PERTEMUAN 9:
LINEAR SINGLY LINKED LIST

A. TUJUAN PEMBELAJARAN

Pada bab ini akan dijelaskan mengenai penggunaan Linear Singly Linked List yang terdapat pada struktur data. Di modul ini, Anda harus mampu:

9.1 Merepresentasikan Linear Singly Linked List dalam bahasa pemrograman .

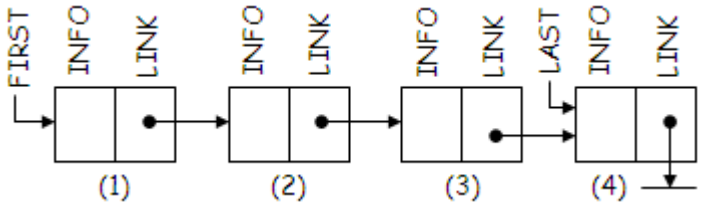
B. URAIAN MATERI

Tujuan Pembelajaran 9.1:

Aplikasi Linear Singly Linked List

Link List: sejumlah obyek yang dilink/dihubungkan satu dengan lainnya.
Obyek : gabungan bebrapaelemen data yg dijadikan satu kelompok/struktur/record
Untuk menghubungkan antar obyek perlu variabel tipe pointer yg merupakan salah satu variabel dalam struktur obyek.
Linear Singly Linked List : Link list lurus dengan pointer tunggal

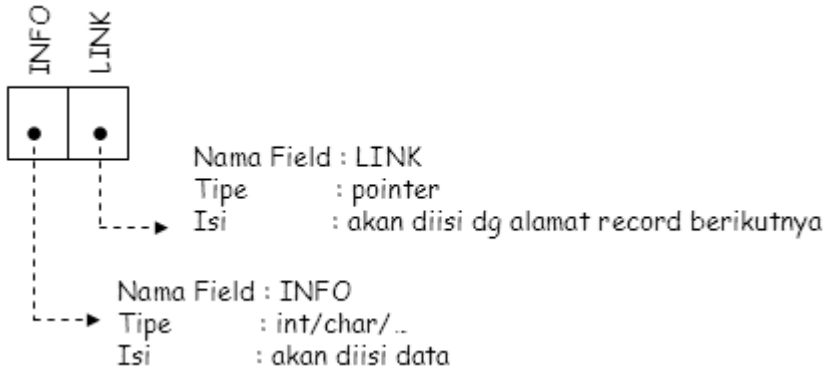
I. Ilustrasi



- Ada 4 simpul : 1, 2, 3, 4
- Setiap simpul terdiri dari 2 elemen/field, yaitu :
  - INFO : bertipe integer
  - LINK : bertipe pointer
- Simpul no.1 :

  - Field INFO berisi 10
  - Field LINK berisi alamat simpul no. 2
- Simpul no.1 ditunjuk oleh pointer FIRST
- Simpul no.4 ditunjuk oleh pointer LAST

Ilustrasi sebuah simpul :



Untuk mempersiapkan sebuah linked list maka harus dideklarasikan sbb :

```
struct SIMPUL{
        int INFO;
        struct SIMPUL *LINK;
};
SIMPUL *P,*Q,*FIRST,*LAST;
```

II. Proses

a. Inisialisasi : persiapan pembuatan linked list
b. Membuat simpul awal
c. Insert simpul kedalam linked list
d. Delete simpul dari linked list

II.1. Inisialisasi

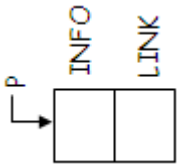
```
FIRST = NULL;
LAST = NULL;
```

II.2. Pembuatan simpul

Instruksi untuk membuat sebuah simpul :

```
P=(SIMPUL*) malloc(sizeof(SIMPUL));
```

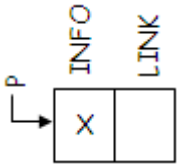
Akan terbentuk sebuah simpul yang alamatnya tersimpan dalam pointer P. Ilustrasi :



Fungsi untuk membuat simpul :

```
void BUAT_SIMPUL(int X)
{
    P=(SIMPUL*) malloc(sizeof(SIMPUL));
    if(P!=NULL)
    {
        P->INFO=X;
    }
    else
        cout<<"Pembuatan simpul gagal";
}
```

