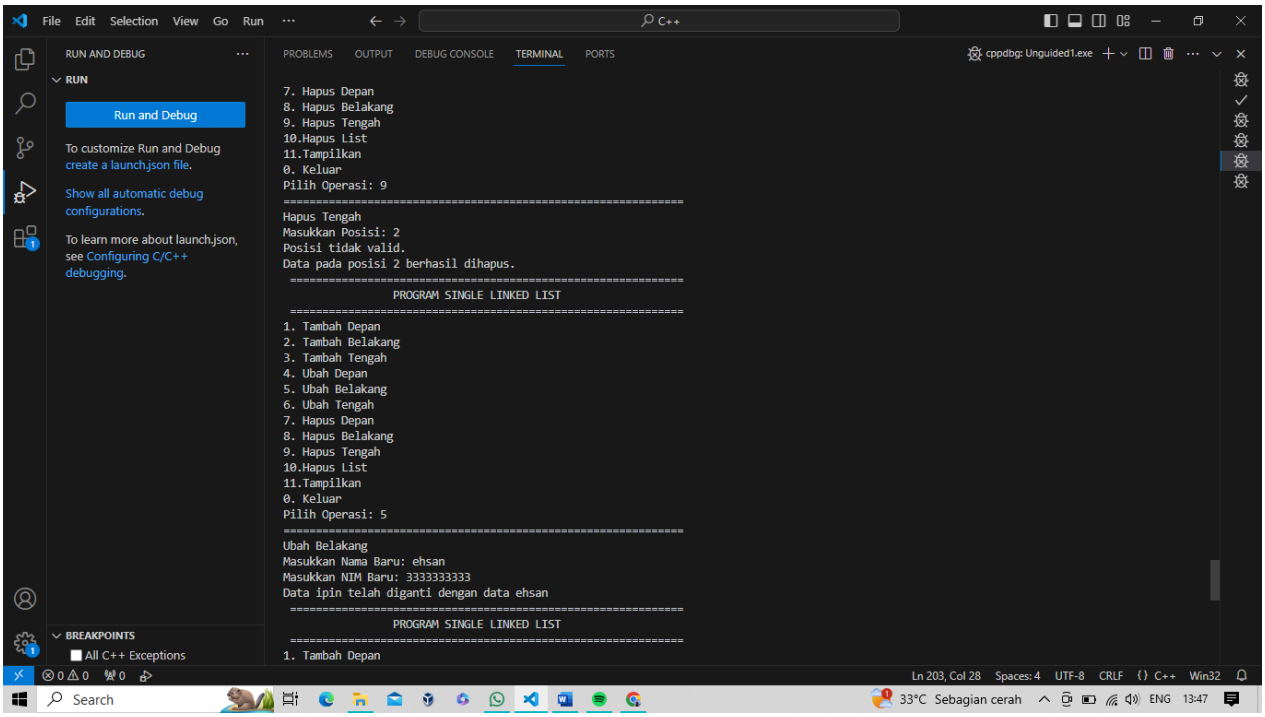
```
                linked_list.ubah_belakang(nama, nim);
                break;
            case 6:
                cout << "Ubah Tengah" << endl;
                cout << "Masukkan Nama Baru: ";
                cin >> nama;
                cout << "Masukkan NIM Baru: ";
                cin >> nim;
                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);
```

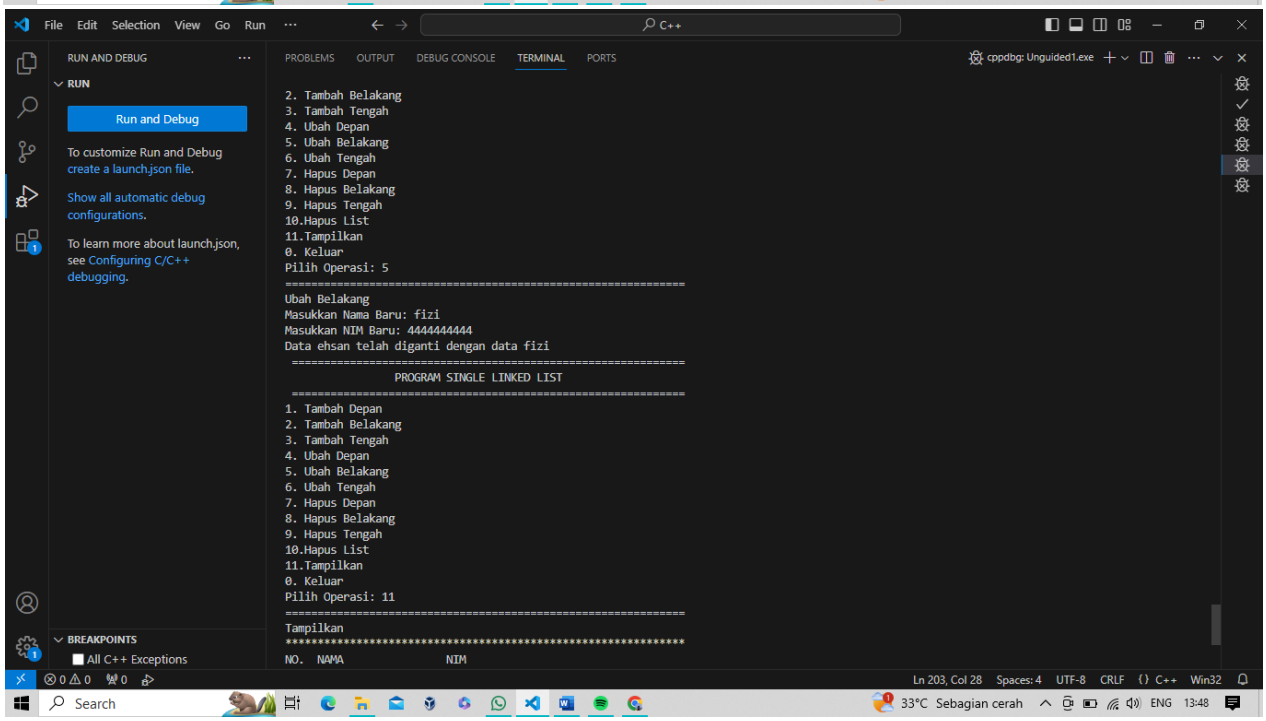
```
    cout <<
"================================================================="
<< endl;

    return 0;
}
```

