PRAKTIKUM STRUKTUR DATA
TUGAS PENDAHULUAN 4

Nama :

Ardhian Dwi Saputra (2311104040)

Dosen :
Yudha Islami Sulistya,S. Kom.
,M.Kom.

PROGRAM STUDI S1 REKAYASA PERANGKAT LUNAK

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2024

B. Soal Tugas Pendahuluan

1. Header File `list.h` (Belum Ditampilkan)

   Program ini mengandalkan header file `list.h`, yang mendeklarasikan struktur data dan tipe tipe yang digunakan dalam implementasi linked list.

```
#include "list.h"
using namespace std;
```

2. Fungsi `createList(List &L)`

   Fungsi ini menginisialisasi list yang diberikan (`L`) dengan membuat elemen pertama (`first(L)`) bernilai `NULL`. Ini berarti list awalnya kosong.

```
void createList(List &L) {
    first(L) = NULL;
}
```

3. Fungsi `allocate(infoType x)`

   Fungsi ini bertugas untuk membuat elemen baru dalam linked list. Elemen baru ini diberi nilai `x` sebagai informasi, dan pointer ke elemen berikutnya (`next(p)`) diinisialisasi dengan `NULL`.

```
address allocate(infoType x) {
    address p = new elmlist;
    info(p) = x;
    next(p) = NULL;
    return p;
}
```

4. Fungsi `insertFirst(List &L, address P)`

   Fungsi ini menyisipkan elemen `P` ke awal linked list `L`. Pointer `next(P)` diset ke elemen pertama dari list sebelumnya, dan elemen pertama list kini menjadi `P`.

```
void insertFirst(List &L, address P) {
    next(P) = first(L);
    first(L) = P;
}
```

5. Fungsi `insertLast(List &L, address P)`

   Fungsi ini menyisipkan elemen `P` ke akhir list. Jika list kosong, elemen pertama langsung diisi dengan `P`. Jika tidak kosong, program akan mencari elemen terakhir dan menautkannya dengan elemen `P`.

```
void insertLast(List &L, address P) {
    if (first(L) == NULL) {
        first(L) = P;
    } else {
        address temp = first(L);
        while (next(temp) != NULL) {
            temp = next(temp);
        }
        next(temp) = P;
    }
}
```

6. Fungsi `insertAfter(List &L, address Prec, address P)`

   Fungsi ini menyisipkan elemen `P` setelah elemen `Prec` dalam list. Pertama, pointer `next(P)` diarahkan ke elemen setelah `Prec`, lalu `next(Prec)` diarahkan ke `P`.

```
void insertAfter(List &L, address Prec, address P) {
    if (Prec != NULL) {
        next(P) = next(Prec);
        next(Prec) = P;
    }
}
```

7. Fungsi `deleteLast(List &L)`

   Fungsi ini menghapus elemen terakhir dari linked list. Jika hanya ada satu elemen, maka elemen tersebut dihapus dan list menjadi kosong. Jika lebih dari satu elemen, fungsi mencari elemen kedua terakhir dan menghapus elemen terakhir.

```
void deleteLast(List &L) {
    if (first(L) != NULL) {
        address temp = first(L);
        if (next(temp) == NULL) {
            first(L) = NULL;
            delete temp;
        } else {
            address prev = NULL;
            while (next(temp) != NULL) {
                prev = temp;
                temp = next(temp);
            }
            next(prev) = NULL;
            delete temp;
        }
    }
}
```

8. Fungsi `deleteAfter(List &L, address Prec)`

   Fungsi ini menghapus elemen yang berada setelah elemen `Prec`. Jika `Prec` valid dan ada elemen setelahnya, elemen tersebut dihapus dan pointer `next(Prec)` di-update ke

elemen setelahnya.

```
void deleteAfter(List &L, address Prec) {
    if (Prec != NULL && next(Prec) != NULL) {
        address temp = next(Prec);
        next(Prec) = next(temp);
        delete temp;
    }
}
```

9. Fungsi `searchInfo(List L, infoType x)`

   Fungsi ini mencari elemen yang menyimpan informasi bernilai `x`. Fungsi mengembalikan `true` jika elemen ditemukan, dan `false` jika tidak.

```
bool searchInfo(List L, infoType x) {
    address temp = first(L);
    while (temp != NULL) {
        if (info(temp) == x) {
            return true;
        }
        temp = next(temp);
    }
    return false;
}
```

10. Fungsi `printInfo(List L)`

    Fungsi ini menampilkan semua elemen dari linked list, mulai dari elemen pertama hingga terakhir.

```
void printInfo(List L) {
    address p = first(L);
    while (p != NULL) {
        cout << info(p);
        p = next(p);
    }
    cout << endl;
}
```

11. Fungsi `main()`

    - Inisialisasi linked list `L` dengan `createList(L)`.

    - Menerima input sebanyak 10 digit dari pengguna dan menyimpannya dalam linkedlist menggunakan `insertLast()`.

    - Setelah semua digit dimasukkan, fungsi `printInfo()` akan menampilkan isi dari linked list.

```cpp
int main() {
    List L;
    createList(L);

    int totalDigits = 10;
    infoType nimDigit;

    cout << "Masukkan NIM perdigit:" << endl;

    for (int i = 1; i <= totalDigits; ++i) {
        cout << "Digit " << i << " : ";
        cin >> nimDigit;

        address P = allocate(nimDigit);
        insertLast(L, P);
    }

    cout << "Isi list: ";
    printInfo(L);

    return 0;
}
```

Berikut output:

```
Microsoft-MIEngine-In-mk3xzk15.ztr' '--stdout=Microsoft-MIEngine-Out-t2r15fdg.h5d' '--stderr=Microsoft-MIEngine-Err
-MIEngine-Pid-kvqayeiu.4vw' '--dbgExe=C:\msys64\ucrt64\bin\gdb.exe' '--interpreter=mi'
Masukkan NIM perdigit:
Digit 1 : 2
Digit 2 : 3
Digit 3 : 1
Digit 4 : 1
Digit 5 : 1
Digit 6 : 0
Digit 7 : 4
Digit 8 : 0
Digit 9 : 4
Digit 10 : 0
Isi list: 2311104040
PS C:\Users\lenovo>
```