NAMA : ZAZKIA MAHARRANI ADZULVI

KELAS : IT-48-03

TUGAS PEKAN 2 : MEMBUAT ADT SINGLE LINKED LIST PART 1

Buatlah ADT SINGLE LINKED LIST yang akan menyimpan data sesuai dengan data yang Tim Anda dapatkan

1. Lakukan MOD 5 terhadap digit terkahir pada NIM ketua Tim Anda.
2. Sisa nya menentukan jenis data apa yang Anda simpan di list
    - Sisa 0 : Data yang berhubungan dengan data terkait sekolah/kampus
    - Sisa 1 : Data yang berhubungan dengan data terkait penjualan barang
    - Sisa 2 : Data yang berhubungan dengan data terkait kesehatan
    - Sisa 3 : Data yang berhubungan dengan data terkait pertandingan
    - Sisa 4: Data yang berhubungan dengan data terkait pekerjaan
3. Data tersebut minimal memiliki 3 sub data. Contoh : data terkait sekolah/ kampus, misal data mahasiswa (tidak boleh digunakan karena dijadikan contoh). Data mahasiswa terdiri dari NIM, NAMA, IPK

LENGKAPILAH ADT SINGLE LINKED LIST BERIKUT INI BESERTA MAIN PROGRAMNYA

A. Isi SLL.h
    a. Deklarasikan List Anda disini

```
Type infotype : <  idPasien : String
                   nama Pasien : String
                   Penyakit : String
                   >
Type infotype :  String
Type adr : ElemenList
Type elmList : < info : infotype
                   Next : adr>
Type List : <adr>
```

Sesuaikan dengan jenis data yang Anda dapatkan sesuai panduan di atas

    b. Tuliskan primitive single linked list based on fungsi procedure yang ada pada file sll.cpp pada poin B

B. Isi SLL.cpp
    a. **Procedure** Create_List (**In/Out** L : List)
    {I.S. –
    F.S. Pointer First dari List L bernilai NIL}
    Kamus data
    ……
    Algoritma
    L.first <- NIL
    END PROCEDURE

b. **Function** New_Elemen (data : infotype)→ adr
{Function akan membuat elemen baru berisi data dan mengembalikan pointer yang menyimpan alamat dari elemen tersebut}
Kamus data
P = adr
Algoritma
If P ≠ NULL THEN
  P.info ← data
  P.next ← NULL
ENDIF
RETURN P

c. **Procedure** Insert_First (**In/Out** L : List, **In** p : adr)
{I.S. Terdefinisi List L yang mungkin kosong dan sebuah elemen baru yang alamatnya disimpan oleh pointer P
F.S. Elemen baru telah masuk menjadi elemen paling pertama di list L}
Kamus data
…….
Algoritma
P.next <- L.firs
L.first <- P
END PROCEDURE

d. **Procedure** Insert_Last (**In/Out** L : List, **In** p : adr)
{I.S. Terdefinisi List L yang mungkin kosong dan sebuah elemen baru yang alamatnya disimpan oleh pointer P
F.S. Elemen baru telah masuk menjadi elemen paling akhir di list L}
Kamus data
Q : adr
Algoritma
if L.first = NIL THEN <
  L.first <- P
Else
  Q <- L.first
  While Q.next ≠ NIL DO
    Q <- Q.mext
  Endwhile
  Q.next <- P
Endif
END PROCEDURE
>

e. **Procedure** Insert_After (**In/Out** L : List, **In** prec, p : adr)
{I.S. Terdefinisi List L yang mungkin kosong, sebuah elemen baru yang alamatnya disimpan oleh pointer P dan pointer prec
F.S. Elemen baru telah masuk menjadi elemen setelah elemen yang alamatnay disimpan oleh pointer prec }

Kamus data

……

Algoritma

P.next <- prec.next

Prec.next <- P

END PROCEDURE

f. **Procedure** Delete_First (**In/Out** L : List, **Out** p : adr)

{I.S. Terdefinisi List L yang mungkin kosong atau Cuma memiliki 1 elemen. F.S. Jika list kosong, maka pointer P di assign dengan NIL, jika tidak kosong maka elemen paling awal dihapus dari List L, alamatnya disimpan oleh pointer p}

Kamus data

…….