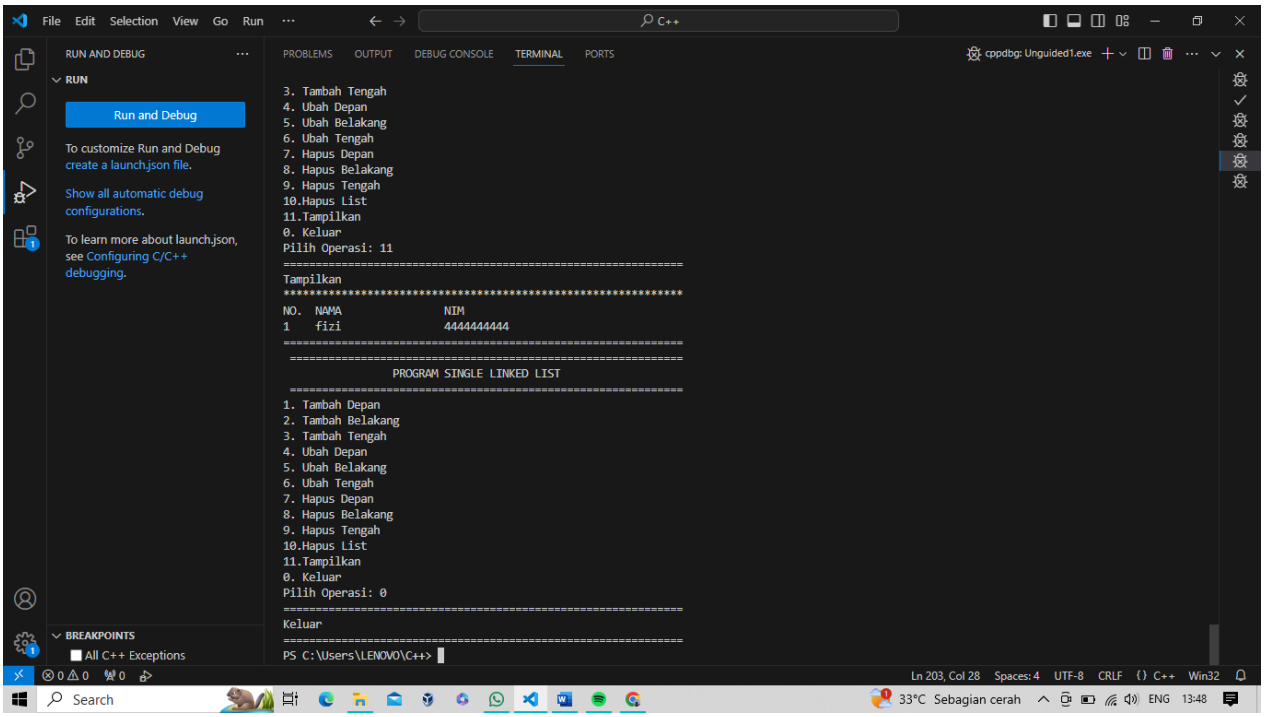
Screenshots Output :






File Edit Selection View Go Run
C++
RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug create a launch.json file.
Show all automatic debug configurations.
To learn more about launch.json, see Configuring C/C++ debugging.
BREAKPOINTS
All C++ Exceptions
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
cppdbg: Unguided1.exe
7. Hapus Depan
8. Hapus Belakang
9. Hapus Tengah
10.Hapus List
11.Tampilkan
0. Keluar
Pilih Operasi: 9
Hapus Tengah
Masukkan Posisi: 2
Posisi tidak valid.
Data pada posisi 2 berhasil dihapus.
PROGRAM SINGLE LINKED LIST
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
Pilih Operasi: 5
Ubah Belakang
Masukkan Nama Baru: ehsan
Masukkan NIM Baru: 3333333333
Data ipin telah diganti dengan data ehsan
PROGRAM SINGLE LINKED LIST
1. Tambah Depan
Ln 203, Col 28 Spaces: 4 UTF-8 CRLF C++ Win32
Search
33°C Sebagian cerah
ENG 13:47


File Edit Selection View Go Run
C++
RUN AND DEBUG
RUN
Run and Debug
To customize Run and Debug create a launch.json file.
Show all automatic debug configurations.
To learn more about launch.json, see Configuring C/C++ debugging.
BREAKPOINTS
All C++ Exceptions
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
cppdbg: Unguided1.exe
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
Pilih Operasi: 5
Ubah Belakang
Masukkan Nama Baru: fizi
Masukkan NIM Baru: 4444444444
Data ehsan telah diganti dengan data fizi
PROGRAM SINGLE LINKED LIST
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
Pilih Operasi: 11
Tampilkan
NO. NAMA NIM
Ln 203, Col 28 Spaces: 4 UTF-8 CRLF C++ Win32
Search
33°C Sebagian cerah
ENG 13:48