Ilustrasi :



Contoh :

```
#include<iostream.h>
#include<stdlib.h>
struct SIMPUL{
        int INFO;
        struct SIMPUL *LINK;
};
SIMPUL *P,*FIRST,*LAST;
void BUAT_SIMPUL(int);
void main(void)
{
```

```
        int x;
        cout<<"Masukan Data : ";cin>>x;
        BUAT_SIMPUL(x);
        cout<<"Data : "<<P->INFO<<endl;
}
void BUAT_SIMPUL(int x)
{
        P=(SIMPUL *)malloc(sizeof(SIMPUL));
        if(P!=NULL)
                P->INFO=x;
        else
                cout<<"Pembuatan Simpul Gagal"<<endl;
}
```

II. 3. Pembuatan simpul awal

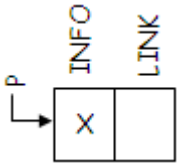
Menjadikan sebuah simpul menjadi simpul awal dari sebuah linked list. Simpul awal ditunjuk oleh pointer FIRST.

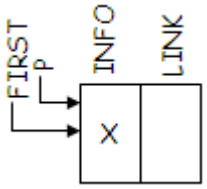
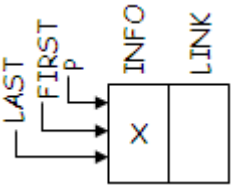
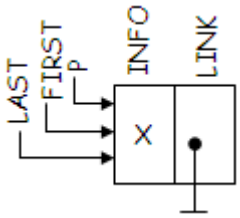
Fungsi :

```
Void AWAL(void)
{
   if(FIRST==NULL)
   {
      FIRST=P;
      LAST=P;
      P->LINK=NULL;
   }
   else
      cout<<"Linked List sudah ada""<<endl;
}
```

Ilustrasi :

Sudah dibuat simpul sbb:



FIRST=P	
LAST=P atau LAST=FIRST	
P->LINK=NULL atau FIRST->LINK=NULL atau LAST->LINK=NULL	

II.4. Insert Kanan

Menyisipkan sebuah simpul baru pada ujung kanan linked list.

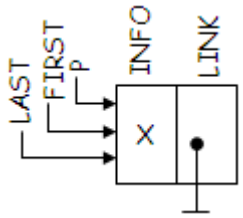
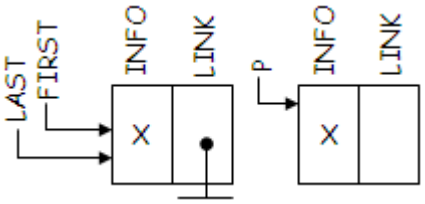
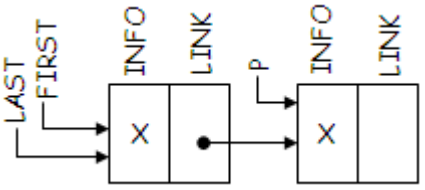
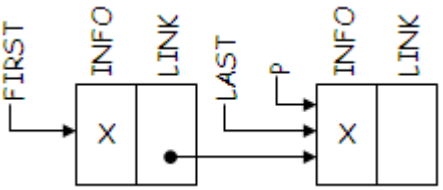
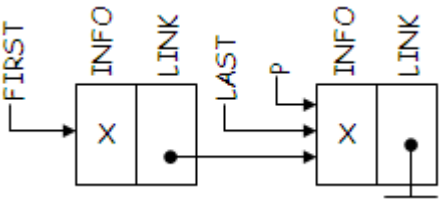
Proses :

- sudah ada linked list
- buat simpul baru
- sisipkan simpul baru tsb diujung kanan linked list

Fungsi :

```
void INSERT_KANAN(void)
{
    if(LAST!=NULL)
    {
        LAST->LINK=P;
        LAST=P;
        P->LINK=NULL;
    }
    else
        cout<<"Linked List belum ada";
}
```

Ilustrasi :

Sudah ada linked list	
Buat simpul baru P=(SIMPUL *)malloc(sizeof(SIMPUL));	
LAST->LINK=P atau FIRST->LINK=P	
LAST=P atau LAST=FIRST->LINK	
P->LINK=NULL atau LAST->LINK=NULL atau FIRST->LINK->LINK=NULL	

II.5. Insert Kiri

Menyisipkan sebuah simpul baru pada ujung kiri linked list.

