STRUKTUR DATA
Tugas Kelompok di kelas

NAMA:
UHAMMAD LUTHFI AZIZ SUNARYA 140810230081
BAGAS DIATAMA WARDOYO 140810230061
MUHAMAD HISYAM AZ-ZAHRAN 140810230065

Dikumpulkan tanggal :
06 Mei 2024

Universitas Padjadjaran
FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM
Program Studi S-1 Teknik Informatika
2024

1. **Source Code Doubly Circular**

```
/*
NAMA ANGGOTA 1       : MUHAMMAD LUTHFI AZIZ SUNARYA
140810230081
NAMA ANGGOTA 2       : BAGAS DIATAMA WARDOYO 140810230061
NAMA ANGGOTA 3       : MUHAMAD HISYAM AZ-ZAHRAN
140810230065
TANGGAL PEMBUATAN    : Senin, 06 - 05 - 2024
NAMA PROGRRAM        : DOUBLY CIRCULAR LINKED LIST
DESKRIPSI            : INI ADALAH PROGRAM DOUBLY CIRCULAR
LINKED LIST YANG BERISI DATA PEGAWAI DENGAN ATRIBUT NIP,
NAMA, GOLONGAN, GAJI DENGAN SEMUA FUNGSI PRIMITIVE
*/

#include <iostream>
#include <iomanip>

struct pegawai
{
    std::string NIP;
    std::string nama;
    int golongan;
    float gaji;
};

struct node
{
    pegawai data;
    node *next;
    node *prev;
};

typedef node *nodePtr;
typedef nodePtr list;

void gaji(nodePtr &newPtr);
// Menampilkan gaji pegawai berdasarkan golongan
void createList(list &head);
// Membuat list kosong
void inputElement(nodePtr &newPtr);
```

```
// Memasukkan element
void createElement(nodePtr &newPtr);
// Membuat element kosong
void insertFirst(list &head, nodePtr &newPtr);
// Menyisipkan data ke depan list
void insertLast(list &head, nodePtr &newPtr);
// Menyisipkan data ke akhir list
void deleteFirst(list &head, nodePtr &pDelete);
// Menghapus data pertama list
void deleteLast(list &head, nodePtr &pDelete);
// Menghapus data terakhir list
void traverse(list head);
// Menampilkan semua data dalam list
void searchList(nodePtr head, nodePtr &cari, std::string
search); // Mencari data dalam list dengan kunci NIP
void insertBefore(list &head, nodePtr cari, nodePtr
&newPtr);     // Menyisipkan data Sebelum NIP
void insertAfter(nodePtr cari, nodePtr &newPtr, list
&head);      // Menyisipkan data Setelah NIP
void deletePCari(list &head, nodePtr cari, nodePtr
&pDelete);     // Menghapus data yang dicari
void deleteAfter(nodePtr cari, nodePtr &pDelete, list
&head);     // Menghapus data setelah yang dicari
void deleteBefore(list &head, nodePtr cari, nodePtr
&pDelete);    // Menghapus data sebelum yang dicari
float average(list head); //Rata rata gaji
void tabel();
void output(list head);

int main()
{
    list head;
    createList(head);
    nodePtr nodeBaru, pDelete, cari;
    int pilih;
    std::string NIP;

    do
    {
        std::cout << "\nMenu:\n"
```

```
                << "1. Tambah Pegawai (First)\n"
                << "2. Tambah Pegawai (Last)\n"
                << "3. Hapus Pegawai (First)\n"
                << "4. Hapus Pegawai (Last)\n"
                << "5. Tampilkan Pegawai\n"
                << "6. Cari Pegawai\n"
                << "7. Insert Before\n"
                << "8. Insert After\n"
                << "9. Hapus Pegawai Berdasarkan NIP\n"
                << "10. Hapus Pegawai Setelah\n"
                << "11. Output\n"
                << "12. Hapus Sebelum\n"
                << "13. Keluar\n"
                << "Pilih: ";
        std::cin >> pilih;
        switch (pilih)
        {
        case 1:
            createElement(nodeBaru);
            insertFirst(head, nodeBaru);
            break;
        case 2:
            createElement(nodeBaru);
            insertLast(head, nodeBaru);
            break;
        case 3:
            deleteFirst(head, pDelete);
            break;
        case 4:
            deleteLast(head, pDelete);
            break;
        case 5:
            traverse(head);
            break;
        case 6:
            std::cout << "Masukkan NIP yang dicari: ";
            std::cin >> NIP;
            searchList(head, cari, NIP);
            break;
        case 7:
```

```
            std::cout << "Masukkan NIP: ";
            std::cin >> NIP;
            searchList(head, cari, NIP);
            if (cari != nullptr)
            {
                createElement(nodeBaru);
                insertBefore(head, cari, nodeBaru);
            }
            else
            {
                std::cout << "Data tidak ditemukan,
insert gagal.\n";
            }
            break;
        case 8:
            std::cout << "Masukkan NIP: ";
            std::cin >> NIP;
            searchList(head, cari, NIP);
            if (cari != nullptr)
