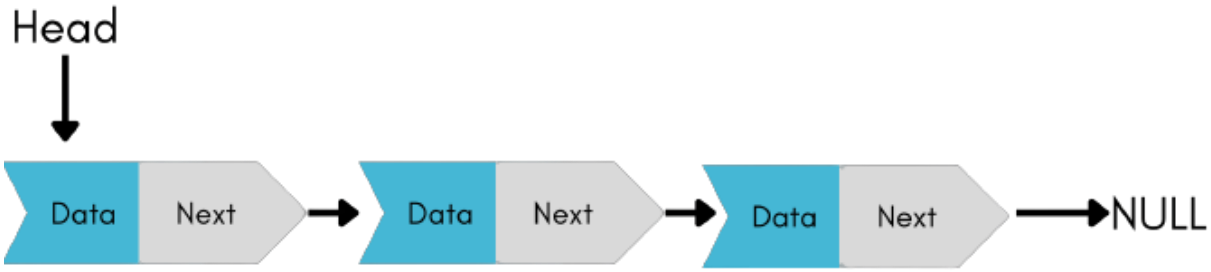
Modul 7

Linked List 2

Tujuan

1. Praktikan dapat semakin memahami dan menguasai konsep linked list.
2. Praktikan dapat menggunakan pemahaman atas konsep linked list untuk mengembangkan linked list.

Dasar Teori



Pada dasarnya, sebuah linked list beserta dengan data-datanya akan tersimpan secara dinamis dalam memory komputer karena menggunakan konsep malloc. Untuk mengakses data tersebut, linked list menggunakan pointer yang menunjuk ke tempat penyimpanan data tersebut. Sebuah elemen di dalam linked list terdiri dari data yang disimpan pada alamat elemen tersebut dan juga alamat untuk data pada elemen selanjutnya di dalam linked list.

Elemen dalam linked list akan berisi 2 tipe data, yaitu datanya sendiri sesuai dengan kebutuhan dan alamat elemen selanjutnya pada linked list. Untuk memanggil elemen pertama, linked list akan menyimpan alamat dari elemen pertama dengan pointer pada bagian head. Untuk mengakses ke data selanjutnya, akan digunakan pemanggilan pointer Next yang menunjuk kepada elemen selanjutnya. Jika sudah berada pada elemen yang terakhir, pointer Next pada elemen tersebut akan menjunjuk ke NULL.

Operasi-Operasi pada Linked List

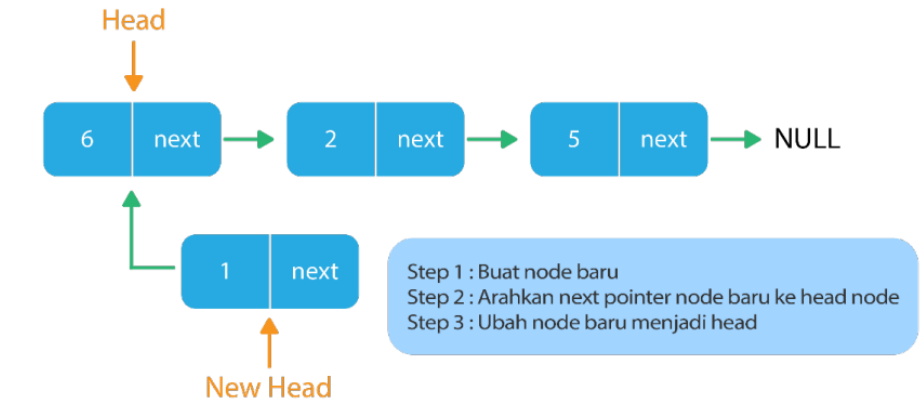
2(dua) dari operasi yang paling penting pada linked list adalah insert dan delete. Insert ditujukan untuk memasukkan data pada linked list, dan delete bertujuan untuk menghapus data dari linked list.