Proses :

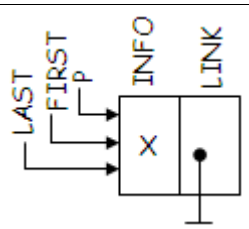
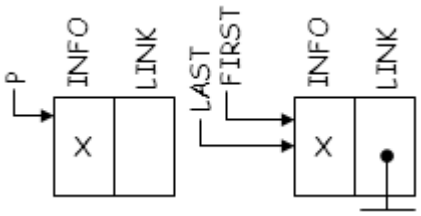
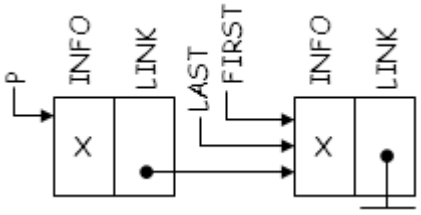
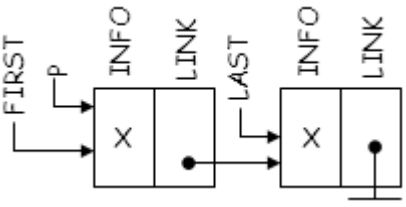
- sudah ada linked list
- buat simpul baru
- sisipkan simpul baru tsb diujung kiri linked list

Fungsi :

```
void INSERT_KIRI(void)
{
    if(FIRST!=NULL)
```

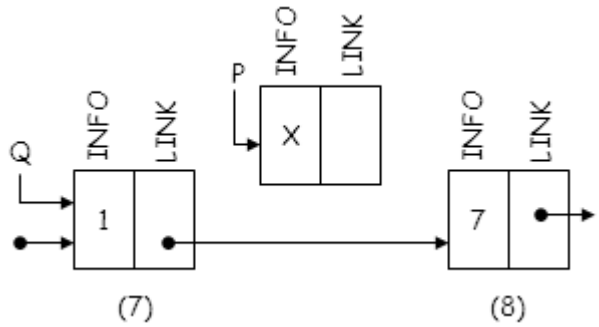
```
    {
        P->LINK=FIRST;
        FIRST=P;
    }
    else
        cout<<"Linked List belum ada";
}
```

Ilustrasi:

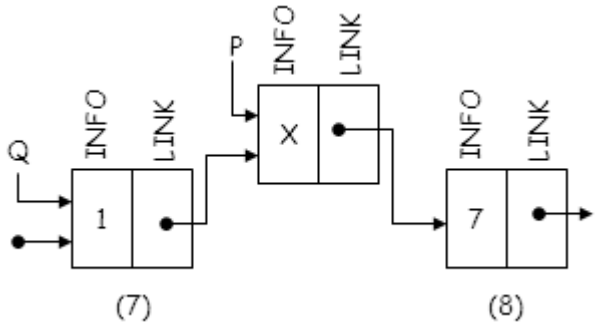
Sudah ada linked list	
Buat simpul baru P=(SIMPUL *)malloc(sizeof(SIMPUL));	
P->LINK=FIRST atau P->LINK=LAST	
FIRST=P	

II.6. Insert Tengah

Menyisipkan sebuah simpul antara dua buah simpul pada linked list.



setelah diinsert menjadi :