            {
                createElement(nodeBaru);
                insertAfter(cari, nodeBaru, head);
            }
            else
            {
                std::cout << "Data tidak ditemukan,
insert gagal.\n";
            }
            break;
        case 9:
            std::cout << "Masukkan NIP yang ingin
dihapus: ";
            std::cin >> NIP;
            searchList(head, cari, NIP);
            if (cari != nullptr)
            {
                deletePCari(head, cari, pDelete);
            }
            else
            {
```

```
                std::cout << "Data tidak ditemukan,
delete gagal.\n";
            }
            break;
        case 10:
            std::cout << "Masukkan NIP: ";
            std::cin >> NIP;
            searchList(head, cari, NIP);
            if (cari != nullptr)
            {
                deleteAfter(cari, pDelete, head);
            }
            else
            {
                std::cout << "Data tidak ditemukan,
delete gagal.\n";
            }
            break;
        case 11:
            output(head);
            break;
        case 12:
            std::cout << "Masukkan NIP: ";
            std::cin >> NIP;
            searchList(head, cari, NIP);
            if (cari != nullptr)
            {
                deleteBefore(head, cari, pDelete);
            }
            else
            {
                std::cout << "Data tidak ditemukan,
delete gagal.\n";
            }
            break;
        case 13:
            std::cout << "Program selesai.\n";
            break;
        default:
            pilih = 13;
```

```
            break;
        }
    } while (pilih != 13);

    return 0;
}

void gaji(nodePtr &newPtr)
{
    switch (newPtr->data.golongan)
    {
    case 1:
        newPtr->data.gaji = 2.5;
        break;
    case 2:
        newPtr->data.gaji = 3.5;
        break;
    case 3:
        newPtr->data.gaji = 5;
        break;
    case 4:
        newPtr->data.gaji = 7.5;
        break;
    default:
        break;
    }
}

void createList(list &head)
{
    head = nullptr;
}

void inputElement(nodePtr &newPtr)
{
    std::cout << "Masukkan NIP : ";
    std::cin >> newPtr->data.NIP;
    std::cout << "Masukkan nama : ";
    std::cin >> newPtr->data.nama;
    std::cout << "Masukkan golongan : ";
```

```
    std::cin >> newPtr->data.golongan;
    gaji(newPtr);
}

void createElement(nodePtr &newPtr)
{
    newPtr = new node;
    inputElement(newPtr);
    newPtr->next = nullptr;
    newPtr->prev = nullptr;
}

void insertFirst(list &head, nodePtr &newPtr)
{
    if (head == nullptr)
    {
        head = newPtr;
        head->next = head;
        head->prev = head;
    }
    else
    {
        newPtr->next = head;
        newPtr->prev = head->prev;
        head->prev->next = newPtr;
        head->prev = newPtr;
        head = newPtr;
    }
}

void insertLast(list &head, nodePtr &newPtr)
{
    if (head == nullptr)
    {
        insertFirst(head, newPtr);
    }
    else
    {
        newPtr->next = head;
        newPtr->prev = head->prev;
```

```
        head->prev->next = newPtr;
        head->prev = newPtr;
    }
}

void deleteFirst(list &head, nodePtr &pDelete)
{
    if (head != nullptr)
    {
        pDelete = head;
        head = head->next;
        if (head != nullptr)
        {
            head->prev = pDelete->prev;
            pDelete->prev->next = head;
        }
        delete pDelete;
    }
}

void deleteLast(list &head, nodePtr &pDelete)
{
    if (head != nullptr)
    {
        pDelete = head->prev;
        head->prev = pDelete->prev;
        pDelete->prev->next = head;
        delete pDelete;
    }
}

void traverse(list head)
{
    if (head != nullptr)
    {
        nodePtr current = head;
        do
        {
            std::cout << "NIP: " << current->data.NIP <<
", Nama: " << current->data.nama << ", Golongan: " <<
```

```
current->data.golongan << ", Gaji: " <<
current->data.gaji << std::endl;
            current = current->next;
        } while (current != head);
    }
}

void searchList(nodePtr head, nodePtr &cari, std::string
search)
{
    bool found = false;
    cari = head;
    while (found == false && cari != nullptr)
    {
        if (cari->data.NIP == search)
        {
            found = true;
            std::cout << "Data ditemukan\n";
            break;
        }
        else
        {
            cari = cari->next;
        }
    }
    if (found == false)
    {