Insert

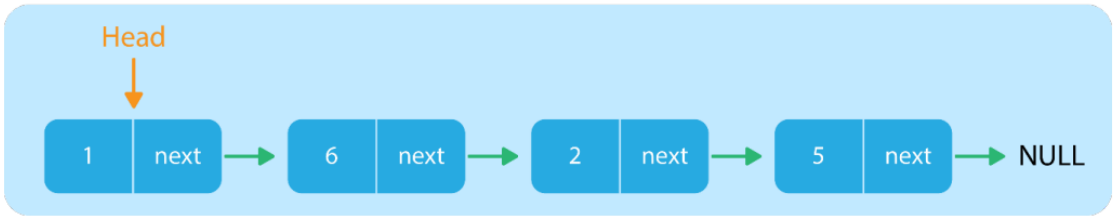
Beberapa operasi insert pada linked list adalah sebagai berikut:

1. InsertFirst
2. InsertLast
3. InsertAfter

InsertFirst
(Memasukkan data sebagai elemen pertama)



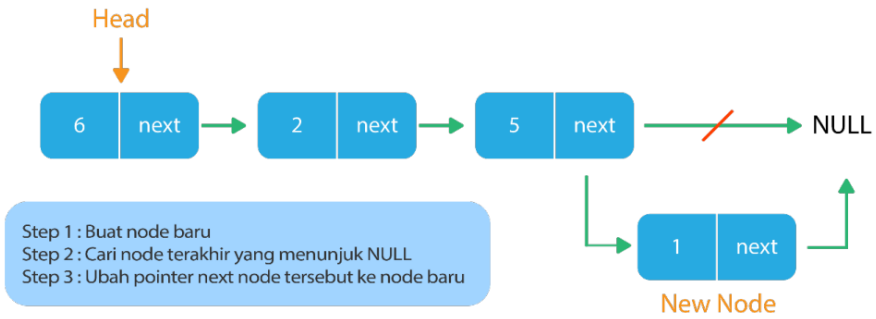
Result :

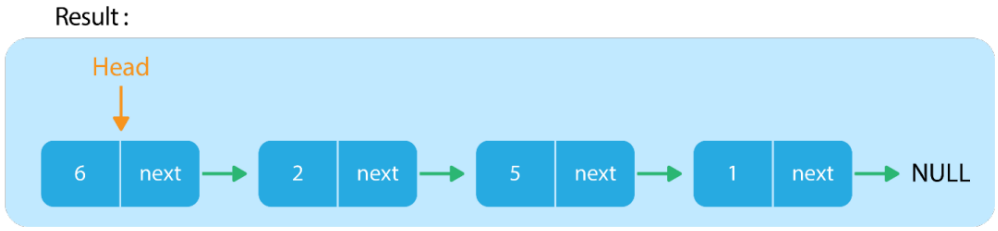


Insertfirst adalah operasi yang berfungsi untuk memasukkan elemen baru menjadi elemen pertama pada linked list. Jika linked list masih kosong, pointer pada head akan terhubung dengan alamat hasil alokasi elemen terbaru. Jika linked list sudah ada isi sebelumnya, maka pointer pada elemen baru akan terlebih dahulu dihubungkan ke data paling awal yang sebelumnya sudah ada, lalu pointer pada head akan dihubungkan dengan alamat hasil alokasi elemen terbaru.

```
void insertFirst(List *L, infoType X)
{   address newNode = alokasi(X);
    newNode->next = (*L).first;
    (*L).first = newNode;
}
```

InsertLast
(Memasukkan data sebagai elemen terakhir)