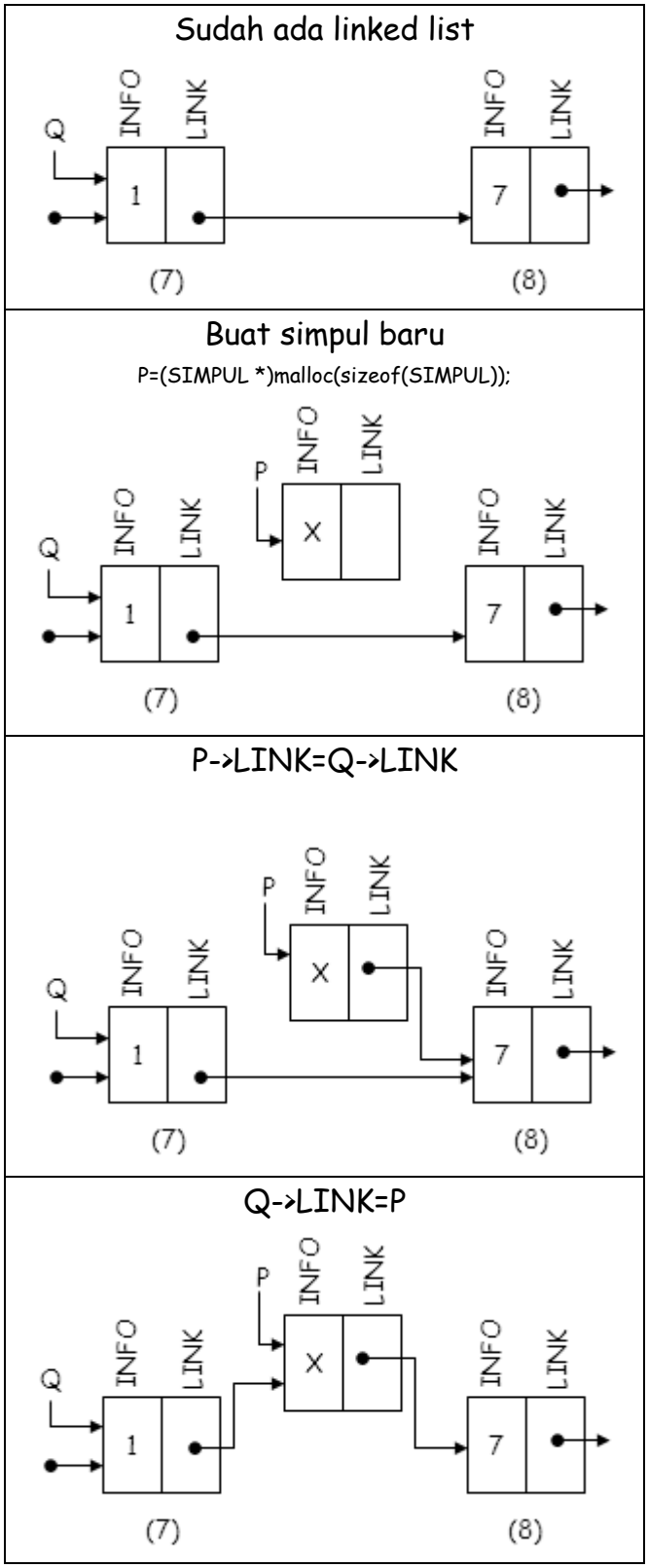
Syarat : simpul no.7 harus sudah ditunjuk oleh pointer Q, caranya :

```
Q=FIRST;
For(i=1;i<=6;i++)
        Q=Q->LINK;
```

Fungsi :

```
Void INSERT_TENGAH(void)
{
    P->LINK=Q->LINK;
    Q->LINK=P;
}
```

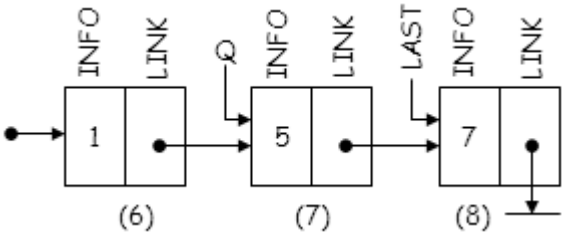
Ilustrasi :


Sudah ada linked list
Q
INFO
LINK
1
7
(7)
(8)
Buat simpul baru
P=(SIMPUL *)malloc(sizeof(SIMPUL));
P
X
P->LINK=Q->LINK
Q->LINK=P

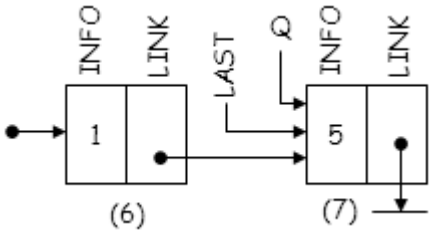
II.7. Delete Kanan/Akhir

Menghapus simpul yang ada pada linked list paling akhir/kanan.

Ilustrasi : sudah ada sebuah linked list



akan dihapus simpul terakhir menjadi :



Syarat agar simpul no.8 dapat dihapus adalah simpulno.7 sudah ditunjuk oleh pointer Q.