Algoritma

```
If L.first ≠ NIL THEN
   P <- NIL
Else
   P <- L.first
   L.first <- P.next
   P.next <- NIL
Endif
END PROCEDURE
```

g. **Procedure** Delete_Last (**In/Out** L : List, **Out** p : adr)

{I.S. Terdefinisi List L yang mungkin kosong atau Cuma memiliki 1 elemen. F.S. Jika list kosong, maka pointer P di assign dengan NIL, jika tidak kosong maka elemen paling akhir dihapus dari List L, alamatnya disimpan oleh pointer p}

Kamus data

Q : adr

Algoritma

```
If L.first ≠ NIL THEN
   P <- NIL
Else If L.first -> next = NIL THEN
  P <- L.first
  L.first <- NIL
Else
  Q <- L.first
  While Q.next .next ≠ NIL DO
     Q <- Q.next
  Endwhile
  P <- Q.next
  Q .next <- NIL
Endif
END PROCEDURE
```

h. **Procedure** Delete_After (**In/Out** L : List, **In** prec : adr,**Out** p : adr)

{I.S. Terdefinisi List L yang mungkin kosong, sebuah elemen baru yang alamatnya disimpan oleh pointer P dan pointer prec
F.S. Elemen baru telah masuk menjadi elemen setelah elemen yang alamatnay disimpan oleh pointer prec }

Kamus data

Q : adr

Algoritma

P <- prec.next
Prec.next <- P.next
P.next <- NIL
END PROCEDURE

i. **Procedure** Show (**In** L : List)

{I.S. Terdefinisi List L yang mungkin kosong.
F.S. Jika list kosong maka tampilkan ke layar "list kosong", jika tidak maka seluruh data pada list ditampilkan ke layar}

Kamus data

Q : adr
I : integer

Algoritma

```
If L.first = NIL THEN
  Output("List kosong")
Else
  Q <- L.first
  I <- 1
 While Q ≠ NIL DO
     Output("Data Pasien ke-", i)
     Output("ID Pasien : ", Q -> info.idPasien)
     Output("Nama Pasien : ", Q -> info.namaPasien)
     Output("Penyakit : ", Q -> info.penyakit)
     Output("---------------------------")
    Q <- Q.next
    i <- i + 1
Endwhile
```

C. Isi Main.cpp

{Lengkapilah titik-titik berikut ini agar pada main program :

1. *Deklarasi variable List*
2. *Mengcreate List*
3. *Memasukkan 4 data ke list dengan memanfaatkan dua jenis insert*
4. *Memanggil show data*
5. *Menghapus 2 data ke list dengan memanfaatkan dua jenis delete}*

//deklarasi variable List. ***Nama variable List adalah huruf pertama pada nama lengkap Anda dan huruf terakhir nama lengkap anggota tim Anda***

Deklarasi

R : List {nama list dari huruf pertama nama ketua dan huruf terakhir nama anggota, contoh : R dari Zazkia dan i dari Adzulvi}

P : adr {Pointer elemen baru}

//meng create list
Create_List(R)

*//meminta user **data pertama** yang akan dimasukkan ke list*
Output("Masukkan ID Pasien : ")
Input(data,idPasien)
Output("Masukkan Nama Pasien : ")
Input(data.namaPasien)
Output("Masukkan Penyakit : ")
Input(data.namaPasien)

//melakukan create new elemen berdasarkan data yang diinputkan user
P <- New_Elemen(data)

//memanggil salah satu jenis insert (jangan insert after)
Insert_First(R, P)

*//meminta user **data kedua** yang akan dimasukkan ke list*
Output("Masukkan ID Pasien : ")
Input(data,idPasien)
Output("Masukkan Nama Pasien : ")
Input(data.namaPasien)
Output("Masukkan Penyakit : ")
Input(data.penyakit)

//melakukan create new elemen berdasarkan data yang diinputkan user
P <- New_Elemen(data)

//memanggil salah satu jenis insert (berbeda dengan insert yang data pertama, jangan Insert after)
Insert_Last(R, P)

*//meminta user **data ketiga** yang akan dimasukkan ke list*
Output("Masukkan ID Pasien : ")

Input(data.idPasien)

Output("Masukkan Nama Pasien : ")

Input(data.namaPasien)

Output("Masukkan Penyakit : ")

Input(data.penyakit)

//melakukan create new elemen berdasarkan data yang diinputkan user
P ← New_Elemen(data)

//memanggil salah satu jenis insert (insert First)

Insert_First(R,P)

*//meminta user **data keempat** yang akan dimasukkan ke list*
Output("Masukkan ID Pasien : ")

Input(data.idPasien)

Output("Masukkan Nama Pasien : ")

Input(data.namaPasien)

Output("Masukkan Penyakit : ")

Input(data.penyakit)

//melakukan create new elemen berdasarkan data yang diinputkan user
P ← New_Elemen(data)

//memanggil salah satu jenis insert (insert Last)
Insert_Last(R, P)

//memanggil show
Show(R)

//melakukan penghapusan data dengan memanfaatkan salah satu jenis delete (bukan delete after)
Delete_First(R, P)

//melakukan penghapusan data dengan memanfaatkan salah satu jenis delete (berbede dengan delete sebelumnya, dan bukan delete after)
Delete_Last(R, P)