**Struktur type**

**Merah: tambahan untuk list ganda tanpa last, list ganda circular**

**Merah+Biru: tambahan untuk list ganda dengan last**

#define Nil NULL

#define info(P) (P)->info

#define next(P) (P)->next

#define prev(P) (P)->prev

#define First(L) (L).First

#define Last(L) (L).Last

typedef enum{false,true} boolean;

typedef int infotype;

typedef struct tElmtList \*address;

typedef struct tElmtList {

infotype info;

address next;

address prev;

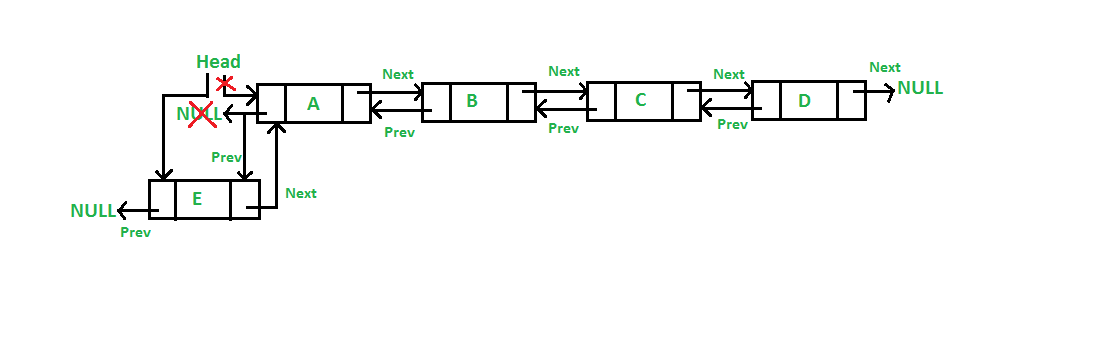
}ElmtList;

typedef struct {

address First;

address Last;

}List;



**Procedure** InsertFirst (input/output: L: List, input :P: address)

{I.S. Sembarang, P sudah dialokasi}

{F.S. Menambahkan elemen ber-address P sebagai elemen pertama}

{Kamus Lokal}

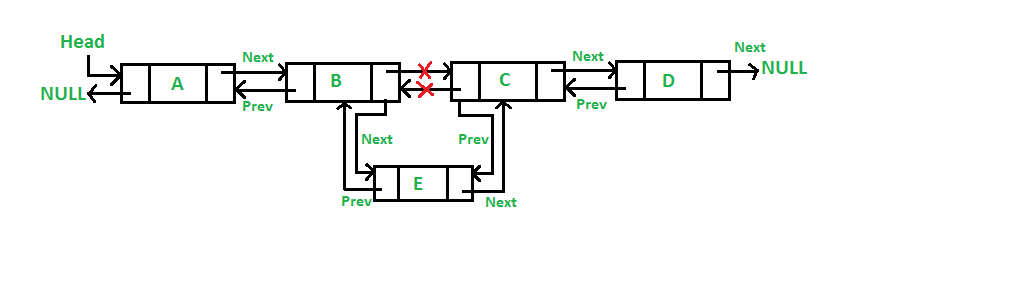
{Algoritma}

Next(P)🡨First(L)

