05. Double Linked List

ARNA FARIZA

YULIANA SETIOWATI

Capaian Pembelajaran

1. Mahasiswa mengerti konsep double linked list dan operasi pada single linked list.
2. Mahasiswa dapat mengimplementasikan double linked list dalam bahasa pemrograman.

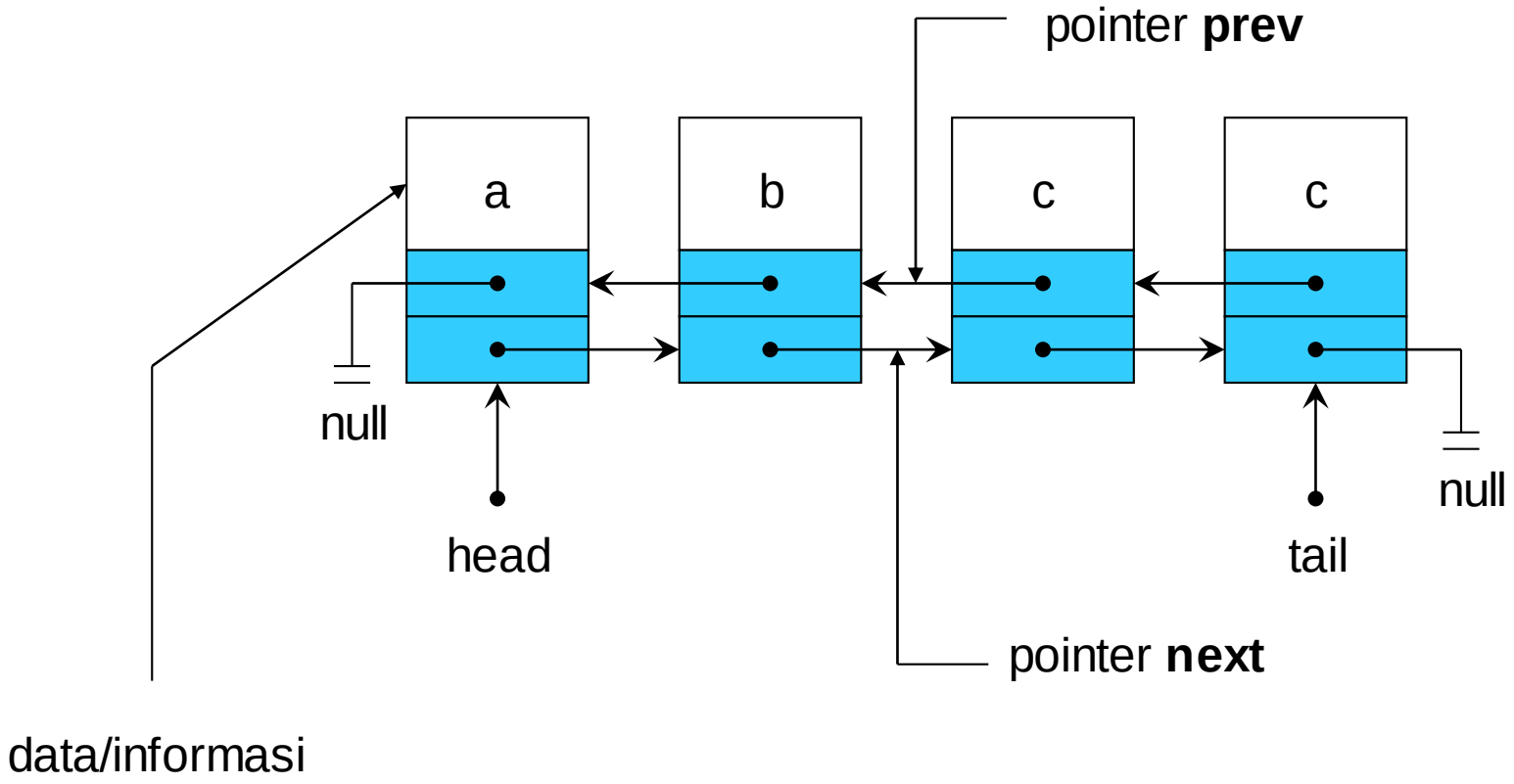
Materi

- Pengertian Double Linked List
- Operasi pada Double Linked List :
  - Mencetak simpul
  - Menyisipkan Simpul
  - Menghapus Simpul
- Implementasi Queue dengan Double Linked List

Double Linked List

- Elemen-elemen dihubungkan dengan dua pointer dalam satu elemen.
- List bisa melintas baik ke depan maupun ke belakang.
- Masing-masing elemen terdiri dari tiga bagian
  - bagian data/informasi
  - pointer *next* yang menunjuk ke elemen berikutnya
  - pointer *prev* yang menunjuk ke elemen sebelumnya
- Untuk menunjukkan *head* dari double linked list, pointer *prev* dari elemen pertama menunjuk NULL.
- Untuk menunjukkan *tail* dari double linked list tersebut, pointer *next* dari elemen terakhir menunjuk NULL.