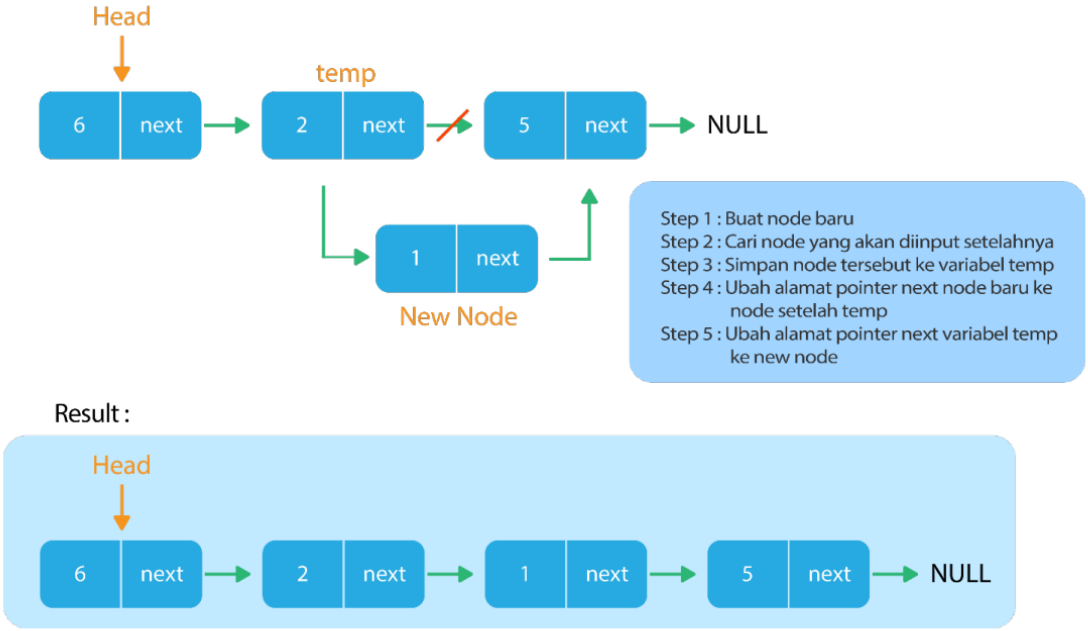




InserLast adalah operasi yang berfungsi untuk memasukkan elemen baru menjadi elemen terakhir pada linked list. Untuk memasukkan elemen menjadi elemen terakhir, perlu terlebih dahulu dicari data yang terakhir pada linked list. Setelah mencari elemen terakhir, pointer pada elemen terakhir tersebut akan terhubung dengan elemen yang terbaru. Jika belum ada elemen sama sekali di linked list, data di masukkan pada elemen pertama linked list seperti insertfirst.

```
void insertLast(List *L, infoType X){
    address P,newNode;
    newNode = alokasi(X);
    if(isEmpty((*L))){
        insertFirst(L,newNode);
    }else{
        for(P=(*L).first;P->next !=NULL;P=P-
>next); P->next = newNode;
    }
}
```

InsertAfter
(Memasukkan data setelah data tertentu)



InserAfter adalah operasi untuk memasukkan elemen baru pada linked list setelah elemen tertentu. Untuk melakukan insertAfter, cari dulu elemen yang akan dimasukkan elemen baru setelahnya. Alokasikan elemen baru, dan hubungkan alamat next dari elemen sebelumnya ke alamat next dari temp, dan masukkan alamat next elemen sebelumnya menjadi alamat temp.