Prev(First(L))🡨P

First(L)🡨P

Prev(P)🡨Nil



**Procedure** InsertAfter (input/output: L: List, input :P: address, input :Prec: address)

{I.S. Prec pastilah elemen list dan bukan elemen terakhir, P sudah dialokasi}

{F.S. Insert P sebagai elemen sesudah elemen beralamat Prec }

{Kamus Lokal}

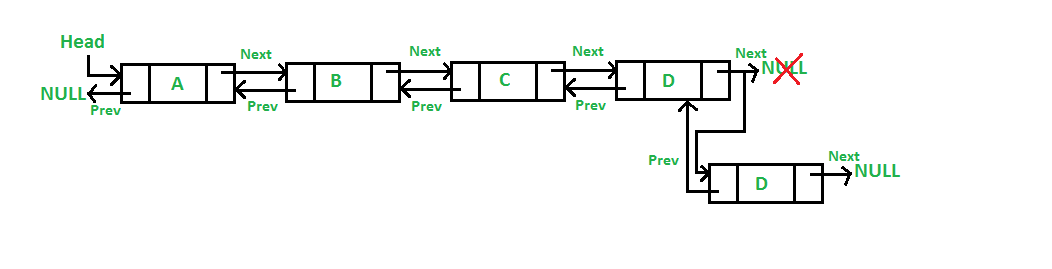
{Algoritma}

Next(P)🡨Next(Prec)

Next(Prec)🡨P

Prev(P)🡨 Prec

Prev(Next(Prec))🡨P



**Procedure** InsertLast (input/output: L: List, input: P: address)

{ I.S. Sembarang, P sudah dialokasi }

{ F.S. P ditambahkan sebagai elemen terakhir yang baru }

{Kamus}

Last: address

{Algoritma}

Next(P)🡨Nil

If(ListEmpty(L)) then {list kosong}

Prev(P)🡨Nil

First(L)🡨P

else

While (Next(Last)!=Nil) do

Last🡨Next(Last)

Next(Last)🡨P

Prev(P)🡨Last

**Buatlah Algoritma (seperti contoh diatas)**

**Untuk spesifikasi setiap primitif mengambil dari List Linier**

1. **List ganda (tanpa Last**)

DelFirst

DelAfter

DelLast

1. **List ganda (dengan Last)**

InsertFirst

InsertAfter

InsertLast

DelFirst

DelAfter

DelLast

1. **List ganda circular (tanpa Last)**

InsertFirst

InsertAfter

InsertLast

DelFirst

DelAfter

DelLast

Jawaban

1. **List ganda (tanpa Last)**

#define Nil NULL

#define info(P) (P)->info

#define next(P) (P)->next

#define prev(P) (P)->prev

#define First(L) (L).First

typedef enum{false,true} boolean;

typedef int infotype;

typedef struct tElmtList \*address;

typedef struct tElmtList {

infotype info;

address next;

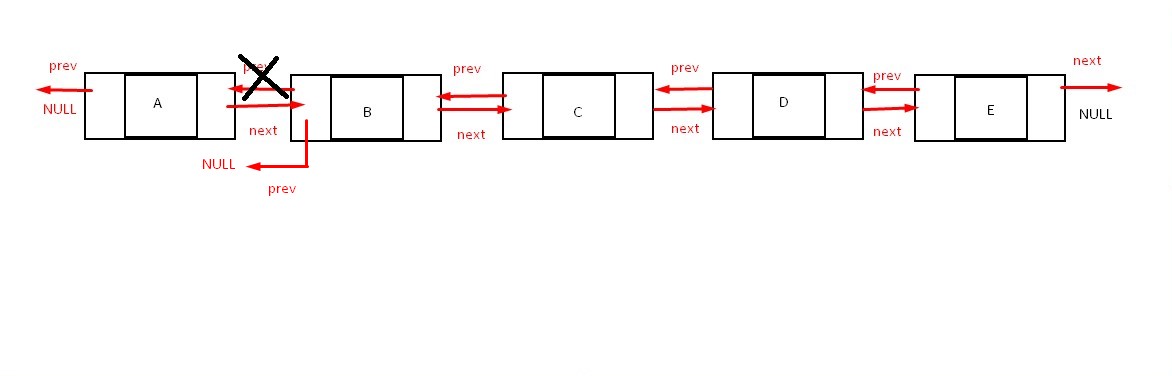
address prev;

}ElmtList;

typedef struct {

address First;

}List;



**Procedure** DelFirst (input/output: L: List, output :P: address)

{I.S. List tidak kosong}

{F.S. P adalah alamat elemen pertama list sebelum penghapusan. Elemen list berkurang satu (mungkin menjadi kosong). First element yg baru adalah suksesor elemen pertama yang lama}