Double Linked List



Deklarasi Simpul pada Double Linked List

```
typedef struct simpul DNode;
struct simpul {
   int data;
   DNode *next;
   DNode *prev;
};
```

Variabel head, tail dan baru

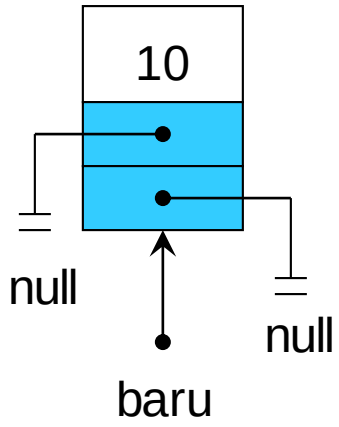
- *head* adalah variabel pointer yang menunjuk ke awal list
- *tail* adalah variabel pointer yang menunjuk ke akhir list
- *baru* adalah variabel pointer yang menunjuk ke simpul baru

```
DNode *head = NULL;
DNode *tail =NULL;
DNode *baru;
```

Alokasi Simpul Baru

```
baru =(DNode *) malloc (sizeof(DNode));
baru->data=x;
baru->next=NULL;
baru->prev=NULL;
```

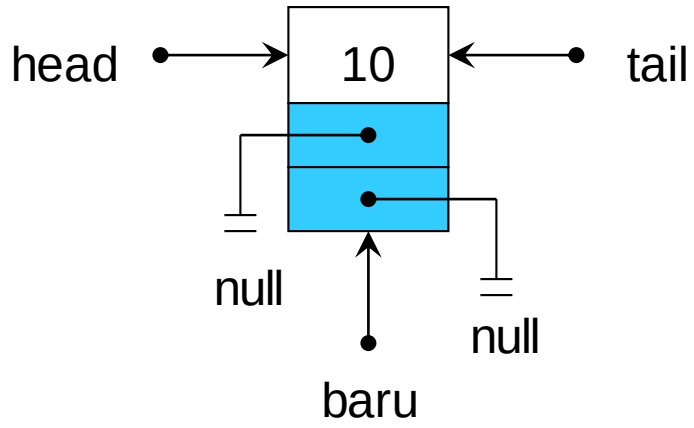
Jika x=10, maka



Membentuk Simpul Awal

- *head* dan *tail* menunjuk awal list, karena hanya ada satu simpul maka *head* dan *tail* menunjuk *baru*.

```
Head = tail = baru;
```

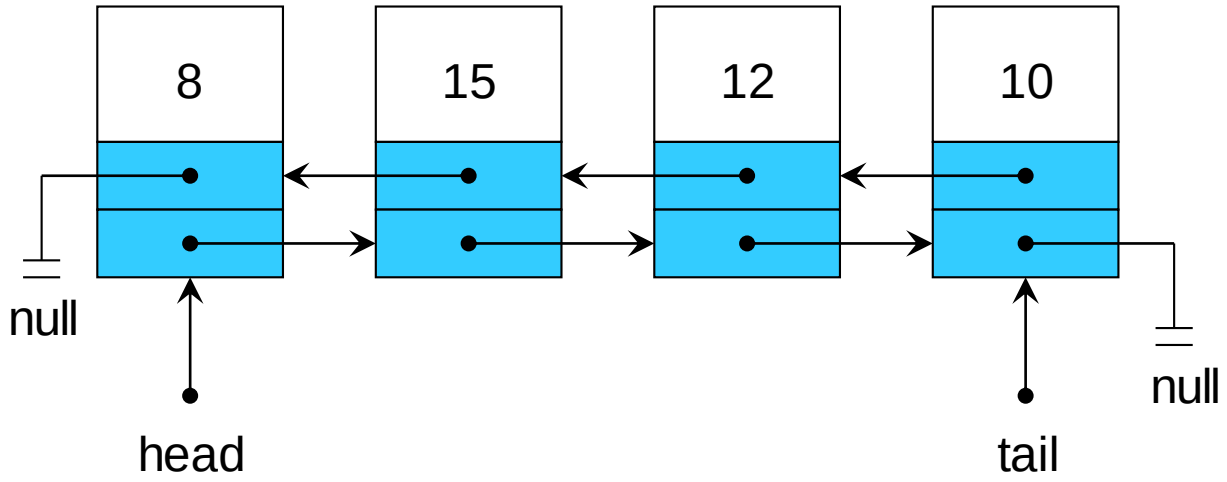


Operasi pada Double Linked List

1. Mencetak Simpul
2. Menyisipkan Simpul
3. Menghapus Simpul

Operasi Mencetak Simpul

- Operasi mencetak simpul dapat dilakukan dengan cara
  - Mencetak dari *head* ke *tail* → 8 15 12 10
  - Mencetak dari *tail* ke *head* → 10 12 15 8



Mencetak dari *head* ke *tail*

```
DNode *p = head;
while (p!= NULL){
   printf("%d ", p->data);
   p = p->next;
}
printf("\n");
```

Mencetak dari *tail* ke *head*

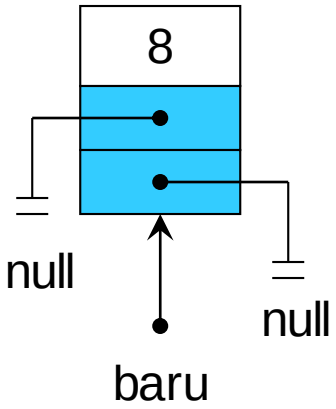
```
DNode *p = tail;
while (p!= NULL){
   printf("%d ", p->data);
   p = p->prev;
}
printf("\n");
```

Operasi Menyisipkan Simpul

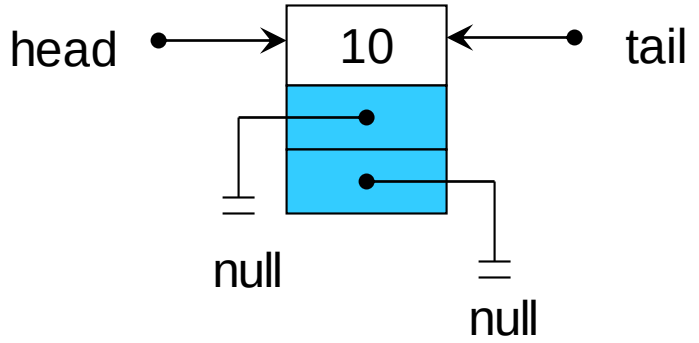
- Operasi menyisipkan simpul terdiri dari:
  - Sisip awal list
  - Sisip akhir list
  - Sisip sebelum simpul tertentu
  - Sisip setelah simpul tertentu