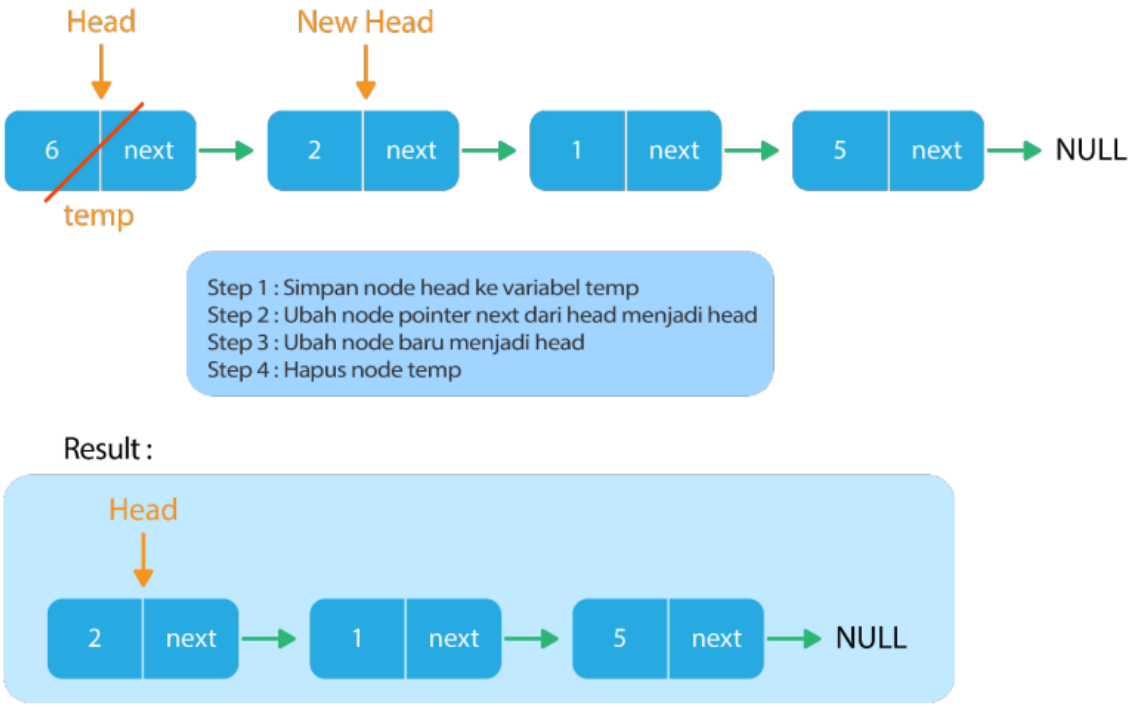
```
void insertAfter(address before, infoType X)
{   newNode = alokasi(X);
    if(before!=NULL){
        newNode->next  = before->next;
        before->next = newNode;
    }
}
```

Delete

Beberapa operasi delete pada linked list adalah sebagai berikut:

1. DeleteFirst
2. DeleteLast
3. DeleteAt

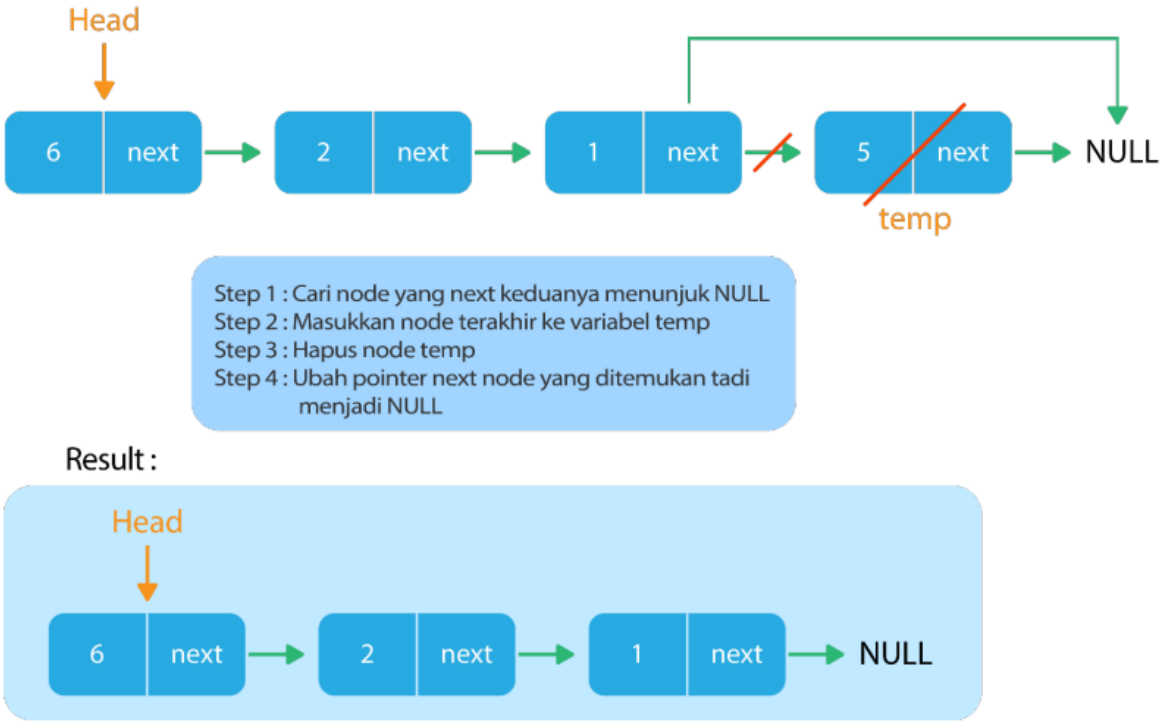
DeleteFirst
(menghapus elemen pertama)