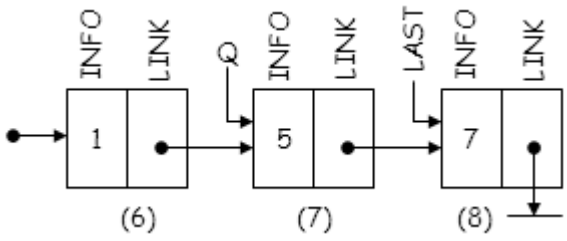
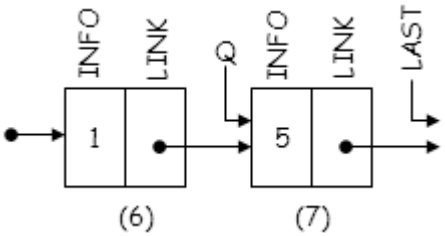
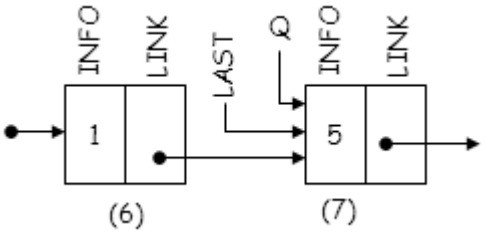
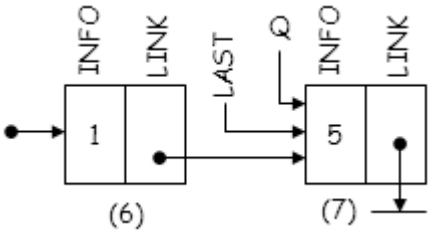
Caranya :

```
Q = FIRST;
while(Q ->LINK != LAST)
        Q = Q -> LINK;
```

Fungsi :

```
void DELETE_KANAN(void)
{
    free(LAST);
    LAST = Q;
    LAST -> LINK = NULL;
}
```

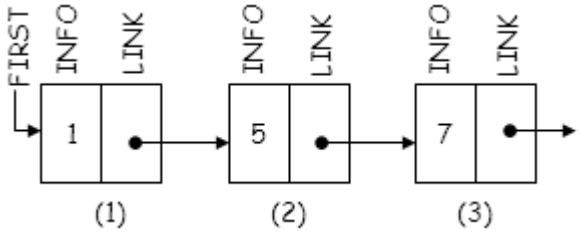
Ilustrasi :

Sudah ada linked list	
free(LAST)	
LAST = Q	
LAST -> LINK = NULL	

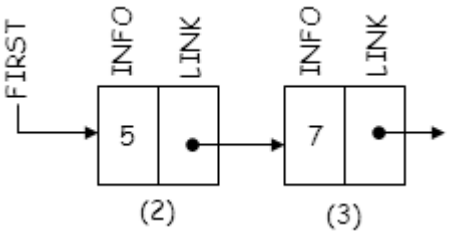
II.8. Delete Kiri/Awal

Menghapus simpul yang ada pada linked list paling awal/kiri.

Ilustrasi : sudah ada sebuah linked list



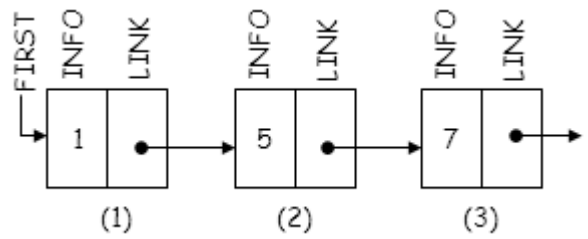
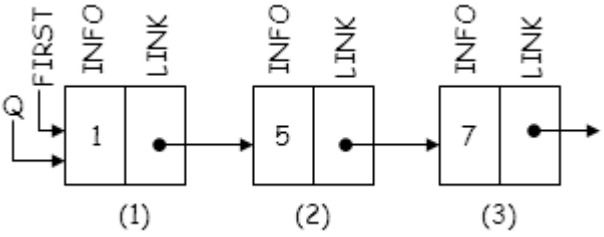
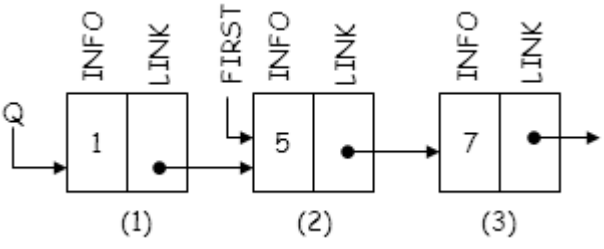
akan dihapus simpul awal menjadi :