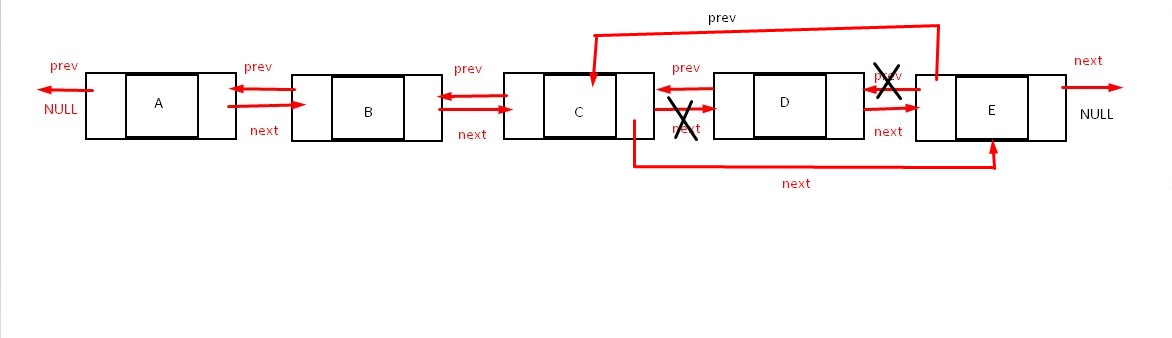
{Kamus Lokal}

{Algoritma}

P🡨First(L)

First(L)🡨Next(First(L))

Prev(First(L))🡨Nil



**Procedure** DelAfter (input/output: L: List, output :Pdel: address, input :Prec: address)

{I.S. List tidak kosong. Prec adalah anggota list}

{F.S. Menghapus Next(Prec): Pdel adalah alamat elemen list yang dihapus}

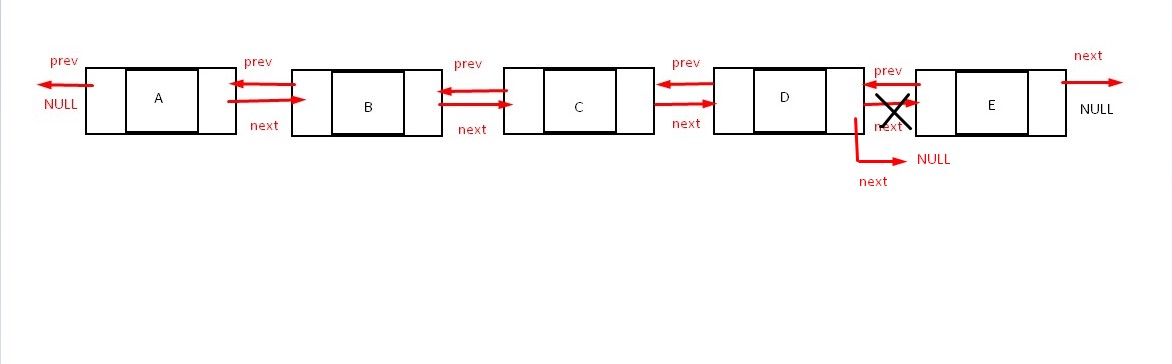
{Kamus Lokal}

{Algoritma}

Pdel🡨Next(Prec)

Next(Prec)🡨Next(Next(Prec))

Prev(Next(Prec))🡨Prec



**Procedure** DelLast (input/output: L: List, output: P: address)

{ I.S. List tidak kosong}

{ F.S. P adalah alamat elemen terakhir list sebelum penghapusan. Elemen list berkurang satu (mungkin menjadi kosong). Last element baru adalah predesesor elemen pertama yg lama, jika ada}

{Kamus}

PrecLast: address

Last: address

{Algoritma}

Last🡨First(L)