DeleteFirst adalah operasi yang digunakan untuk menghapus data pertama pada linkedList. Operasi ini akan melakukan penghapusan dengan terlebih dahulu menghubungkan linkedList pada elemen pertamanya kepada elemen setelah elemen pertama, atau elemen kedua. Lalu dilakukan pembebasan pada elemen pertama.

```
void deleteFirst(List *L){
    if(!isEmpty((*L))){
        address del = (*L).first;
        (*L).first = (*L).first-
>next;  free(del);
    }
}
```

DeleteLast
(Menghapus elemen terakhir)

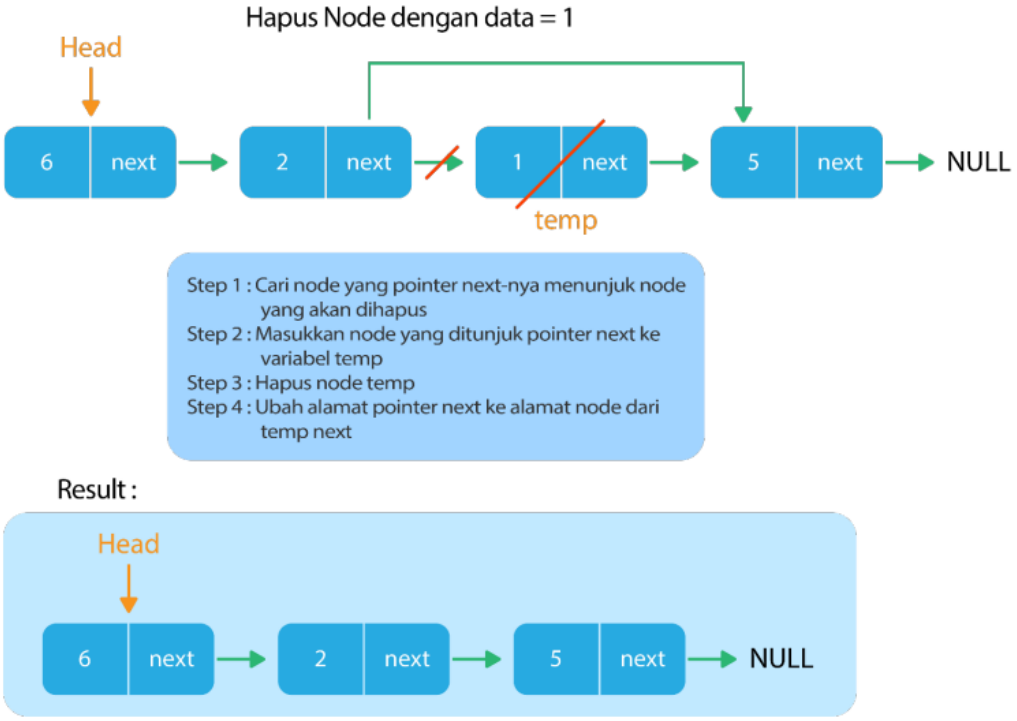


DeleteLast adalah operasi untuk menghapus data terakhir dari linked list. Cara menghapusnya adalah dengan mencari elemen sebelum elemen terakhir, lalu membebebaskan elemen terakhir. Elemen sebelum elemen terakhir tersebut akan dihubungkan ke null karena sudah tidak ada lagi data di elemen next/selanjutnya dari elemen tersebut.

```
void deleteLast(List *L){
    address P;

    if(!isEmpty((*L))){
        if(isOneElement((*L))){
            deleteFirst(L);
        }else{
            for(P = (*L).first;P->next->next != NULL;P=P->next);
            free(P->next);
            P->next = NULL;
        }
    }
}
```

DeleteAt
(menghapus elemen tertentu)