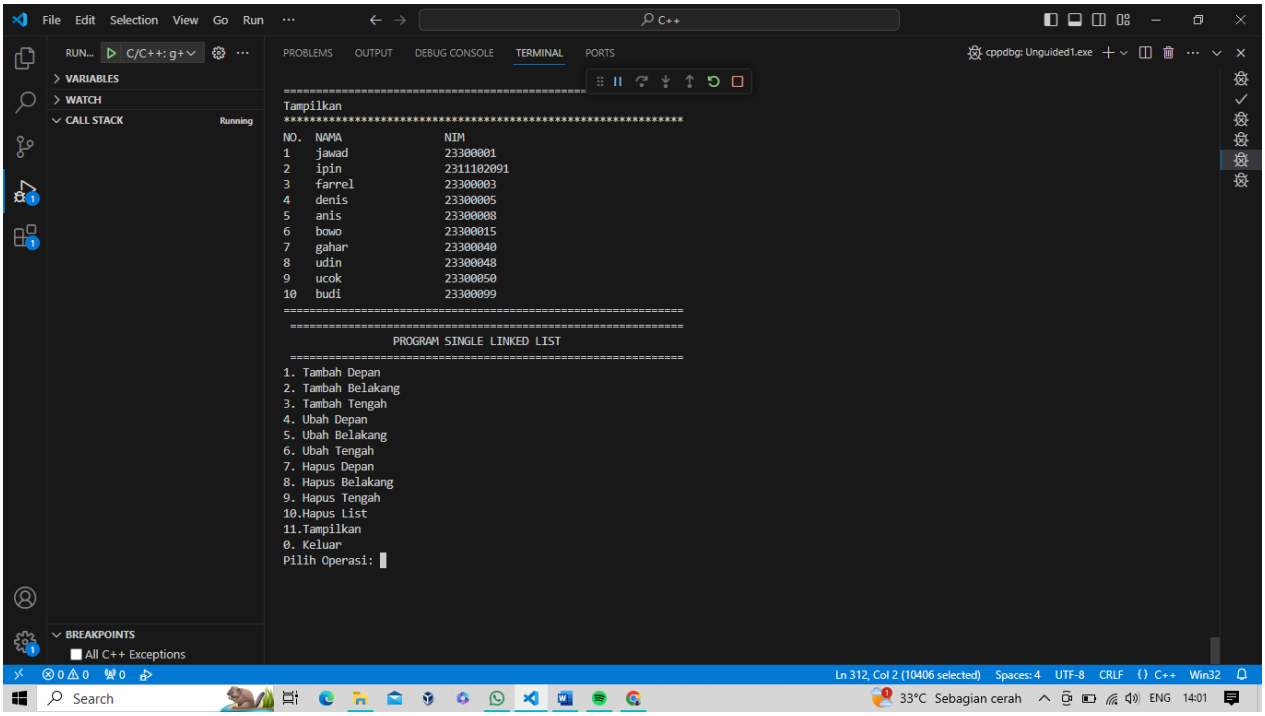


Deskripsi :

Program di atas merupakan implementasi dari linked list dalam bahasa pemrograman C++, di mana setiap elemen linked list direpresentasikan sebagai sebuah objek `Mahasiswa_091` yang memiliki dua atribut, yaitu `Nama_091` dan `NIM_091`, serta sebuah pointer `next` yang menunjukkan ke elemen berikutnya. Kelas `LinkedList` menyediakan berbagai fungsi untuk operasi dasar pada linked list, seperti penambahan elemen di depan, belakang, atau tengah, penghapusan elemen dari depan, belakang, atau tengah, serta untuk menampilkan isi dari linked list. Pada fungsi `main()`, program menyediakan menu operasi yang dapat dipilih oleh pengguna, antara lain penambahan elemen di depan, belakang, atau tengah, penghapusan elemen dari depan, belakang, atau tengah, pengubahan data pada elemen depan, belakang, atau tengah, dan menampilkan isi dari linked list. Pengguna diminta untuk memasukkan data mahasiswa seperti nama dan NIM sesuai dengan operasi yang dipilih.

Selain itu, program juga memperhatikan kasus-kasus khusus, seperti ketika linked list kosong, ketika posisi yang dimasukkan tidak valid pada operasi penambahan atau penghapusan tengah, serta memberikan pesan kesalahan jika pilihan menu tidak valid. Program terus berjalan hingga pengguna memilih untuk keluar (memasukkan pilihan 0). Setelah itu, program menampilkan pesan "Keluar" dan berakhir.

Unguided 2
Screenshots Output :



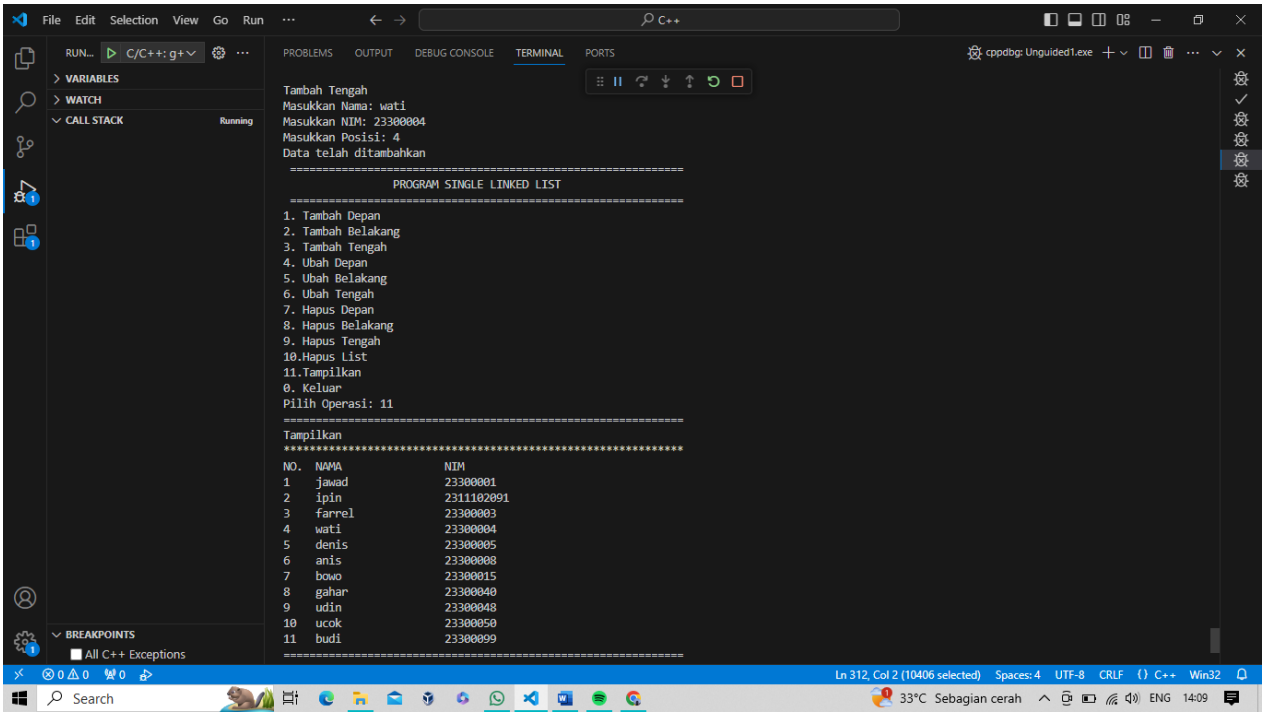
Deskripsi :

Setelah membuat menu, data dimasukkan sesuai urutan yang diminta dari soal Unguided2. Pertama, data dengan nama Jawad dan NIM 23300001 dimasukkan menggunakan operasi insert depan (menu nomor 4). Kemudian, data lainnya dimasukkan menggunakan operasi insert belakang (menu nomor 5). Setelah semua data lain telah dimasukkan, data dengan nama dan NIM yang diminta dari soal Unguided2 dimasukkan menggunakan operasi insert tengah (menu nomor 6). Setelah semua data dimasukkan, tampilkan data yang telah dimasukkan.