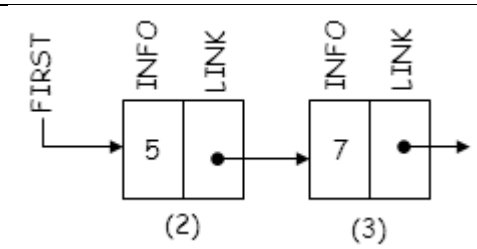


Fungsi :

```
void DELETE_KIRI(void)
{
    Q = FIRST;
    FIRST = Q -> LINK;
    free(Q);
}
```

Ilustrasi :

Sudah ada linked list	
Q = FIRST	
FIRST = Q -> LINK	

free(Q)	

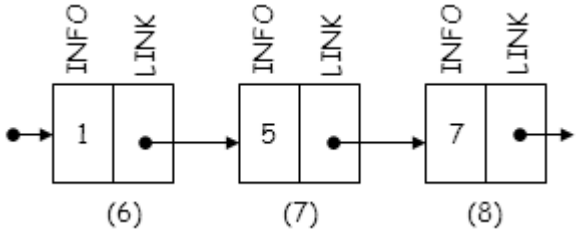
Tuliskan cara lain!

Tip : Tempatkan Q pada simpul kedua, hapus simpul pertama, pindahkan FIRST ke simpul kedua.

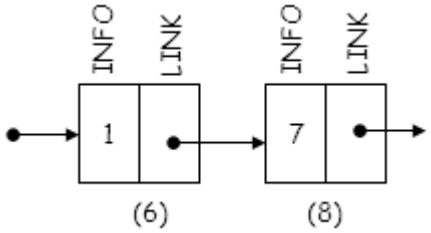
II.9. Delete Tengah

Menghapus simpul yang ada diantara dua simpul lain.

Ilustrasi : sudah ada linked list



simpul no.7 akan dihapus sehingga menjadi :



Syarat agar simpul no.7 bisa dihapus maka simpul no.6 harus sudah ditunjukoleh Q.

Caranya :

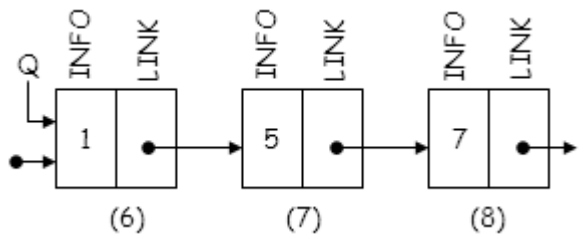
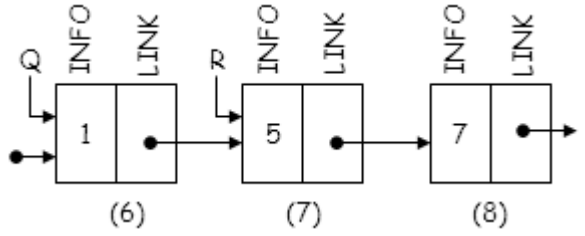
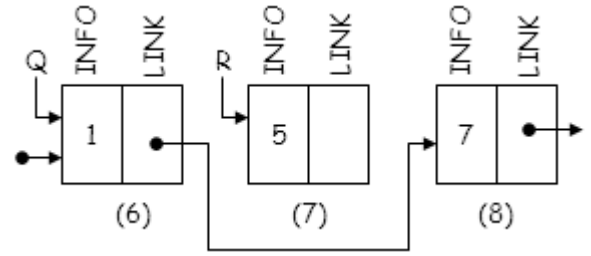
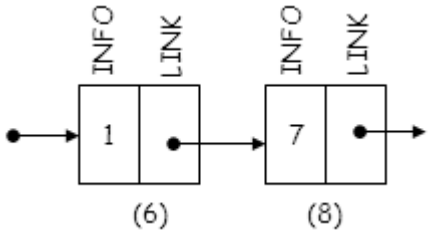
```
Q = FIRST;
For(I = 1; I <= 5; I++)
        Q = Q -> LINK;
```