Sisip Awal List

Buat simpul baru:

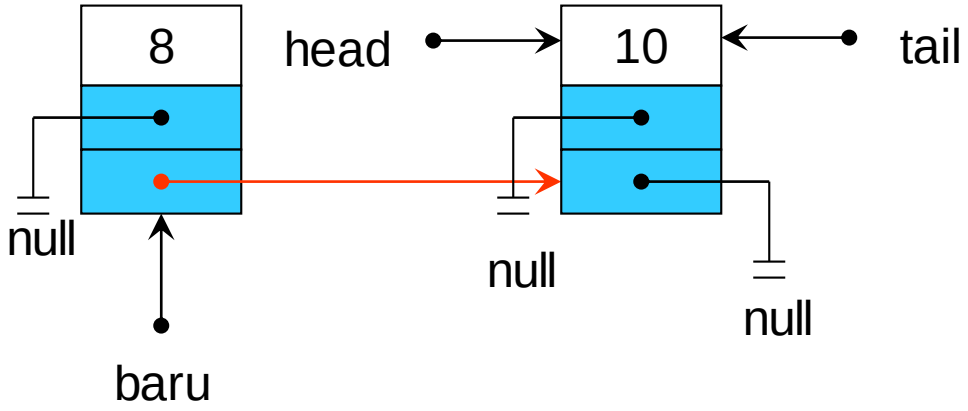


Linked list:



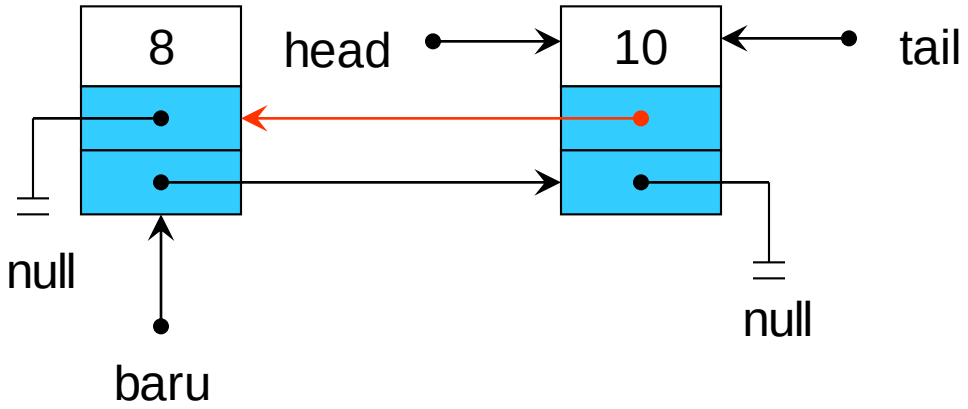
Sisip Awal List

1. *baru->next* menunjuk simpul *head*



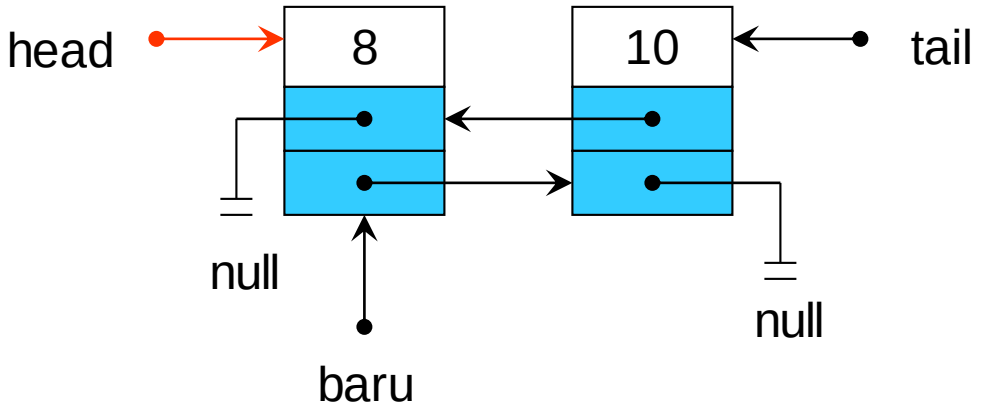
Sisip Awal List

2. *head->prev* menunjuk *baru*



Sisip Awal List

3. *head* menunjuk *baru*

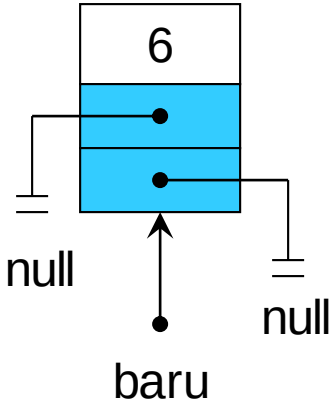


Sisip Awal List

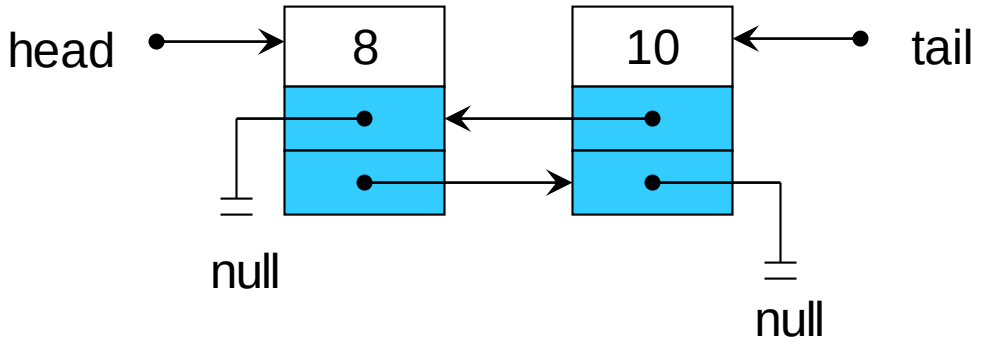
```
baru->next = head;
head->prev = baru;
head = baru;
```