        std::cout << "Data tidak ditemukan\n";
        cari = nullptr;
    }
}

void insertBefore(list &head, nodePtr cari, nodePtr
&newPtr)
{
    if (cari->prev == nullptr)
    {
        insertFirst(head, newPtr);
    }
    else
```

```
    {
        newPtr->next = cari;
        newPtr->prev = cari->prev;
        cari->prev->next = newPtr;
        cari->prev = newPtr;
    }
}

void insertAfter(nodePtr cari, nodePtr &newPtr, list
&head)
{
    if (cari->next == nullptr)
    {
        insertLast(head, newPtr);
    }
    else
    {
        newPtr->next = cari->next;
        newPtr->prev = cari;
        cari->next->prev = newPtr;
        cari->next = newPtr;
    }
}

void deletePCari(list &head, nodePtr cari, nodePtr
&pDelete)
{
    if (cari == head)
    {
        deleteFirst(head, pDelete);
    }
    else if (cari->next == head)
    {
        deleteLast(head, pDelete);
    }
    else
    {
        cari->prev->next = cari->next;
        cari->next->prev = cari->prev;
        delete cari;
```

```
    }
}

void deleteAfter(nodePtr cari, nodePtr &pDelete, list
&head)
{
    if (cari->next == head)
    {
        deleteFirst(head, pDelete);
    }
    else
    {
        pDelete = cari->next;
        cari->next = pDelete->next;
        pDelete->next->prev = cari;
        delete pDelete;
    }
}

void deleteBefore(list &head, nodePtr cari, nodePtr
&pDelete)
{
    if (head != nullptr && cari != head && cari !=
head->next)
    {
        pDelete = cari->prev;
        pDelete->prev->next = cari;
        cari->prev = pDelete->prev;
        if (pDelete == head->prev)
        {                       // jika yang dihapus
adalah node terakhir
            head->prev = cari; // update tail
        }
        delete pDelete;
    }
    else
    {
        std::cout << "Tidak dapat menghapus sebelum head
atau node pertama." << std::endl;
    }
```

```
}

float average(list head)
{
    nodePtr current = head;
    int jumlahList = 0;
    float jumlahGaji = 0;
    float average = 0;
    if (current != nullptr)
    {
        do
        {
            jumlahGaji += current->data.gaji;
            jumlahList++;
            current = current->next;
        } while (current != head);
        average = jumlahGaji / jumlahList;
    }
    return average;
}

void tabel()
{
    for (int i = 0; i < 55; i++)
    {
        std::cout << "-";
    }
    std::cout << std::endl;
}

void output(list head)
{
    if (head == nullptr)
    {
        std::cout << "Linked list kosong." << std::endl;
        return;
    }

    nodePtr current = head;
    int no = 1;
```

```
    float gajiTerendah = current->data.gaji;
    float gajiTertinggi = current->data.gaji;
    std::cout << "           DATA GAJI PEGAWAI PT ABCD.
tbk \n";
    tabel();
    std::cout << std::left << std::setw(5) << "No"
              << std::setw(15) << "NIP"
              << std::setw(20) << "Nama"
              << std::setw(10) << "Golongan"
              << std::setw(10) << "Gaji" << std::endl;
    tabel();
    while (current->next != head)
    {
        std::cout << std::setw(5) << no++
                  << std::setw(15) << std::left <<
current->data.NIP
                  << std::setw(20) << std::left <<
current->data.nama
                  << std::setw(10) <<
current->data.golongan
                  << std::setw(10) << std::fixed <<
std::setprecision(2) << current->data.gaji << std::endl;

        // Cek gaji terendah
        if (current->data.gaji < gajiTerendah)
        {
            gajiTerendah = current->data.gaji;
        }

        // Cek gaji tertinggi
        if (current->data.gaji > gajiTertinggi)
        {
            gajiTertinggi = current->data.gaji;
        }

        current = current->next;
    }

    // Output data untuk node terakhir
    std::cout << std::setw(5) << no
```

```
                << std::setw(15) << std::left <<
current->data.NIP
                << std::setw(20) << std::left <<
current->data.nama
                << std::setw(10) << current->data.golongan
                << std::setw(10) << std::fixed <<
std::setprecision(2) << current->data.gaji << std::endl;

    tabel();
    std::cout << "Gaji Rata - rata : " << std::fixed <<
std::setprecision(2) << average(head) << std::endl;
    std::cout << "Gaji Terendah  : " << std::fixed <<
std::setprecision(2) << gajiTerendah << std::endl;
    std::cout << "Gaji Tertinggi : " << std::fixed <<
std::setprecision(2) << gajiTertinggi << std::endl;
}
```