PrecLast🡨Nil

While (Next(Last)!=Nil) do

PrecLast🡨Last

Last🡨Next(Last)

P🡨Last

If(PrecLast = Nil) then {Last masih First(L)}

First(L)🡨Nil

else

Next(PrecLast)🡨Nil

1. **List ganda (dengan Last)**

#define Nil NULL

#define info(P) (P)->info

#define next(P) (P)->next

#define prev(P) (P)->prev

#define First(L) (L).First

#define Last(L) (L).Last

typedef enum{false,true} boolean;

typedef int infotype;

typedef struct tElmtList \*address;

typedef struct tElmtList {

infotype info;

address next;

address prev;

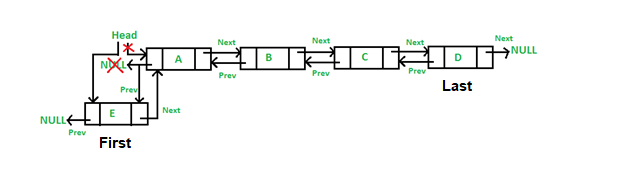
}ElmtList;

typedef struct {

address First;

address Last;

}List;



**Procedure** InsertFirst (input/output: L: List, input :P: address)

{I.S. Sembarang, P sudah dialokasi}

{F.S. Menambahkan elemen ber-address P sebagai elemen pertama}

{Kamus Lokal}

{Algoritma}

Prev(P)🡨Nil

If(ListEmpty(L)) then {list kosong}

First(L)🡨P

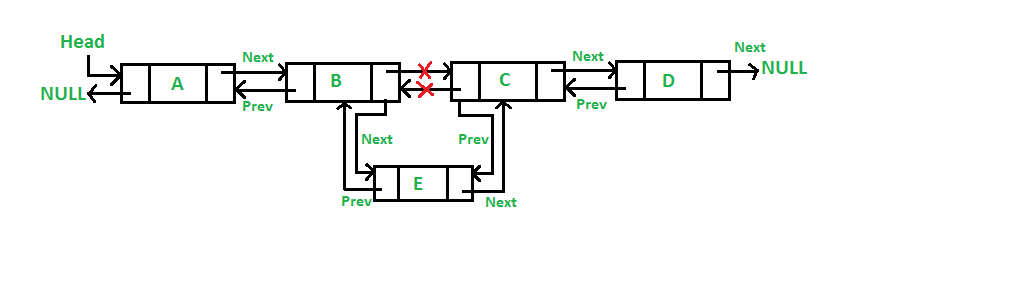
Last(L)🡨P

else

Prev(First(L))🡨P

Next(P)🡨First(L)

First(L)🡨P



Last

First

**Procedure** InsertAfter (input/output: L: List, input :P: address, input :Prec: address)

{I.S. Prec pastilah elemen list dan bukan elemen terakhir, P sudah dialokasi}

{F.S. Insert P sebagai elemen sesudah elemen beralamat Prec }

{Kamus Lokal}

{Algoritma}

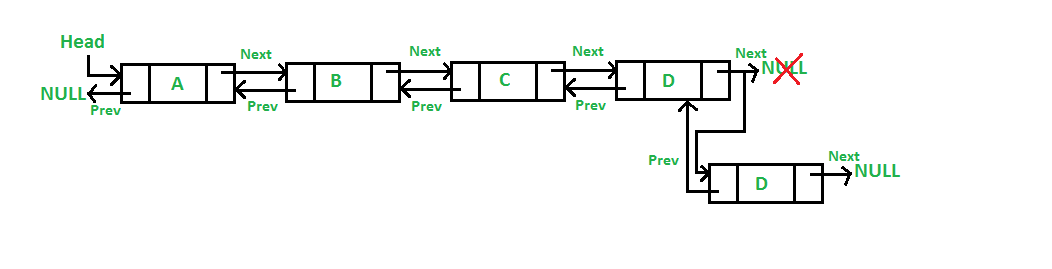
Prev(P)🡨Prec

Prev(Next(Prec))🡨P

Next(P)🡨Next(Prec)