Fungsi :

```
void DELETE_TENGAH(void)
{
    R = Q -> LINK;
```

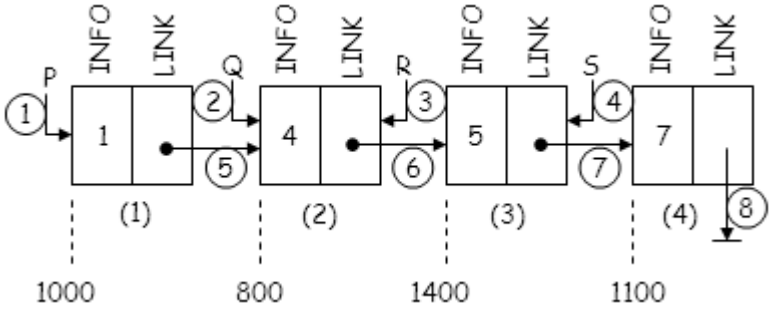
```
    Q -> LINK = R -> LINK;
    free(R);
}
```

Ilustrasi :

Sudah ada linked list	
R = Q -> LINK	
Q -> LINK = R -> LINK	
free(R)	

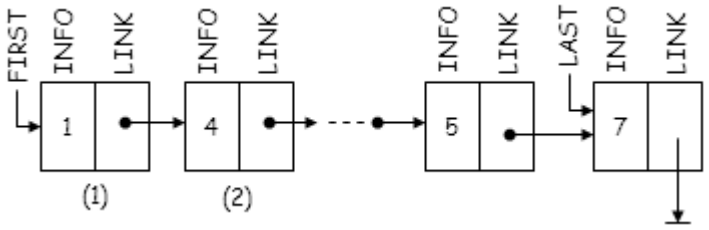
Latihan :

1. Perhatikan linked list berikut :



a) Sebutkan nama dan isi tiap-tiap pointer
b) Sebutkan pointer-pointer yang isinya sama
c) Sebutkan TRUE atau FALSE kondisi tiap pernyataan dibawah ini :
    i. if(P->LINK==R)
    ii. if(Q->LINK==R->LINK)
    iii. if(Q->LINK->LINK==S->LINK)
    iv. if(Q==R)
    v. if(Q-LINK==R)
    vi. if(R->LINK->INFO==5)
    vii. if(Q->INFO==4)

2. Perhatikan linked list berikut :



Jumlah simpul lebih dari 20 buah

Susunlah pernyataan bahasa C untuk:

a) Menempatkan pointer Q sehingga menunjuk simpul nomor 1
b) Menempatkan pointer Q sehingga menunjuk simpul nomor 7
c) Menempatkan pointer Q sehingga menunjuk simpul nomor terakhir
d) Menenmpatkan pointer Q sehingga menunjuk simpul dengan INFO = 50
e) Menghapus simpul yang berisi INFO = 40

TUGAS

Buat program animasi Linear Singly Linked List untuk mengelola data mahasiswa dengan struktur mahasiswa sbb : NAMA, NIM, GENDER, NILAI STRUKTUR DATA. Program dibuat dalam bentuk menu dengan pilihan : INSERT DATA, HAPUS DATA, CETAK DATA, EXIT.

Ket :

INSER DATA : menyisipkan satu simpul pada akhir linked list

HAPUS DATA :menghapus satu simpul pada akhir linked list

CETAK DATA : mencetak seluruh isi linked list

EXIT : Keluar/selesai

Tampilan menu :

```
LIN. SINGLY LINKED LIST
=======================
1. INSERT DATA
2. HAPUS DATA
3. CETAK DATA
4. EXIT
Pilihan (1 – 4) :
```

C. DAFTAR PUSTAKA

Buku

1. Esakov, Jeffrey, Tom Weiss, Data Structures An Advanced Approach Using C, Prentice-Hall, Inc. 1989
2. Hariyanto, Bambang, Struktur Data, Informatika Bandung, Pebruari 2000
3. Kadir, Abdul, Pemrograman Dasar Turbo C, Andi Offset, Yogyakarta, 1991
4. Kruse, Robert L. Data Structures & Program Design, Prentice-Hall, Inc. 1987
5. Standish, Thomas A. Data Structures, Algorithms & Software Principles In C, Addison Wesley, 1995