Hasil Run

```
Menu:
1. Tambah Pegawai (First)
2. Tambah Pegawai (Last)
3. Hapus Pegawai (First)
4. Hapus Pegawai (Last)
5. Tampilkan Pegawai
6. Cari Pegawai
7. Insert Before
8. Insert After
9. Hapus Pegawai Berdasarkan NIP
10. Hapus Pegawai Setelah
11. Output
12. Hapus Sebelum
13. Keluar
Pilih: 1
Masukkan NIP : 001
Masukkan nama : Bagas
Masukkan golongan : 2

Menu:
1. Tambah Pegawai (First)
```

```
2. Tambah Pegawai (Last)
3. Hapus Pegawai (First)
4. Hapus Pegawai (Last)
5. Tampilkan Pegawai
6. Cari Pegawai
7. Insert Before
8. Insert After
9. Hapus Pegawai Berdasarkan NIP
10. Hapus Pegawai Setelah
11. Output
12. Hapus Sebelum
13. Keluar
Pilih: 1
Masukkan NIP : 002
Masukkan nama : Diatama
Masukkan golongan : 3

Menu:
1. Tambah Pegawai (First)
2. Tambah Pegawai (Last)
3. Hapus Pegawai (First)
4. Hapus Pegawai (Last)
5. Tampilkan Pegawai
6. Cari Pegawai
7. Insert Before
8. Insert After
9. Hapus Pegawai Berdasarkan NIP
10. Hapus Pegawai Setelah
11. Output
12. Hapus Sebelum
13. Keluar
Pilih: 1
Masukkan NIP : 005
Masukkan nama : Wardoyo
Masukkan golongan : 2

Menu:
1. Tambah Pegawai (First)
2. Tambah Pegawai (Last)
```

```
3. Hapus Pegawai (First)
4. Hapus Pegawai (Last)
5. Tampilkan Pegawai
6. Cari Pegawai
7. Insert Before
8. Insert After
9. Hapus Pegawai Berdasarkan NIP
10. Hapus Pegawai Setelah
11. Output
12. Hapus Sebelum
13. Keluar
Pilih: 11
           DATA GAJI PEGAWAI PT ABCD. tbk
-----------------------------------------------------
No   NIP            Nama                Golongan  Gaji
-----------------------------------------------------
1    005            Wardoyo             2         3.50
2    002            Diatama             3         5.00
3    001            Bagas               2         3.50
-----------------------------------------------------
Gaji Rata - rata : 4.00
Gaji Terendah  : 3.50
Gaji Tertinggi : 5.00

Menu:
1. Tambah Pegawai (First)
2. Tambah Pegawai (Last)
3. Hapus Pegawai (First)
4. Hapus Pegawai (Last)
5. Tampilkan Pegawai
6. Cari Pegawai
7. Insert Before
8. Insert After
9. Hapus Pegawai Berdasarkan NIP
10. Hapus Pegawai Setelah
11. Output
12. Hapus Sebelum
13. Keluar
Pilih: 12
```

```
Masukkan NIP: 001
Data ditemukan

Menu:
1. Tambah Pegawai (First)
2. Tambah Pegawai (Last)
3. Hapus Pegawai (First)
4. Hapus Pegawai (Last)
5. Tampilkan Pegawai
6. Cari Pegawai
7. Insert Before
8. Insert After
9. Hapus Pegawai Berdasarkan NIP
10. Hapus Pegawai Setelah
11. Output
12. Hapus Sebelum
13. Keluar
Pilih: 11
           DATA GAJI PEGAWAI PT ABCD. tbk
-----------------------------------------------------
No   NIP            Nama                 Golongan  Gaji
-----------------------------------------------------
1    005            Wardoyo              2         3.50
2    001            Bagas                2         3.50
-----------------------------------------------------
Gaji Rata - rata : 3.50
Gaji Terendah  : 3.50
Gaji Tertinggi : 3.50
```