Sisip Akhir List

Buat simpul baru:

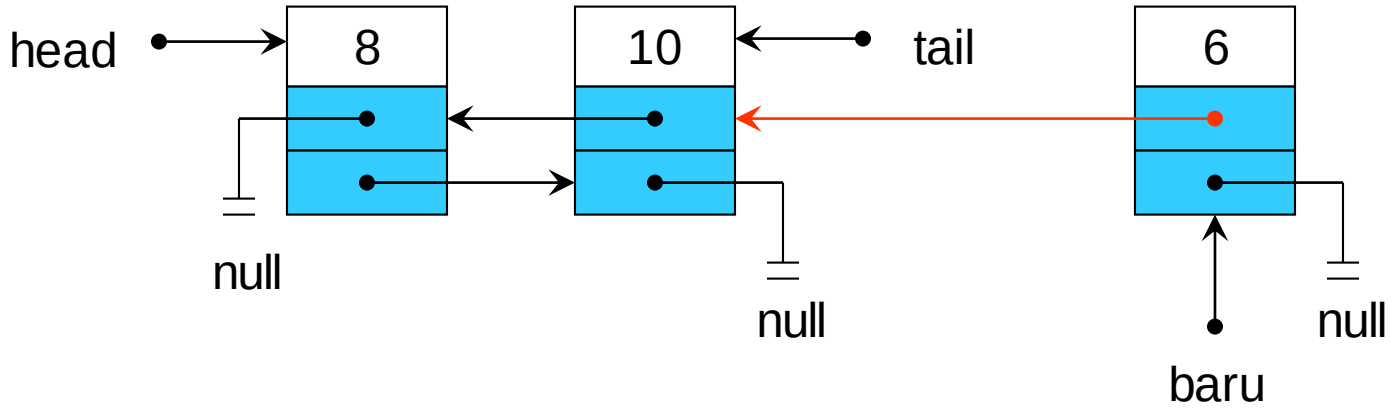


Linked list:



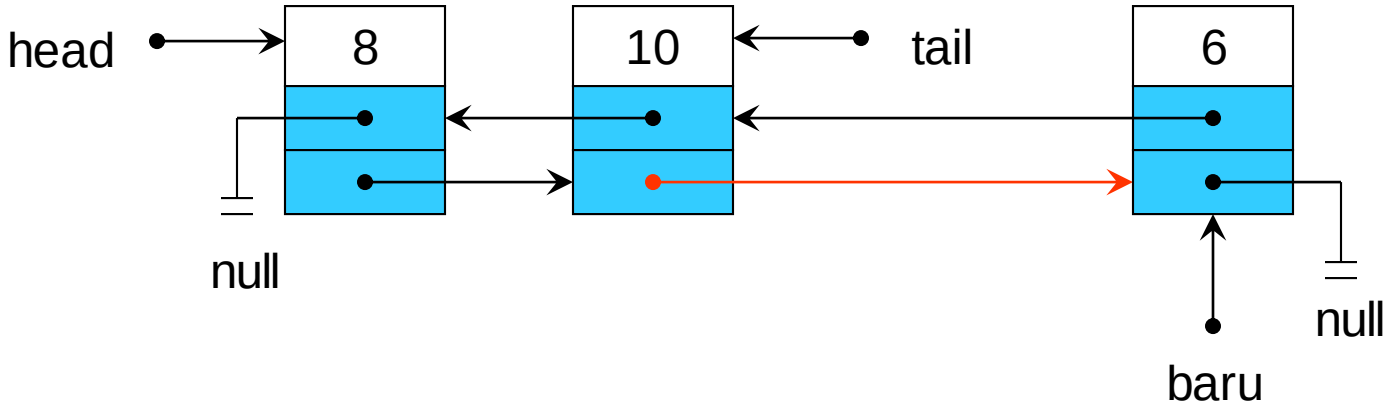
Sisip Akhir List

1. *baru->prev* menunjuk simpul *tail*



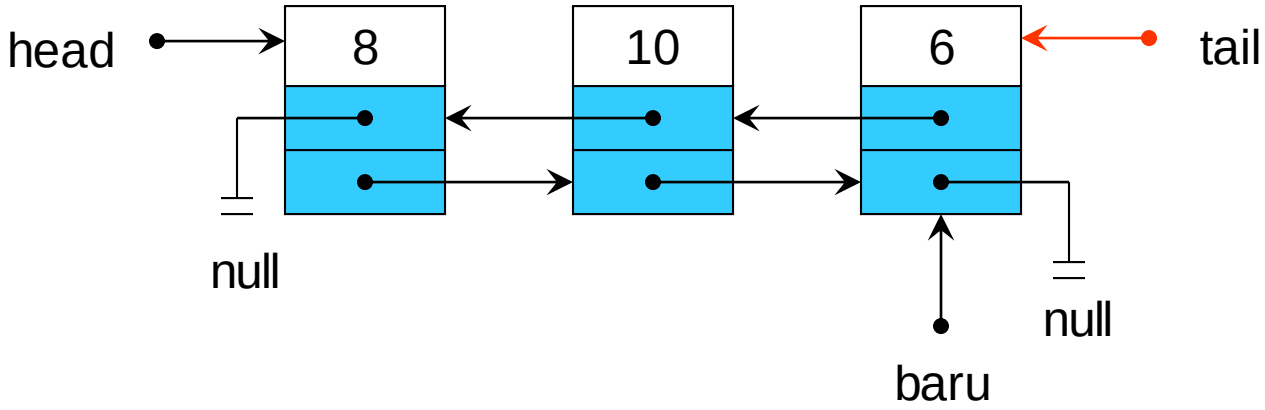
Sisip Akhir List

2. *tail->next* menunjuk simpul *baru*



Sisip Akhir List

3. *tail* menunjuk *baru*

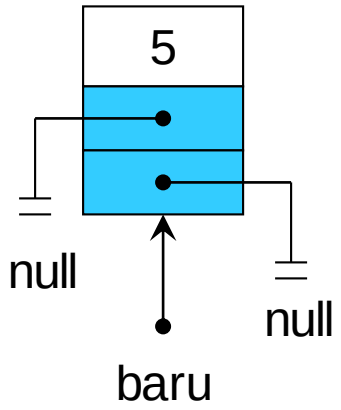


Sisip Akhir List

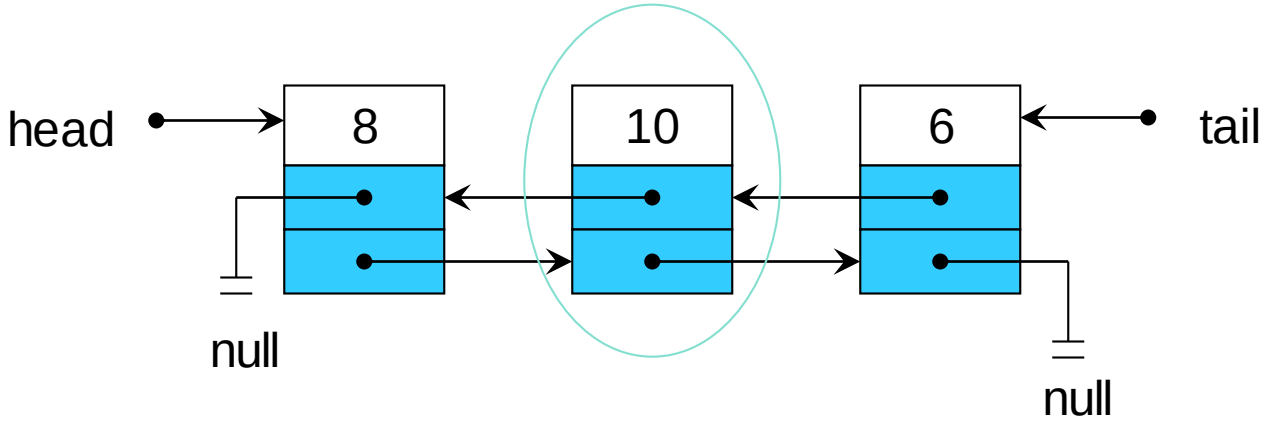
```
baru->prev = tail;
tail->next = baru;
tail = baru;
```

Sisip Setelah Simpul x (misal x=10)

Buat simpul baru:

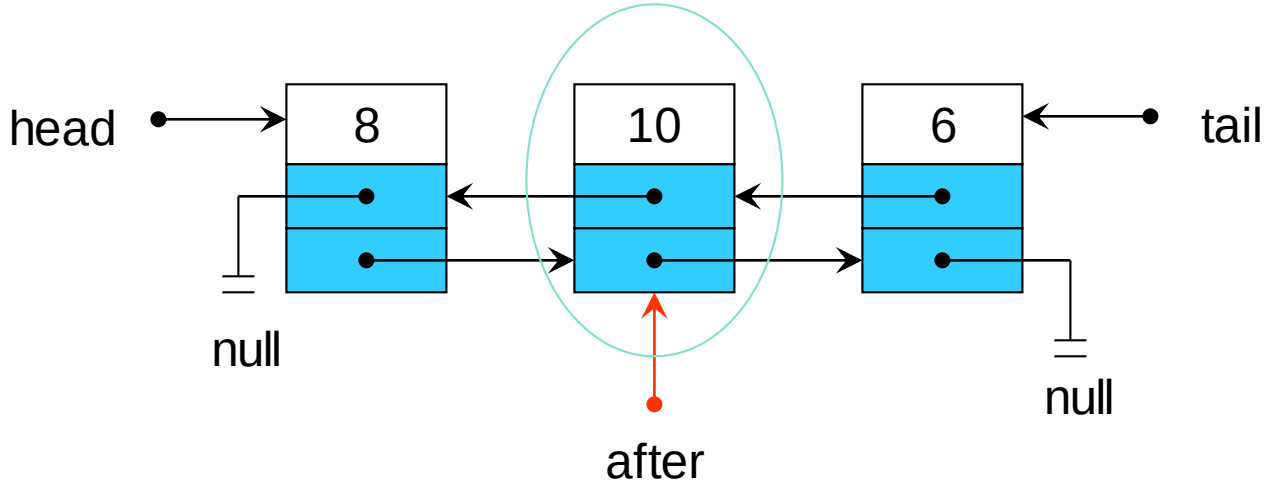


Linked list:



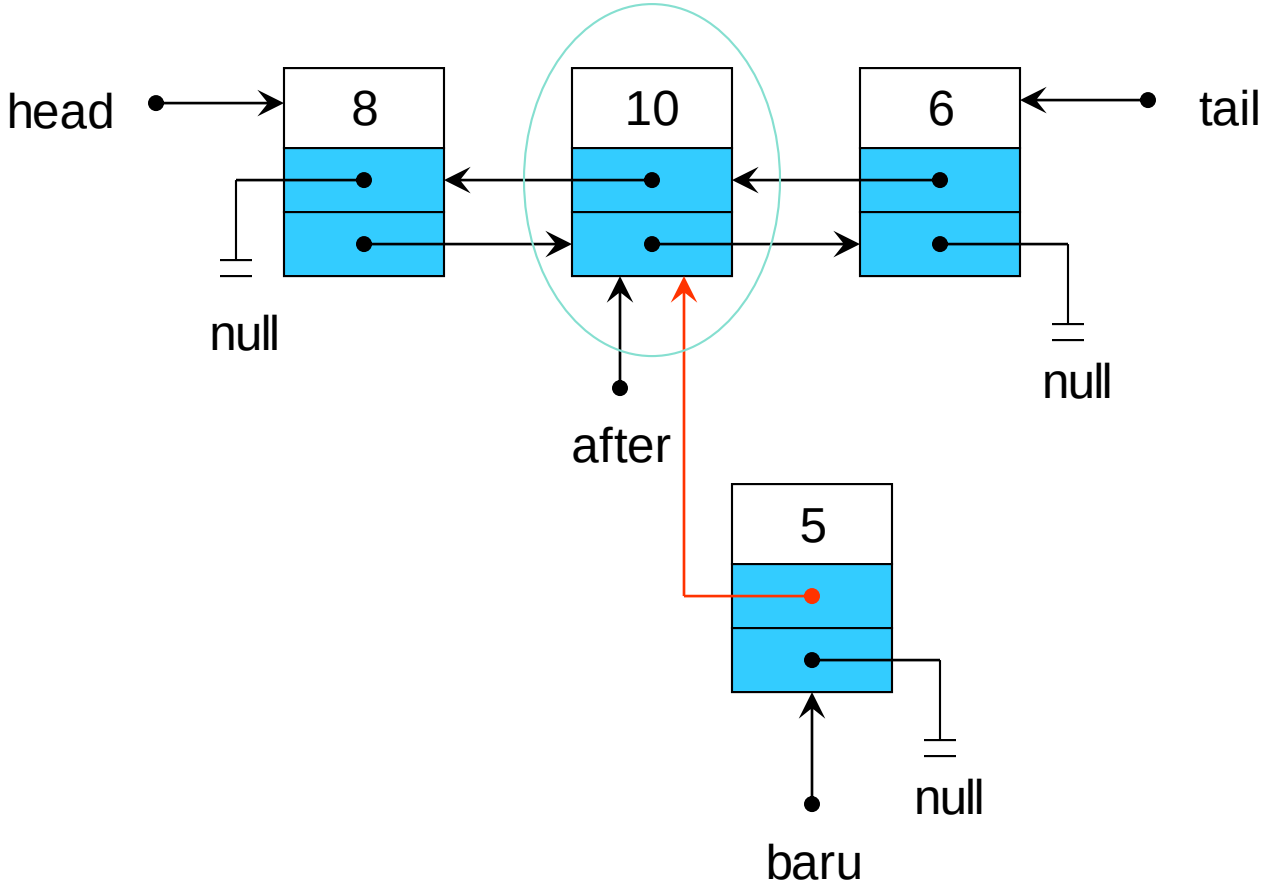
Sisip Setelah Simpul x (misal x=10)

1. *after* diarahkan ke posisi simpul 10, *after* dapat dimulai dari *head* atau *tail*



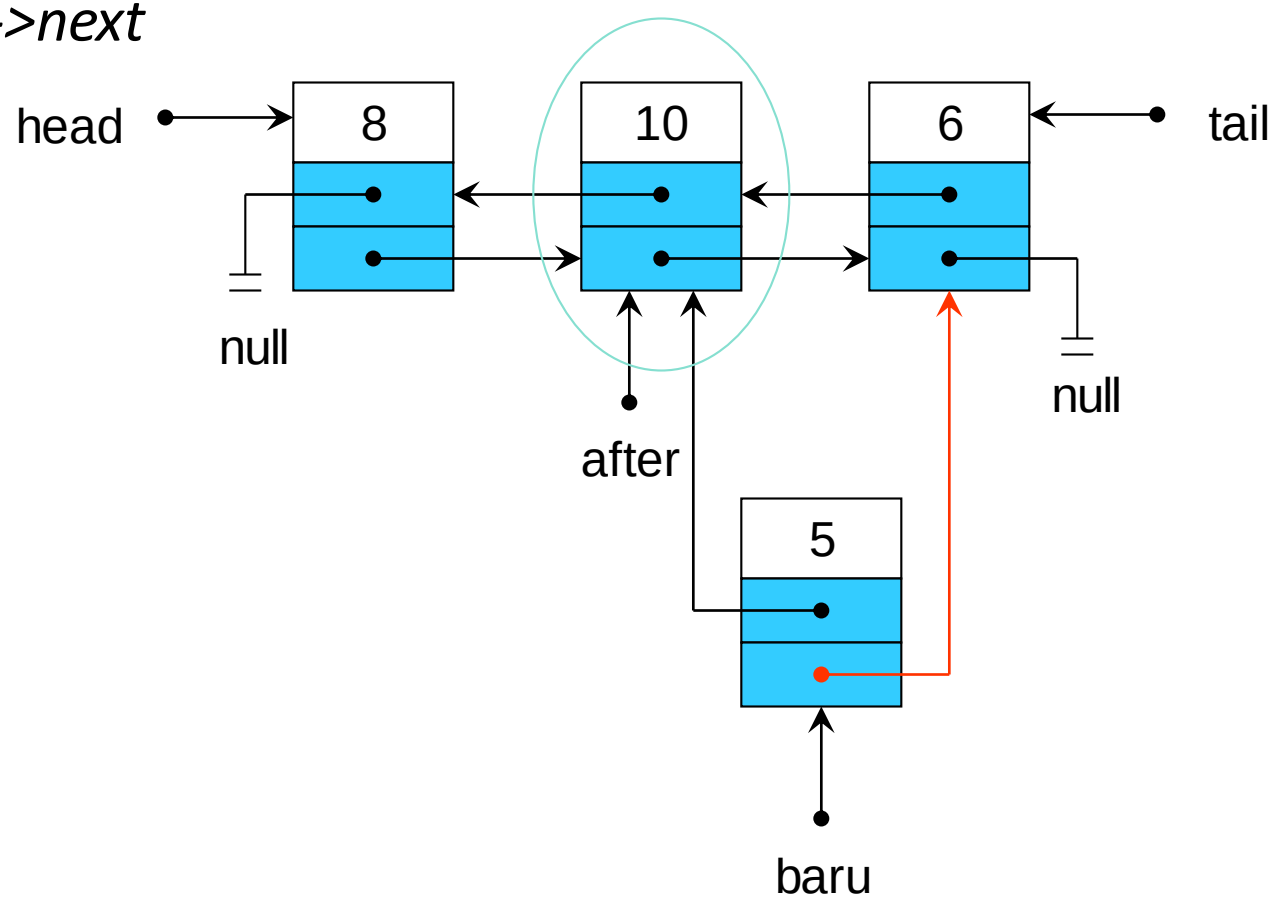
Sisip Setelah Simpul x (misal x=10)