Next(Prec)🡨P

Last



~~Last~~

First

**Procedure** InsertLast (input/output: L: List, input: P: address)

{ I.S. Sembarang, P sudah dialokasi }

{ F.S. P ditambahkan sebagai elemen terakhir yang baru }

{Kamus}

{Algoritma}

Next(P)🡨Nil

If(ListEmpty(L)) then {list kosong}

Prev(P)🡨Nil

First(L)🡨P

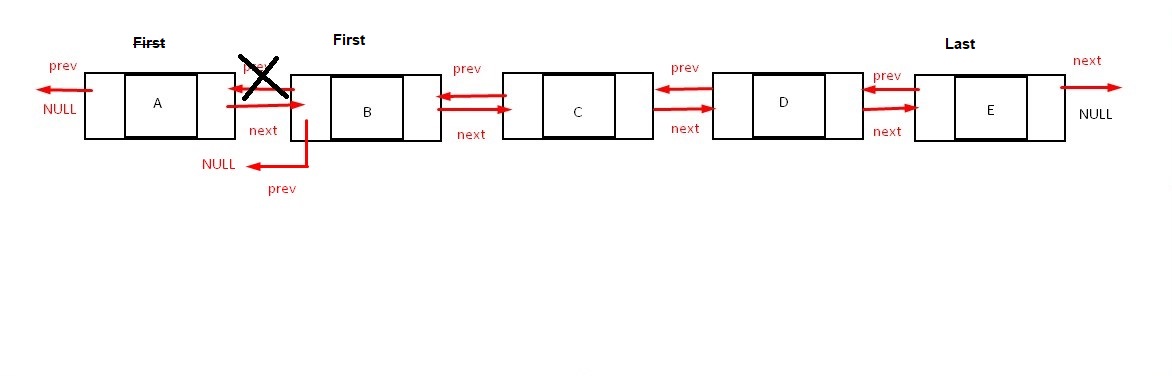
Last(L)🡨P

else

Prev(P)🡨Last(L)

Next(Last(L))🡨P

Last(L)🡨P



**Procedure** DelFirst (input/output: L: List, output :P: address)

{I.S. List tidak kosong}

{F.S. P adalah alamat elemen pertama list sebelum penghapusan. Elemen list berkurang satu (mungkin menjadi kosong). First element yg baru adalah suksesor elemen pertama yang lama}

{Kamus Lokal}

{Algoritma}

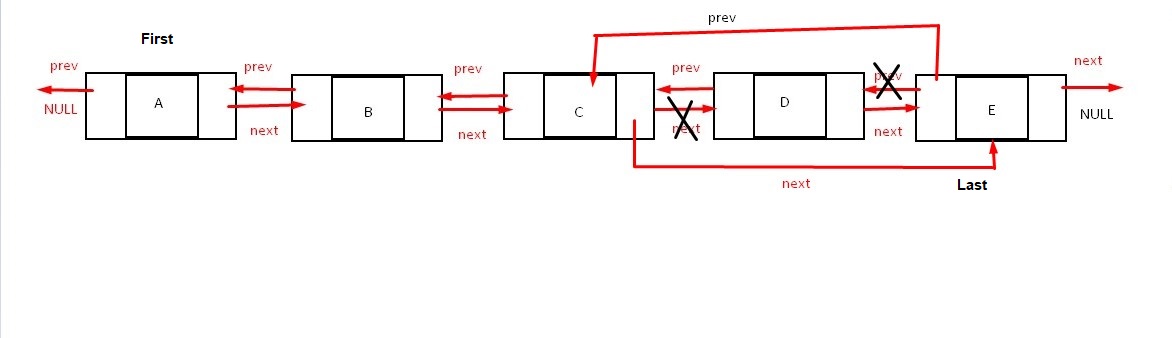
P🡨First(L)

First(L)🡨Next(First(L))