2. **Source Code Single Circular**

```
/*
NAMA ANGGOTA 1: MUHAMMAD LUTHFI AZIZ SUNARYA 140810230081
NAMA ANGGOTA 2: BAGAS DIATAMA WARDOYO 140810230061
NAMA ANGGOTA 3: MUHAMAD HISYAM AZ-ZAHRAN 140810230065
TANGGAL PEMBUATAN : Senin, 06 - 05 - 2024
NAMA PROGRRAM : SINGLE CIRCULAR LINKED LIST
DESKRIPSI : INI ADALAH PROGRAM SINGLE CIRCULAR LINKED LIST YANG
BERISI DATA PEGAWAI DENGAN ATRIBUT NIP, NAMA, GOLONGAN, GAJI DENGAN
SEMUA FUNGSI PRIMITIVE
```

```
*/

#include <iostream>
#include <iomanip>

struct pegawai
{
    std::string NIP;
    std::string nama;
    int golongan;
    float gaji;
};

struct node
{
    pegawai data;
    node *next;
};

typedef node *nodePtr;
typedef nodePtr list;

void gaji(nodePtr &newPtr)
{
    switch (newPtr->data.golongan)
    {
    case 1:
        newPtr->data.gaji = 2.5;
        break;
    case 2:
        newPtr->data.gaji = 3.5;
        break;
    case 3:
        newPtr->data.gaji = 5;
        break;
    case 4:
        newPtr->data.gaji = 7.5;
        break;
```

```
    default:
        break;
    }
}

void createList(list &head)
{
    head = nullptr;
}

void inputElement(nodePtr &newPtr)
{
    std::cout << "Masukkan NIP : ";
    std::cin >> newPtr->data.NIP;
    std::cout << "Masukkan nama : ";
    std::cin >> newPtr->data.nama;
    std::cout << "Masukkan golongan : ";
    std::cin >> newPtr->data.golongan;
    gaji(newPtr);
}

void createElement(nodePtr &newPtr)
{
    newPtr = new node;
    inputElement(newPtr);
    newPtr->next = nullptr;
}

void insertFirst(list &head, nodePtr &newPtr)
{
    if (head == nullptr)
    {
        head = newPtr;
        newPtr->next = head; // menunjuk kembali ke head
    }
    else
    {
        newPtr->next = head->next;
```

```
        head->next = newPtr;
    }
}

void insertLast(list &head, nodePtr &newPtr)
{
    if (head == nullptr)
    {
        insertFirst(head, newPtr);
    }
    else
    {
        newPtr->next = head->next;
        head->next = newPtr;
        head = newPtr; // update head
    }
}

void deleteFirst(list &head, nodePtr &pDelete)
{
    if (head != nullptr)
    {
        pDelete = head->next;
        if (pDelete == head)
        { // satu-satunya node
            head = nullptr;
        }
        else
        {
            head->next = pDelete->next;
        }
        delete pDelete;
    }
}

void deleteLast(list &head, nodePtr &pDelete)
{
    if (head != nullptr)
```

```
    {
        nodePtr current = head;
        while (current->next != head)
        {
            current = current->next;
        }
        pDelete = current->next;
        current->next = pDelete->next;
        head = current; // update head
        delete pDelete;
    }
}

void traverse(list head)
{
    if (head != nullptr)
    {
        nodePtr current = head->next; // mulai dari node pertama
        do
        {
            std::cout << "NIP: " << current->data.NIP << ", Nama: "
<< current->data.nama << ", Golongan: " << current->data.golongan <<
", Gaji: " << current->data.gaji << std::endl;
            current = current->next;
        } while (current != head->next); // berhenti saat kembali ke
head
    }
}

void searchList(nodePtr head, nodePtr &cari, std::string search)
{
    bool found = false;
    cari = head->next;
    while (found == false && cari != head)
    {
        if (cari->data.NIP == search)
        {
            found = true;
```

```
            std::cout << "Data ditemukan\n";
            break;
        }
        else
        {
            cari = cari->next;
        }
    }
    if (found == false)
    {
        std::cout << "Data tidak ditemukan\n";
        cari = nullptr;
    }
}

void insertBefore(list &head, nodePtr cari, nodePtr &newPtr)
{
    if (cari == head->next)
    {
        insertFirst(head, newPtr);
    }
    else
    {