2. *baru->prev* menunjuk *after*



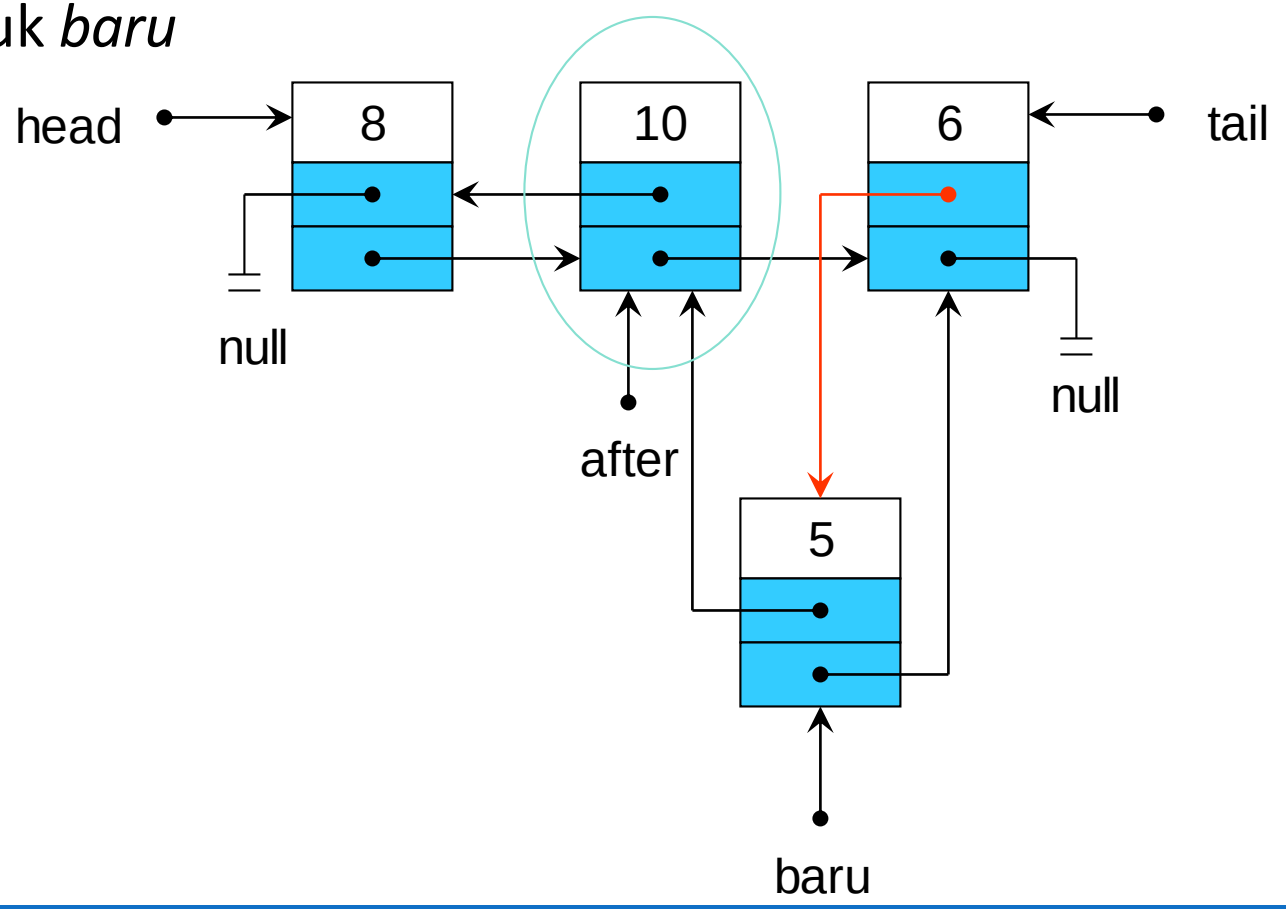
Sisip Setelah Simpul x (misal x=10)