Operasi DeleteAt akan menghapus data pada elemen tertentu. Untuk melakukan penghapusan ini, perlu dicari terlebih dahulu elemen sebelum elemen yang ingin dihapus. Elemen tersebut yang berada sebelum elemen yang ingin dihapus akan dihubungkan ke elemen selanjutnya dari elemen yang dihapus. Lalu hapus elemen yang ingin dihapus. Untu melakukan operasi ini, perlu digunakan operasi FindNode(Bisa dilihat di guided) untuk mencari data yang ingin dihapus terlebih dahulu.

```
void deleteAt(List *L, infoType X){
    address P, del;
    del = findNode((*L),X);
    if(del != null){
      if((*L).first == del){
          deleteFirst(L);
      }else{
          P=(*L).first;
          while(P->next !=  del){
              P = P->next;
          }
          P->next = del->next;
          free(del);
      }
    }
}
```

GUIDED

header.h

```c
#include <stdio.h>
#include <stdlib.h>
#include <stdbool.h>
#include <conio.h>

//tipe data yang akan diteirma oleh linked list adalah infotype, yaitu int
typedef int infotype;

/*address digunakan untuk menyimpan alamat pointer dari node*/
typedef struct node *address;

//node akan menyimpan data dan juga menunjuk ke node berikutnya yang berisi data berikutnya
typedef struct node{
    infotype data;
    address next;
}node;
//list menunjuk ke data pertama, yaitu first
typedef struct{
    address first;
}List;

void createEmpty(List *L);                          //buat list kosong
bool isEmpty(List L);                               //cek apakah list kosong
bool isOneElement(List L);                          //cek apakah list hanya 1 elemen
address alokasi(infotype X);                        //mengalokasikan data ke sebuah address, membuat node
void insertFirst(List *L, address newNode);         //memasukkan data ke awal linked list
void insertAfter(address before, address newNode);  //memasukkan data ke linked list setelah node tertentu
void insertLast(List *L,address newNode);           //memasukkan data ke akhir linked list
void deleteFirst(List *L);                          //menghapus node pertama
void deleteAt(List *L, address del);                //menghapus node tertentu
void deleteLast(List *L);                           //menghapus node terakhir
void printData(List L);                             //menampilkan data pada linkedlist
address findNode(List L,infotype X);                //mencari data pada linkedList menggunakan tipe data tertentu
```

Masih ada lanjutannya di bawah ya, ini baru header…

source.c

```c
#include "header.h"

void createEmpty(List *L){
    (*L).first = NULL;
}

bool isEmpty(List L){
    return L.first == NULL;
}

bool isOneElement(List L){
    return !isEmpty(L) && L.first->next == NULL;
}

address alokasi(infotype X){
    address p;
    p = (node*)malloc(sizeof(node));
    p->data = X;
    p->next = NULL;
    return p;
}

void insertFirst(List *L, address newNode){
    newNode->next = (*L).first;
    (*L).first = newNode;
}


void insertAfter(address before, address newNode){
    if(before!=NULL){
        newNode->next  = before->next;
        before->next = newNode;
    }
}

void insertLast(List *L, address newNode){
    address P;
    if(isEmpty((*L))){
        insertFirst(L,newNode);
    }else{
        for(P=(*L).first;P->next !=NULL;P=P->next);
        P->next = newNode;
    }
}

void deleteFirst(List *L){
    if(!isEmpty((*L))){
        address del = (*L).first;
        (*L).first = (*L).first->next;
        free(del);
    }
}

void deleteAt(List *L, address del){
    address P;
    if(!isEmpty((*L))){
        if((*L).first == del){
            deleteFirst(L);
        }else{
            P=(*L).first;
            while(P->next !=  del){
                P = P->next;
            }
            P->next = del->next;
            free(del);
        }
    }
}
```

source.c (Lanjutan)

```
void deleteLast(List *L){
    address P;

    if(!isEmpty((*L))){
        if(isOneElement((*L))){
            deleteFirst(L);
        }else{
            for(P = (*L).first;P->next->next != NULL;P=P->next);
            free(P->next);
            P->next = NULL;
        }
    }
}

void printData(List L){
    address P;
    printf("\n\t");
    for(P=L.first;P!=NULL;P=P->next)
    printf("%d ,",P->data);
}

address findNode(List L, infotype X){
    address P;
    for(P=L.first;P!=NULL && P->data != X;P=P->next);
    return P;
}
```