Prev(First(L))🡨Nil

If(First(L) = Nil) then {Satu elemen}

Last(L)🡨Nil



**Procedure** DelAfter (input/output: L: List, output :Pdel: address, input :Prec: address)

{I.S. List tidak kosong. Prec adalah anggota list}

{F.S. Menghapus Next(Prec): Pdel adalah alamat elemen list yang dihapus}

{Kamus Lokal}

{Algoritma}

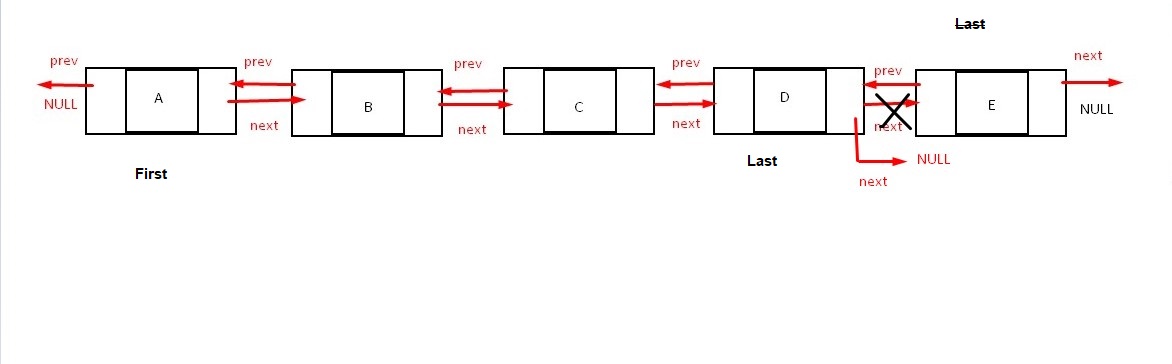
Pdel🡨Next(Prec)

If(Pdel = Last(L)) then {Pdel adalah Last(L)}

Last(L)🡨Prev(Last(L))

Next(Prec)🡨Next(Next(Prec))

Prev(Next(Prec))🡨Prec



**Procedure** DelLast (input/output: L: List, output: P: address)

{ I.S. List tidak kosong}

{ F.S. P adalah alamat elemen terakhir list sebelum penghapusan. Elemen list berkurang satu (mungkin menjadi kosong). Last element baru adalah predesesor elemen pertama yg lama, jika ada}

{Kamus}

Last: address

{Algoritma}

P🡨Last(L)

If(P = First(L)) then {Last(L) sama dengan First(L)}

Last(L)🡨Nil

First(L)🡨Nil

else

Last(L)🡨Prev(Last(L))

Next(Last(L))🡨Nil

1. List ganda circular (tanpa Last)

#define Nil NULL

#define info(P) (P)->info

#define next(P) (P)->next

#define prev(P) (P)->prev

#define First(L) (L).First

typedef enum{false,true} boolean;

typedef int infotype;

typedef struct tElmtList \*address;

typedef struct tElmtList {

infotype info;

address next;

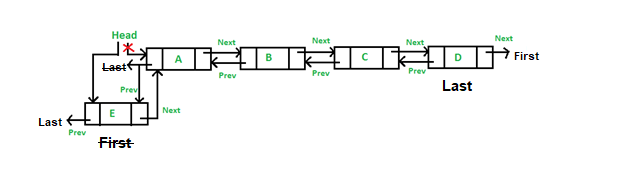
address prev;

}ElmtList;

typedef struct {

address First;

}List;



