Tugas 2 STD

Nama : Muhammad Rafli Ramadhan
Kelas : IF-44-04
NIM : 13012200204

SLL.h

```
#ifndef SLL_H_INCLUDED
#define SLL_H_INCLUDED
#include <iostream>
#include <string>

using namespace std;

struct dokter {
  string kode;
  string name;
};

typedef struct dokter Infotype;
typedef struct elmList *adr;

Struct elmList {
  Infotype info;
  adr      next;
};
```

```
struct List {
  adr First;
};

Void Create_List ( List *L);
adr New_Elemen(String kode, string name);
Void Insert_First (List *L, adr p);
Void Insert_Last ( List *L, adr p );
Void Insert_After (List *L, adr prec, adr p );
Void Delete_first (List *L, adr p );
Void Delete_Last ( List *L, adr p);
Void Delete_After(List *L, adr prec, adr p);
Void show ( List L);

#endif
```

SLL.cpp

```
#include "SLL.h"
Void create_List (List *L ) {
   L->First = Null;
}

adr New_Elemen (string kode, string nama ) {
  Cout << "Executing New_Elemen " << kode << " " << name << endl;
  adr p = new elmList;
  p-> info.kode = kode;
```

```
    P -> next = Null;
    return p;
}

Void Insert_First (List *L, adr p) {
    Cout << "Executing Insert_first" << endl << endl;
    p -> next = L -> First;
    L -> First = P;
    p = Null;
}

Void Insert_Last (List * L, adr P) {
    cout << "Executing Insert_Last" << endl << endl;
    If (L -> First != Null) {
        adr temp = L -> first;
        while (temp -> next != Null) {
            temp = temp -> next;
        }
        temp -> next = P;
        P = Null;
    }else {
        Insert_first (L, P);
    }
}

Void Insert_After (List *L, adr prec, adr p) {
    Cout << "Executing Insert_After" << endl;
    If (L -> first != Null) {
```

```
    adr temp = L -> first;
    while (temp -> info.kode != prec -> info.kode) {
        if (temp -> next == NULL) {
            return;
        }
        temp = temp -> next;
    }
    p -> next = temp -> next;
    temp -> next = p;
  }
}

Void Delete_first (List *L, adr p) {
    Cout << "Executing Delete_first" << endl;
    if (L -> First != Null) {
        p = L -> First;
        L -> first = p -> next;
        p -> next = Null;
        detele p;
    } else {
        cout << "Warning: Empty List" << endl;
    }
    Cout << endl;
}

Void Delete_last (List *L, adr p) {
    Cout << "Executing Delete_last" << endl;
    If (L -> First != Null) {
```

```
    p = L->first;
    if (p->next == NULL){
        Delete_Last (L,p);
    } else {
        adr temp = p->next;
        while (temp->next != NULL) {
            p = temp;
            temp = temp->next;
        }
        p->next = NULL;
        delete temp;
    }else{
        cout << "Warning : Empty List" << endl;
    }
    cout << endl;
}

Void Delete-After (List *L, adr prec, adr p) {
    cout << "Executing Delete-After" << endl;
    if(L->first != NULL) {
        adr temp = L->first;
        while (temp->info.kode != prec->info.kode){
            if(temp->next == NULL){
                return;
            }
```

```
            temp = temp -> next;
        }
        p = prec -> next
        prec -> next = p -> next;
        p -> next = NULL;
    }
}

Void Show ( List L) {
  Cout << "Executing show" <<endl;
  IF( L.First != Null) {
    adr p = L.First;
    while (p != Null) {
      Cout << p-> info.kode << " - " << p -> info.nama << endl;
      P = p-> next;
    }
  }
  Cout << endl;
}
```

main.CPP

```
int main() {
   List MMRA;
   Infotype data;

   Create_List(MMRA);

   adr p = New_elemen ("001", "Rahusien salim");
   Insert_first (&MMRA, P);
```

```
adr p = New_elemen ("002", "Muhammad fadil Maulana Akbar");
Insert_Last (&MMRA, P);

adr p = New_elemen ("003", "Muhammad Rafli Ramadhan");
Insert_First (&MMRA, P);

adr p = New_elemen ("004", "Shafa Diva Syahira");
Insert_Last (&MMRA, P);

Show (MMRA);

Delete_first (&MMRA);

Delete_last (&L, P);

Show (MMRA);

}
```