main.c

```
#include "header.h"


int main(int argc, char *argv[]) {
    List L;
    address temp,before;
    int bil;
    char menu;

    createEmpty(&L);

    do{
        system("cls");
        //tampil data yang terisi dalam linkedlist
        if(!isEmpty(L)){
            printData(L);
        }else{
            printf("\n\t Data Masih Kosong");
        }
        printf("\n\n\t Guided Linked List 2\n");
        printf("\n\t 1. Insert First");
        printf("\n\t 2. Insert After");
        printf("\n\t 3. Insert Last");
        printf("\n\t 4. Delete First");
        printf("\n\t 5. Delete At");
        printf("\n\t 6. Delete Last");
        printf("\n\t 0. EXIT");
        printf("\n Input Menu: \t");menu= getch();
        printf("\n\t\n");
        switch(menu){
            case '1':
                printf("\n\t Enter Number\t: ");scanf("%d",&bil);
                /*alokasi bisa dilakukan di luar seperti ini, atau juga bisa dilakukan di dalam file source.c
                  dengan menganti tipe data pada parameter menjadi infotype/tipe data lain,
                  dan mengalokasi data sebelum melakukan proses insertfirst atau proses insert lainnya*/
                temp = alokasi(bil);
                printf("%d",bil);
                insertFirst(&L,temp);
                printf("\n\t Inserted successfully");
                break;
```

main.c(Lanjutan)

```
            case '2':
                printf("\n\tInsert Number after : ");scanf("%d",&bil);
                /*lihat findNode di source, yang berfungsi untuk mencari data sesuai inputan pengguna
                before merupakan data yang kita ingin masukkan setelahnya*/
                before = findNode(L,bil);
                if(before == NULL){
                    printf("\n\t[!] 404 Number Not Found");
                    break;
                }
                printf("\n\t Enter Number\t: ");scanf("%d",&bil);
                temp = alokasi(bil);
                insertAfter(before,temp);
                printf("\n\t Inserted successfully");
                break;

            case '3':
                printf("\n\t Enter number\t: ");scanf("%d",&bil);
                temp = alokasi(bil);
                insertLast(&L,temp);
                printf("\n\t Inserted successfully");\
                break;

            case '4':
                if(isEmpty(L)){
                    printf("\n\t[!] List Kosong!");
                    break;
                }

                deleteFirst(&L);
                printf("\n\t Delete Successfully");
                break;

            case '5':
                if(isEmpty(L)){
                    printf("\n\t [!] List Is Empty");
                    break;
                }
                printf("\n\tNumber you want to delete\t:");scanf("%d",&bil);
                temp = findNode(L,bil);
                //karena deleteAt pada guided ini menggunakan address, maka dicari dulu address nya menggunakan findNode
                if(temp == NULL){
                    printf("\n\t[!] 404 Number Not Found");
                    break;
                }
                deleteAt(&L, temp);
                printf("\n\t Deleted successfully");
                break;

            case '6':
                if(isEmpty(L)){
                    printf("\n\t [!] List is empty");
                    break;
                }
                deleteLast(&L);
                printf("\n\t Deleted Successfully");
                break;


            case '0':
                //XXXX ganti jadi 4 digit NPM, A ganti jadi Kelas, NAMAPRAKTIKAN ganti jadi nama kalian
                printf("\n\t Exit [XXXX - A - NAMAPRAKTIKAN]");
                break;

            default :
                printf("\n\t[!] Menu Doesn't Exist");
                break;

        }getch();
    }while(menu!= '0');

    return 0;
}
```

KETENTUAN PENGUMPULAN

Kumpulkan maksimal **H-1** SEBELUM PRAKTIKUM

Format pengumpulan:

GD7_X_YYYYY

X = kelas

Y = 5 digit NPM

Tips:

Pelajari cara memindah-mindahkan elemen dalam linked list.