**Procedure** InsertFirst (input/output: L: List, input :P: address)

{I.S. Sembarang, P sudah dialokasi}

{F.S. Menambahkan elemen ber-address P sebagai elemen pertama}

{Kamus Lokal}

{Algoritma}

If(ListEmpty(L)) then {list kosong}

Prev(P)🡨P

Next(P)🡨P

First(L)🡨P

else

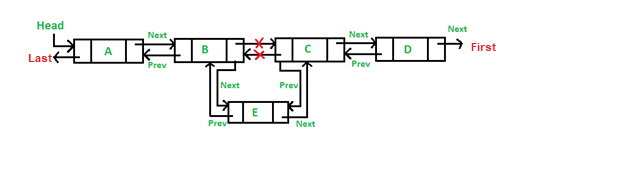
Prev(P)🡨Prev(First(L))

Next(P)🡨First(L)

Next(Prev(First(L)))🡨P

Prev(First(L))🡨P

First(L)🡨P



First

Last

**Procedure** InsertAfter (input/output: L: List, input :P: address, input :Prec: address)

{I.S. Prec pastilah elemen list dan bukan elemen terakhir, P sudah dialokasi}

{F.S. Insert P sebagai elemen sesudah elemen beralamat Prec }

{Kamus Lokal}

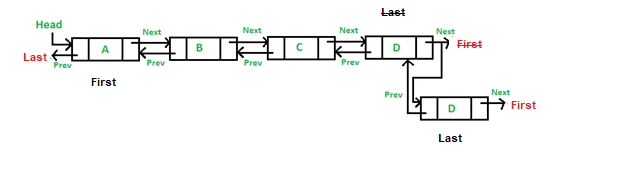
{Algoritma}

Prev(P)🡨Prec

Next(P)🡨Next(Prec)

Prev(Next(Prec))🡨P

Next(Prec)🡨P



**Procedure** InsertLast (input/output: L: List, input: P: address)

{ I.S. Sembarang, P sudah dialokasi }

{ F.S. P ditambahkan sebagai elemen terakhir yang baru }

{Kamus}

{Algoritma}

If(ListEmpty(L)) then {list kosong}

Prev(P)🡨P

Next(P)🡨P

First(L)🡨P

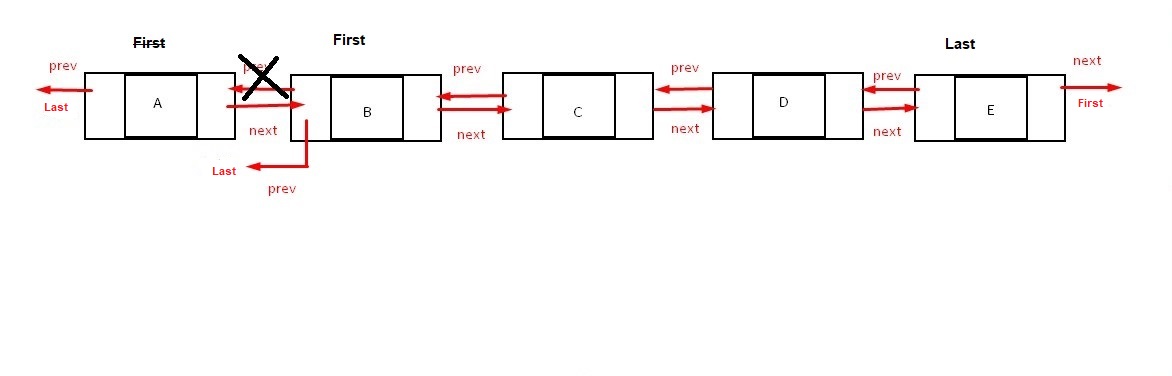
else

Prev(P)🡨Prev(First(L))

Next(P)🡨First(L)

Next(Prev(First(L)))🡨P

Prev(First(L))🡨P

****

**Procedure** DelFirst (input/output: L: List, output :P: address)

{I.S. List tidak kosong}

{F.S. P adalah alamat elemen pertama list sebelum penghapusan. Elemen list berkurang satu (mungkin menjadi kosong). First element yg baru adalah suksesor elemen pertama yang lama}

{Kamus Lokal}

{Algoritma}

P🡨First(L)