        nodePtr current = head->next;
        while (current->next != cari)
        {
            current = current->next;
        }
        newPtr->next = cari;
        current->next = newPtr;
    }
}

void insertAfter(nodePtr cari, nodePtr &newPtr, list &head)
{
    newPtr->next = cari->next;
    cari->next = newPtr;
    if (cari == head)
```

```
    {                   // jika yang ditambahkan setelah head
        head = newPtr; // update head
    }
}

void deletePCari(list &head, nodePtr cari, nodePtr &pDelete)
{
    if (cari->next == cari)
    { // satu-satunya node
        head = nullptr;
    }
    else
    {
        nodePtr current = head;
        while (current->next != cari)
        {
            current = current->next;
        }
        current->next = cari->next;
        if (cari == head)
        {                   // jika node yang dihapus adalah head
            head = current; // update head
        }
    }
    delete cari;
}

void deleteAfter(nodePtr cari, nodePtr &pDelete, list &head)
{
    pDelete = cari->next;
    cari->next = pDelete->next;
    if (pDelete == head)
    {                 // jika node yang dihapus adalah head
        head = cari; // update head
    }
    delete pDelete;
}
```

```
void deleteBefore(list &head, nodePtr cari, nodePtr &pDelete)
{
    if (head != nullptr && cari != head->next && cari != head)
    {
        nodePtr current = head->next;
        while (current->next != cari)
        {
            current = current->next;
        }
        pDelete = current;
        current->next = cari;
        if (pDelete == head)
        {                // jika yang dihapus adalah head
            head = cari; // update head
        }
        delete pDelete;
    }
    else
    {
        std::cout << "Tidak dapat menghapus sebelum head atau node
pertama." << std::endl;
    }
}

float average(list head)
{
    if (head != nullptr)
    {
        nodePtr current = head->next;
        int jumlahList = 0;
        float jumlahGaji = 0;
        do
        {
            jumlahGaji += current->data.gaji;
            jumlahList++;
            current = current->next;
        } while (current != head->next);
        return jumlahGaji / jumlahList;
```

```
    }
    return 0;
}

void tabel()
{
    for (int i = 0; i < 55; i++)
    {
        std::cout << "-";
    }
    std::cout << std::endl;
}

void output(list head)
{
    if (head == nullptr)
    {
        std::cout << "Linked list kosong." << std::endl;
        return;
    }

    nodePtr current = head->next;
    int no = 1;
    float gajiTerendah = current->data.gaji;
    float gajiTertinggi = current->data.gaji;
    std::cout << "           DATA GAJI PEGAWAI PT ABCD. tbk \n";
    tabel();
    std::cout << std::left << std::setw(5) << "No"
              << std::setw(15) << "NIP"
              << std::setw(20) << "Nama"
              << std::setw(10) << "Golongan"
              << std::setw(10) << "Gaji" << std::endl;
    tabel();
    do
    {
        std::cout << std::setw(5) << no++
                  << std::setw(15) << std::left << current->data.NIP
                  << std::setw(20) << std::left <<
```

```
current->data.nama
                  << std::setw(10) << current->data.golongan
                  << std::setw(10) << std::fixed <<
std::setprecision(2) << current->data.gaji << std::endl;

        // Cek gaji terendah
        if (current->data.gaji < gajiTerendah)
        {
            gajiTerendah = current->data.gaji;
        }

        // Cek gaji tertinggi
        if (current->data.gaji > gajiTertinggi)
        {
            gajiTertinggi = current->data.gaji;
        }

        current = current->next;
    } while (current != head->next);

    tabel();
    std::cout << "Gaji Rata - rata : " << std::fixed <<
std::setprecision(2) << average(head) << std::endl;