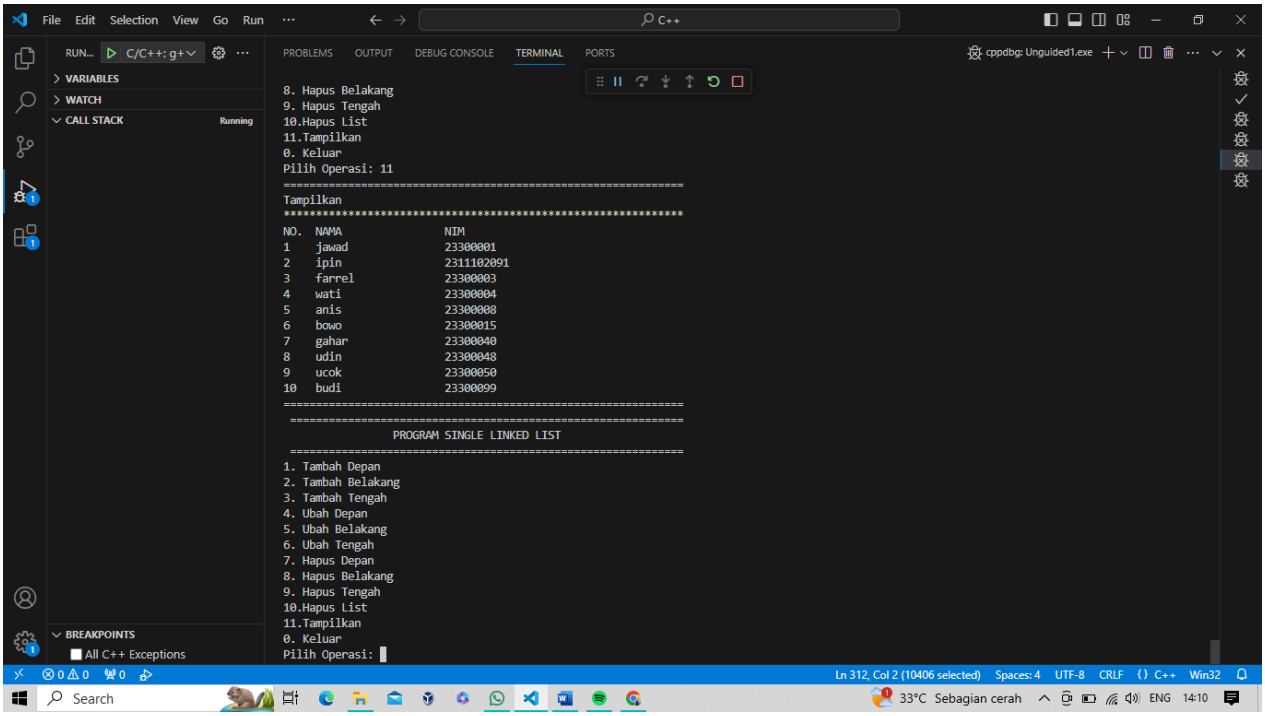
Unguided 3

a) Tambahkan data berikut diantara Farrel dan Denis:

Wati 2330004

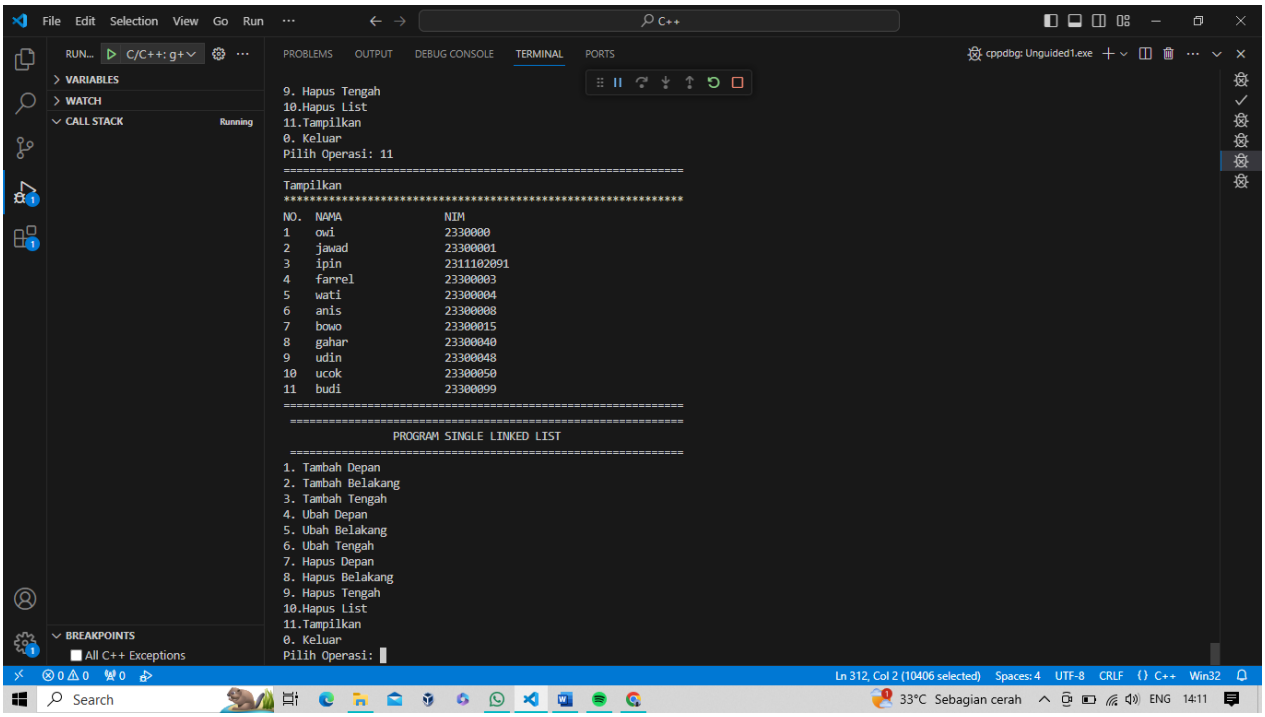


b) Hapus data Denis



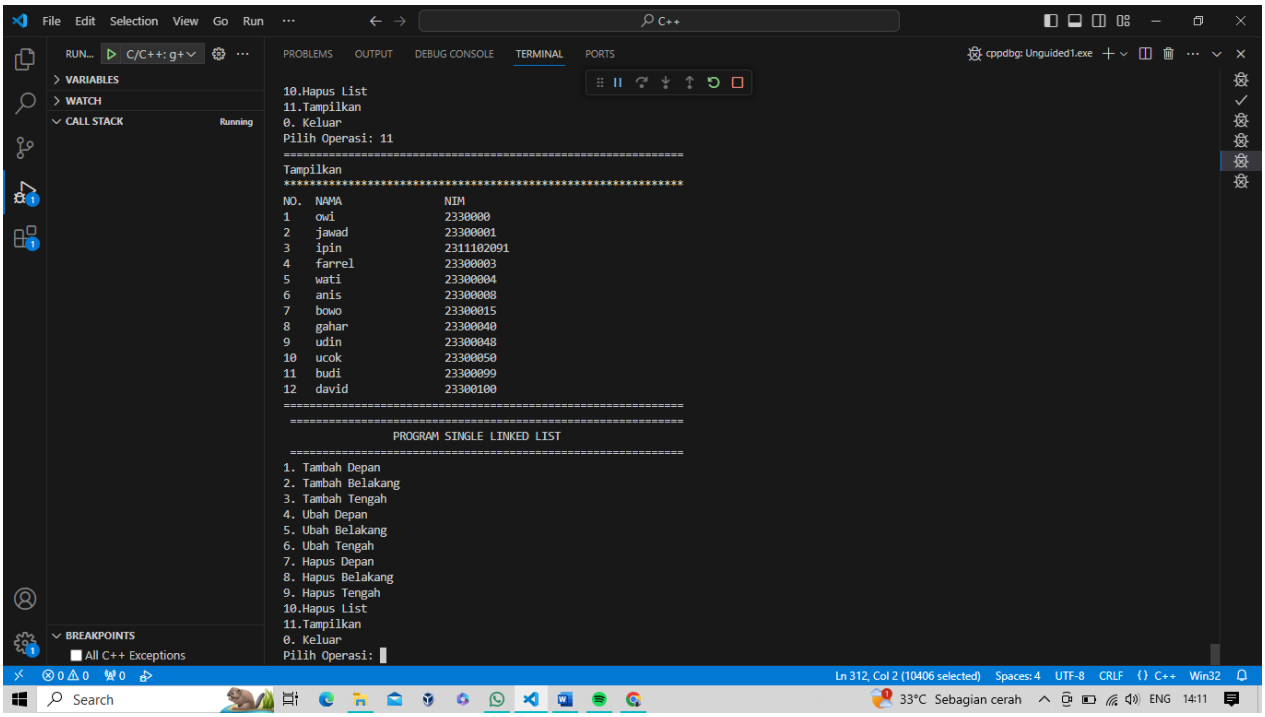
c) Tambahkan data berikut di awal:

Owi 2330000



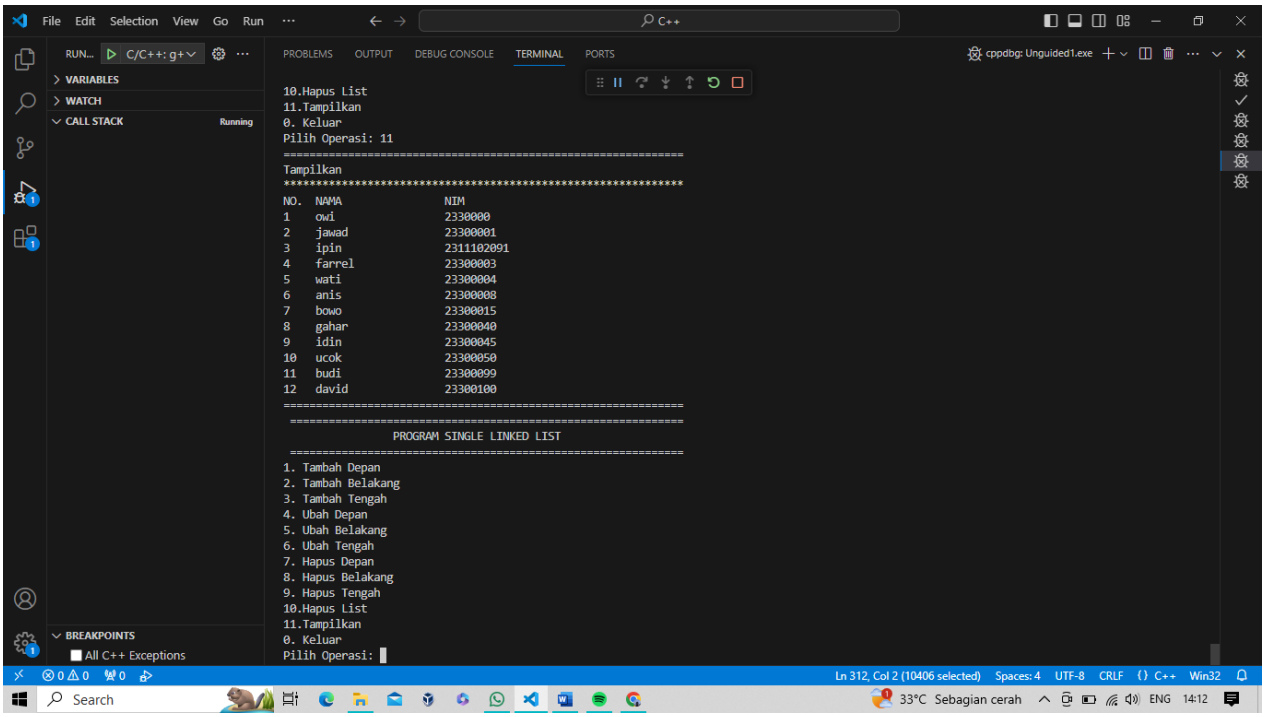
d) Tambahkan data berikut di akhir:

David 23300100



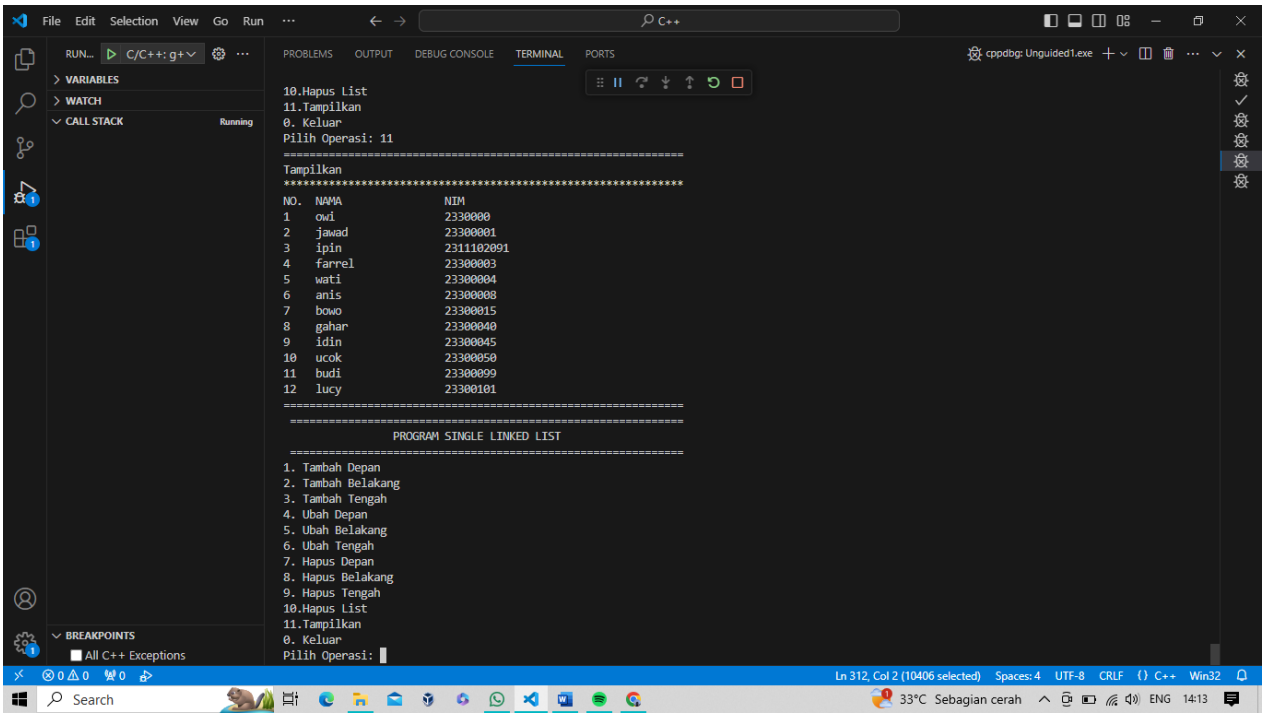
e) Ubah data Udin menjadi data berikut:

Idin 23300045

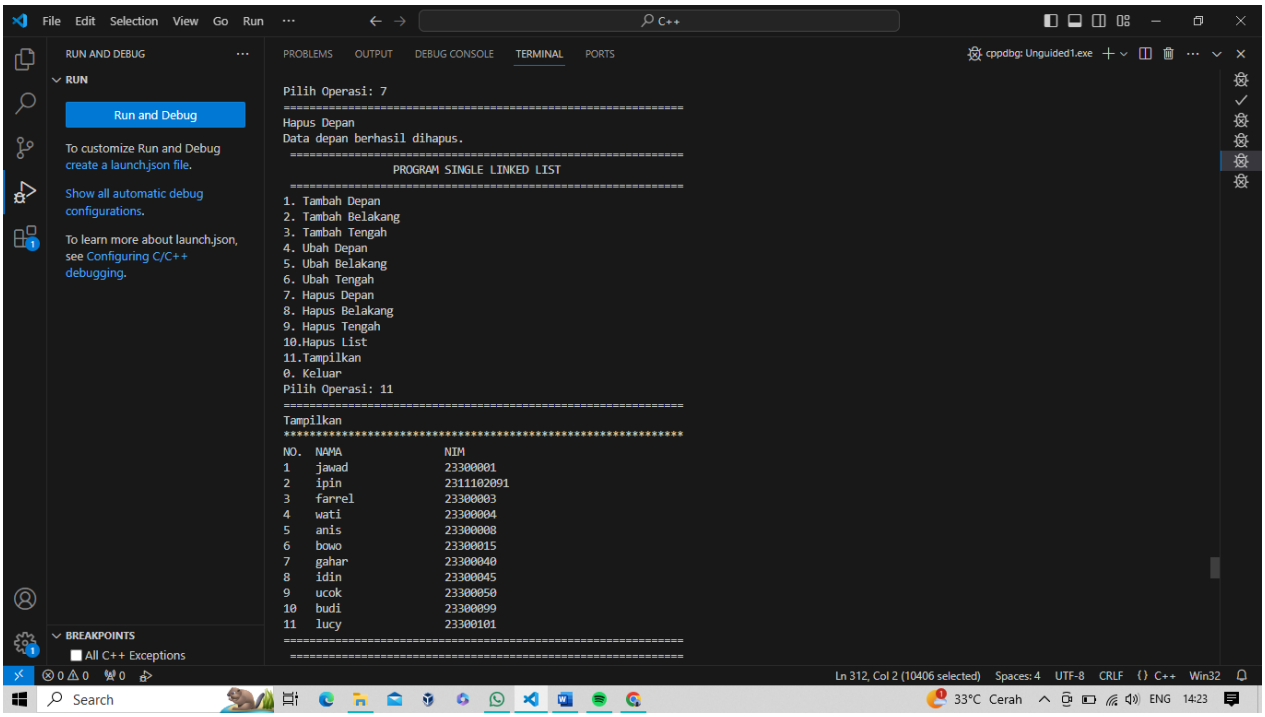


f) Ubah data terkahir menjadi berikut:

Lucy 23300101

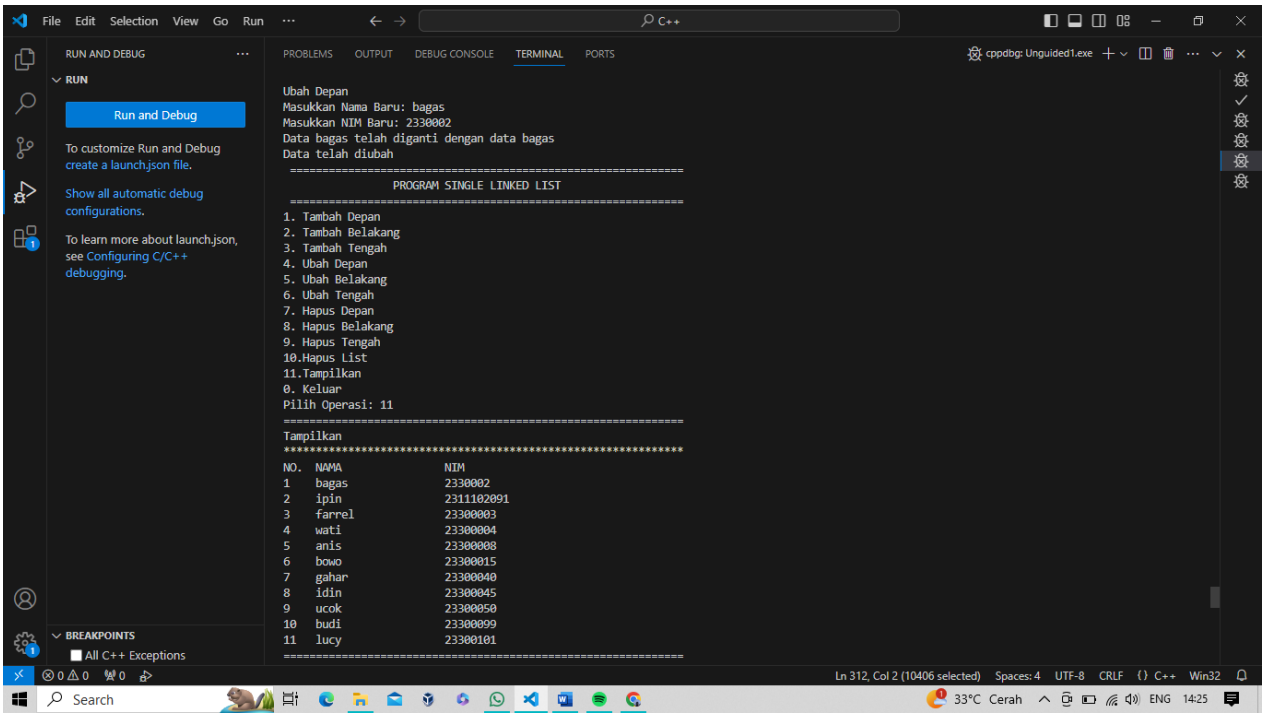


g) Hapus data awal

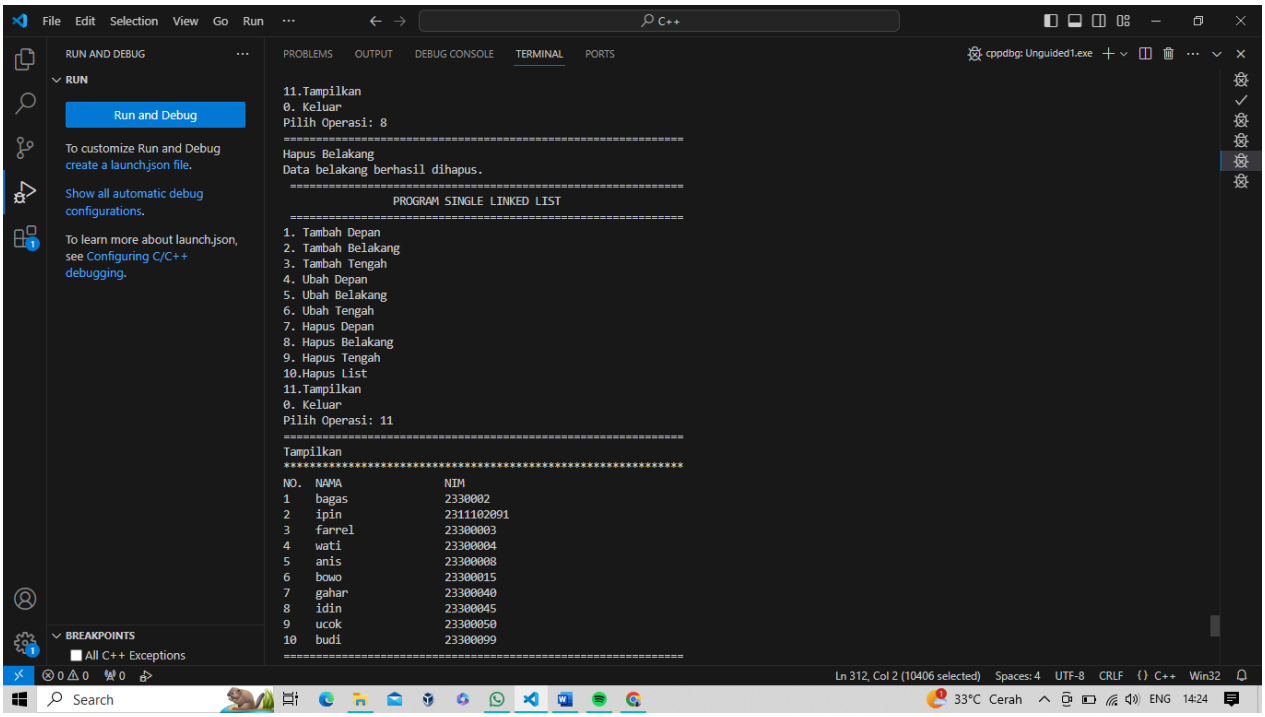


h) Ubah data awal menjadi berikut:

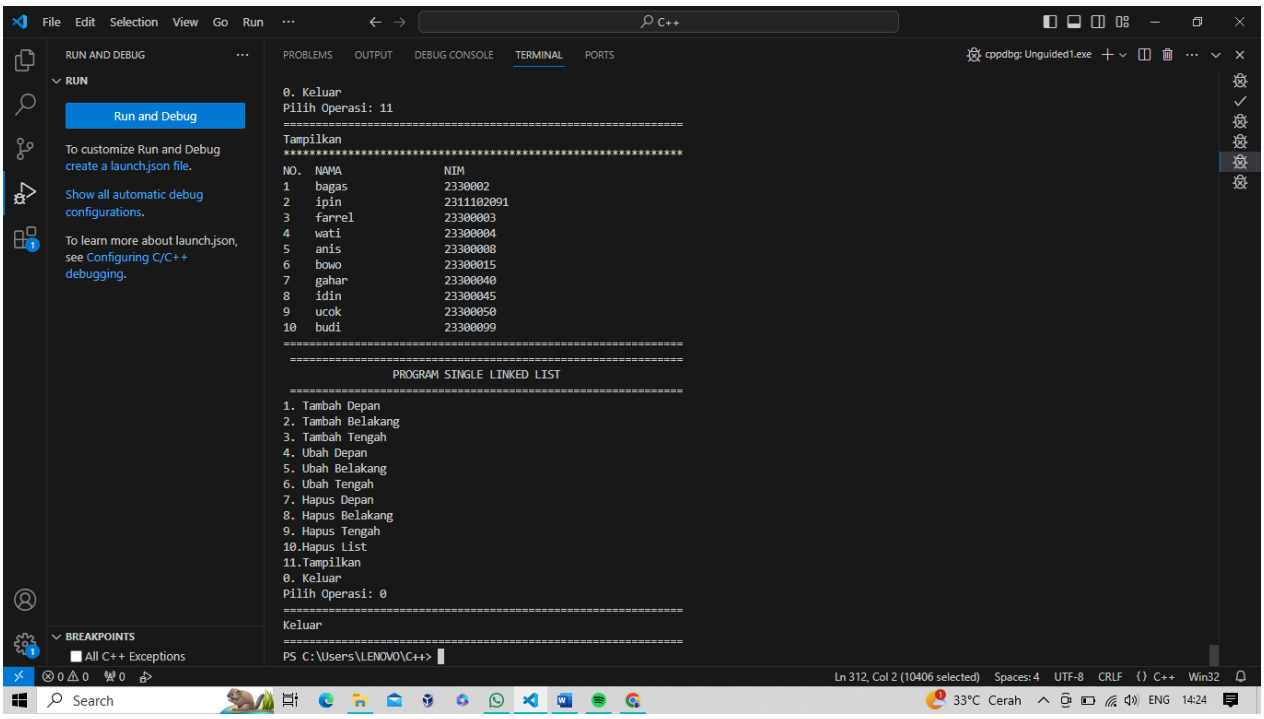
Bagas 2330002



i) Hapus data akhir



j) Tampilkan seluruh data



D. Kesimpulan

Kesimpulan dari ketiga kode program di atas adalah implementasi dari struktur data linked list dalam bahasa pemrograman C++. Ketiga kode tersebut memiliki fokus yang sama yaitu untuk menyediakan operasi dasar pada linked list seperti penambahan, penghapusan, pengubahan, dan penampilan elemen-elemen linked list. Masing-masing kode memiliki struktur dan pendekatan yang sedikit berbeda dalam implementasinya, namun tetap mencapai tujuan yang sama yaitu mengelola data dengan struktur linked list.

Kode pertama adalah implementasi linked list non-circular dengan menggunakan pendekatan berbasis fungsi. Struktur data linked list disusun dalam bentuk fungsi-fungsi terpisah yang masing-masing menangani operasi tertentu seperti penambahan, penghapusan, penghitungan jumlah elemen, dan penampilan isi linked list. Kode kedua merupakan implementasi linked list circular dengan menggunakan pendekatan berbasis class. Struktur data linked list didefinisikan dalam sebuah class yang memiliki berbagai fungsi untuk melakukan operasi-operasi dasar pada linked list. Program juga menyediakan menu untuk memilih operasi yang diinginkan oleh pengguna. Kode ketiga adalah implementasi linked list dengan pendekatan yang lebih modern menggunakan class dan struct, serta dilengkapi dengan menu interaktif untuk memilih operasi. Program ini juga menyediakan fitur-fitur tambahan seperti penambahan dan penghapusan elemen di posisi tertentu.

Secara keseluruhan, ketiga kode program tersebut menunjukkan cara yang berbeda dalam mengimplementasikan struktur data linked list dengan tujuan yang sama yaitu memungkinkan pengguna untuk melakukan operasi-operasi dasar pada linked list seperti penambahan, penghapusan, pengubahan, dan penampilan isi data.

E. Referensi

[1] Hamidah, N.A. (2020) 'Membedakan Single Linked List circular dan non circular,' *blog.spot*, 14 May. https://nadiaaprilianihamidah04.blogspot.com/2020/05/membedakan-single-linked-list-circular.html.

[2] Trivusi (2022b) 'Struktur Data Linked List: Pengertian, Karakteristik, dan Jenis-jenisnya,' *Trivusi*, 16 September. https://www.trivusi.web.id/2022/07/struktur-data-linked-list.html.