If(Next(P)=P) then {Satu elemen}

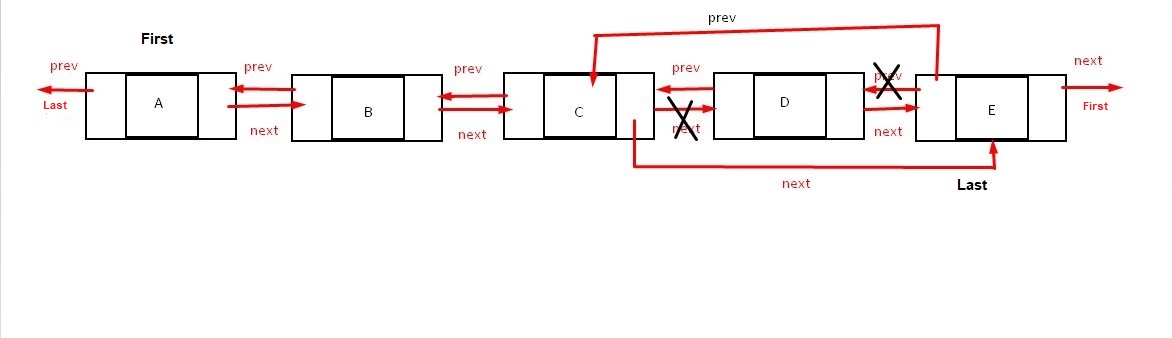
First(L)🡨Nil

else

Next(Prev(First(L)))🡨Next(First(L))

Prev(Next(First(L)))🡨Prev(First(L))

First(L)🡨Next(First(L))



**Procedure** DelAfter (input/output: L: List, output :Pdel: address, input :Prec: address)

{I.S. List tidak kosong. Prec adalah anggota list}

{F.S. Menghapus Next(Prec): Pdel adalah alamat elemen list yang dihapus}

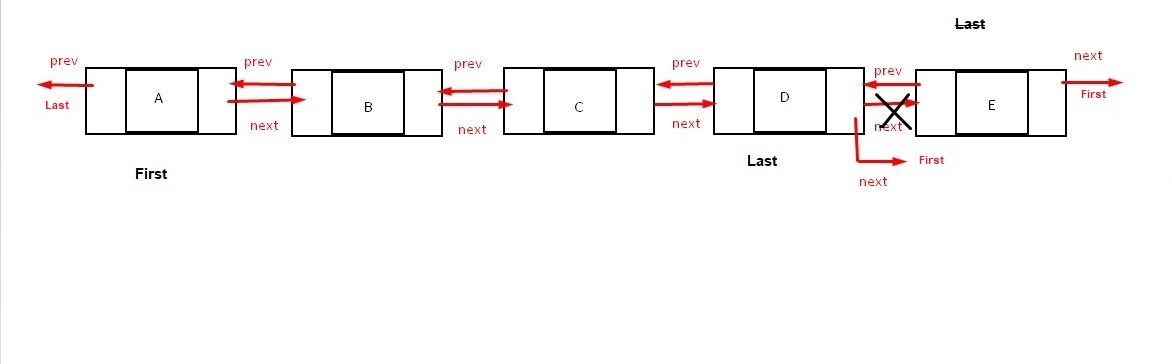
{Kamus Lokal}

{Algoritma}

Pdel🡨Next(Prec)

Next(Prec)🡨Next(Next(Prec))

Prev(Next(Prec)🡨Prec



**Procedure** DelLast (input/output: L: List, output: P: address)

{ I.S. List tidak kosong}

{ F.S. P adalah alamat elemen terakhir list sebelum penghapusan. Elemen list berkurang satu (mungkin menjadi kosong). Last element baru adalah predesesor elemen pertama yg lama, jika ada}

{Kamus}

{Algoritma}

P🡨Prev(First(L))

If(Next(First(L))=First(L)) then {Satu elemen}

First(L)🡨Nil

else

Prev(First(L))🡨Prev(P)

Next(Prev(First(L)))🡨First(L)