    std::cout << "Gaji Terendah  : " << std::fixed <<
std::setprecision(2) << gajiTerendah << std::endl;
    std::cout << "Gaji Tertinggi : " << std::fixed <<
std::setprecision(2) << gajiTertinggi << std::endl;
}

int main()
{
    list head;
    createList(head);
    nodePtr nodeBaru, pDelete, cari;
    int pilih;
    std::string NIP;

    do
```

```
{
    std::cout << "\nMenu:\n"
              << "1. Tambah Pegawai (First)\n"
              << "2. Tambah Pegawai (Last)\n"
              << "3. Hapus Pegawai (First)\n"
              << "4. Hapus Pegawai (Last)\n"
              << "5. Tampilkan Pegawai\n"
              << "6. Cari Pegawai\n"
              << "7. Insert Before\n"
              << "8. Insert After\n"
              << "9. Hapus Pegawai Berdasarkan NIP\n"
              << "10. Hapus Pegawai Setelah\n"
              << "11. Output\n"
              << "12. Hapus Pegawai Sebelum\n"
              << "13. Keluar\n"
              << "Pilih: ";
    std::cin >> pilih;
    switch (pilih)
    {
    case 1:
        createElement(nodeBaru);
        insertFirst(head, nodeBaru);
        break;
    case 2:
        createElement(nodeBaru);
        insertLast(head, nodeBaru);
        break;
    case 3:
        deleteFirst(head, pDelete);
        break;
    case 4:
        deleteLast(head, pDelete);
        break;
    case 5:
        traverse(head);
        break;
    case 6:
        std::cout << "Masukkan NIP yang dicari: ";
```

```
            std::cin >> NIP;
            searchList(head, cari, NIP);
            break;
        case 7:
            std::cout << "Masukkan NIP: ";
            std::cin >> NIP;
            searchList(head, cari, NIP);
            if (cari != nullptr)
            {
                createElement(nodeBaru);
                insertBefore(head, cari, nodeBaru);
            }
            else
            {
                std::cout << "Data tidak ditemukan, insert
gagal.\n";
            }
            break;
        case 8:
            std::cout << "Masukkan NIP: ";
            std::cin >> NIP;
            searchList(head, cari, NIP);
            if (cari != nullptr)
            {
                createElement(nodeBaru);
                insertAfter(cari, nodeBaru, head);
            }
            else
            {
                std::cout << "Data tidak ditemukan, insert
gagal.\n";
            }
            break;
        case 9:
            std::cout << "Masukkan NIP yang ingin dihapus: ";
            std::cin >> NIP;
            searchList(head, cari, NIP);
            if (cari != nullptr)
```

```
            {
                deletePCari(head, cari, pDelete);
            }
            else
            {
                std::cout << "Data tidak ditemukan, delete
gagal.\n";
            }
            break;
        case 10:
            std::cout << "Masukkan NIP: ";
            std::cin >> NIP;
            searchList(head, cari, NIP);
            if (cari != nullptr)
            {
                deleteAfter(cari, pDelete, head);
            }
            else
            {
                std::cout << "Data tidak ditemukan, delete
gagal.\n";
            }
            break;
        case 11:
            output(head);
            break;
        case 12:
            std::cout << "Masukkan NIP: ";
            std::cin >> NIP;
            searchList(head, cari, NIP);
            if (cari != nullptr)
            {
                deleteBefore(head, cari, pDelete);
            }
            else
            {
                std::cout << "Data tidak ditemukan, delete
gagal.\n";
```

```
            }
            break;
        case 13:
            std::cout << "Program selesai.\n";
            break;
        default:
            pilih = 13;
            break;
        }
    } while (pilih != 13);

    return 0;
}
```

Hasil Run

Menu:
1. Tambah Pegawai (First)
2. Tambah Pegawai (Last)
3. Hapus Pegawai (First)
4. Hapus Pegawai (Last)
5. Tampilkan Pegawai
6. Cari Pegawai
7. Insert Before
8. Insert After
9. Hapus Pegawai Berdasarkan NIP
10. Hapus Pegawai Setelah
11. Output
12. Hapus Pegawai Sebelum
13. Keluar

Pilih: 1
Masukkan NIP : 001
Masukkan nama : Bagas
Masukkan golongan : 2

Menu:
1. Tambah Pegawai (First)
2. Tambah Pegawai (Last)
3. Hapus Pegawai (First)
4. Hapus Pegawai (Last)
5. Tampilkan Pegawai
6. Cari Pegawai
7. Insert Before

8. Insert After
9. Hapus Pegawai Berdasarkan NIP
10. Hapus Pegawai Setelah
11. Output
12. Hapus Pegawai Sebelum
13. Keluar
Pilih: 11

DATA GAJI PEGAWAI PT ABCD. tbk

No	NIP	Nama	Golongan	Gaji
1	001	Bagas	2	3.50

Gaji Rata - rata : 3.50
Gaji Terendah : 3.50
Gaji Tertinggi : 3.50

Menu:
1. Tambah Pegawai (First)
2. Tambah Pegawai (Last)
3. Hapus Pegawai (First)
4. Hapus Pegawai (Last)
5. Tampilkan Pegawai
6. Cari Pegawai
7. Insert Before
8. Insert After
9. Hapus Pegawai Berdasarkan NIP
10. Hapus Pegawai Setelah
11. Output
12. Hapus Pegawai Sebelum
13. Keluar
Pilih: 2
Masukkan NIP : 003
Masukkan nama : Hisyam
Masukkan golongan : 3

Menu:
1. Tambah Pegawai (First)
2. Tambah Pegawai (Last)
3. Hapus Pegawai (First)
4. Hapus Pegawai (Last)
5. Tampilkan Pegawai
6. Cari Pegawai
7. Insert Before
8. Insert After
9. Hapus Pegawai Berdasarkan NIP

10. Hapus Pegawai Setelah
11. Output
12. Hapus Pegawai Sebelum
13. Keluar
Pilih: 11

DATA GAJI PEGAWAI PT ABCD. tbk

No	NIP	Nama	Golongan	Gaji
1	001	Bagas	2	3.50
2	003	Hisyam	3	5.00

Gaji Rata - rata : 4.25
Gaji Terendah : 3.50
Gaji Tertinggi : 5.00

Menu:
1. Tambah Pegawai (First)
2. Tambah Pegawai (Last)
3. Hapus Pegawai (First)
4. Hapus Pegawai (Last)
5. Tampilkan Pegawai
6. Cari Pegawai
7. Insert Before
8. Insert After
9. Hapus Pegawai Berdasarkan NIP
10. Hapus Pegawai Setelah
11. Output
12. Hapus Pegawai Sebelum
13. Keluar
Pilih: 1
Masukkan NIP : 002
Masukkan nama : Luhfi
Masukkan golongan : 1

Menu:
1. Tambah Pegawai (First)
2. Tambah Pegawai (Last)
3. Hapus Pegawai (First)
4. Hapus Pegawai (Last)
5. Tampilkan Pegawai
6. Cari Pegawai
7. Insert Before
8. Insert After
9. Hapus Pegawai Berdasarkan NIP
10. Hapus Pegawai Setelah

11. Output
12. Hapus Pegawai Sebelum
13. Keluar
Pilih: 11

DATA GAJI PEGAWAI PT ABCD. tbk

No	NIP	Nama	Golongan	Gaji
1	002	Luhfi	1	2.50
2	001	Bagas	2	3.50
3	003	Hisyam	3	5.00

Gaji Rata - rata : 3.67
Gaji Terendah : 2.50
Gaji Tertinggi : 5.00

Menu:
1. Tambah Pegawai (First)
2. Tambah Pegawai (Last)
3. Hapus Pegawai (First)
4. Hapus Pegawai (Last)
5. Tampilkan Pegawai
6. Cari Pegawai
7. Insert Before
8. Insert After
9. Hapus Pegawai Berdasarkan NIP
10. Hapus Pegawai Setelah
11. Output
12. Hapus Pegawai Sebelum
13. Keluar
Pilih: 6
Masukkan NIP yang dicari: 002
Data ditemukan