3. *baru->next* menunjuk *after->next*



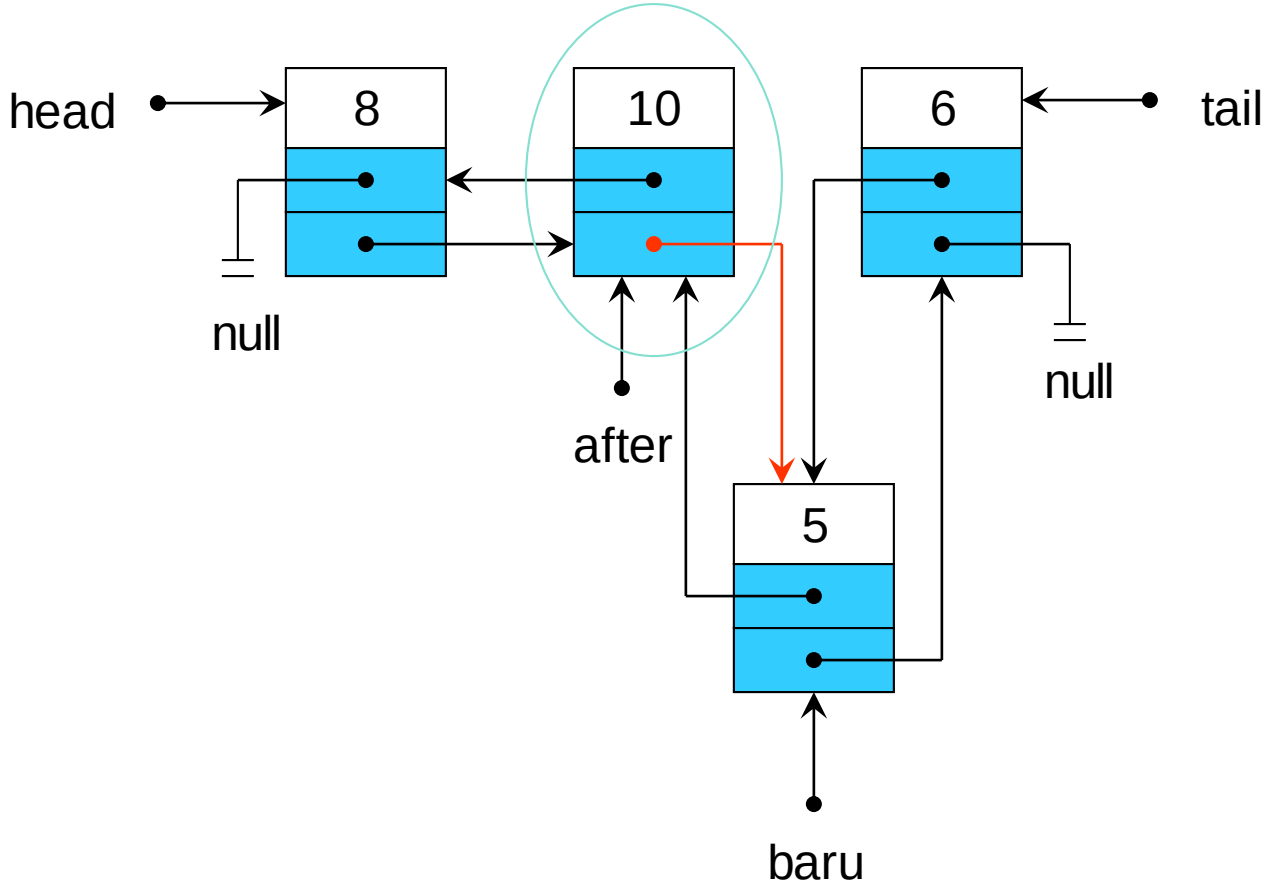
Sisip Setelah Simpul x (misal x=10)

4. *after->next->prev* menunjuk *baru*



Sisip Setelah Simpul x (misal x=10)

5. *after->next* menunjuk *baru*



Sisip Setelah Simpul Tertentu

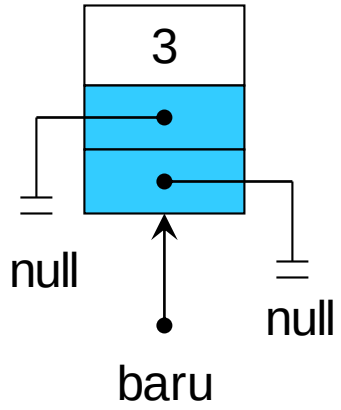
```
DNode *after = head;
while (after->data!= x)
   after = after->next;
baru->prev = after;
baru->next = after->next;
after->next->prev = baru;
after->next = baru;
```

Sisip Setelah Simpul Tertentu

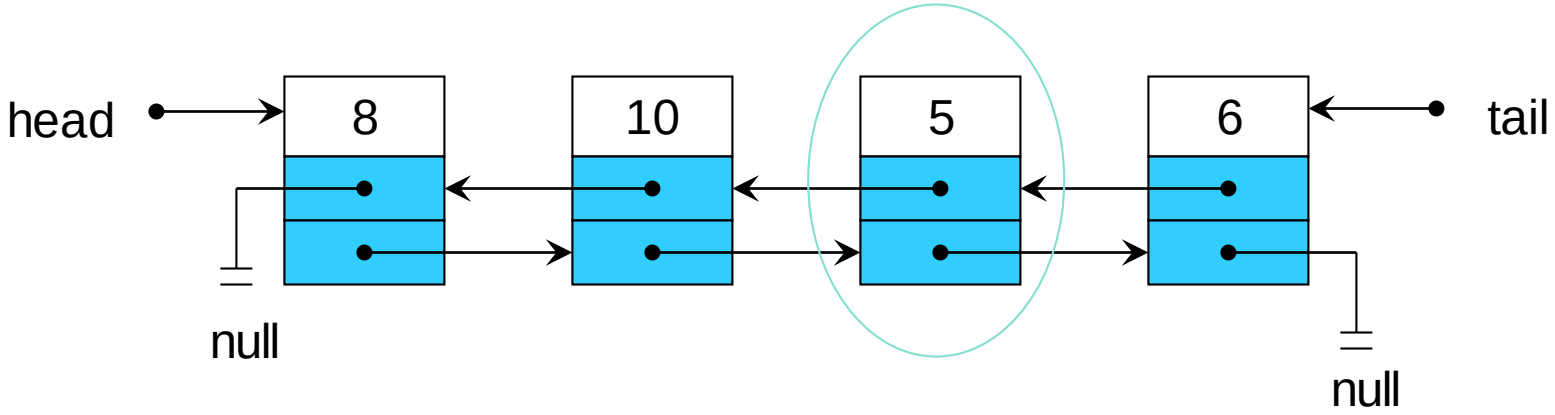
```
DNode *after = tail;
while (after->data!= x)
   after = after->prev;
baru->prev = after;
baru->next = after->next;
after->next->prev = baru;
after->next = baru;
```

Sisip Sebelum Simpul x (misal x=5)

Buat simpul baru:

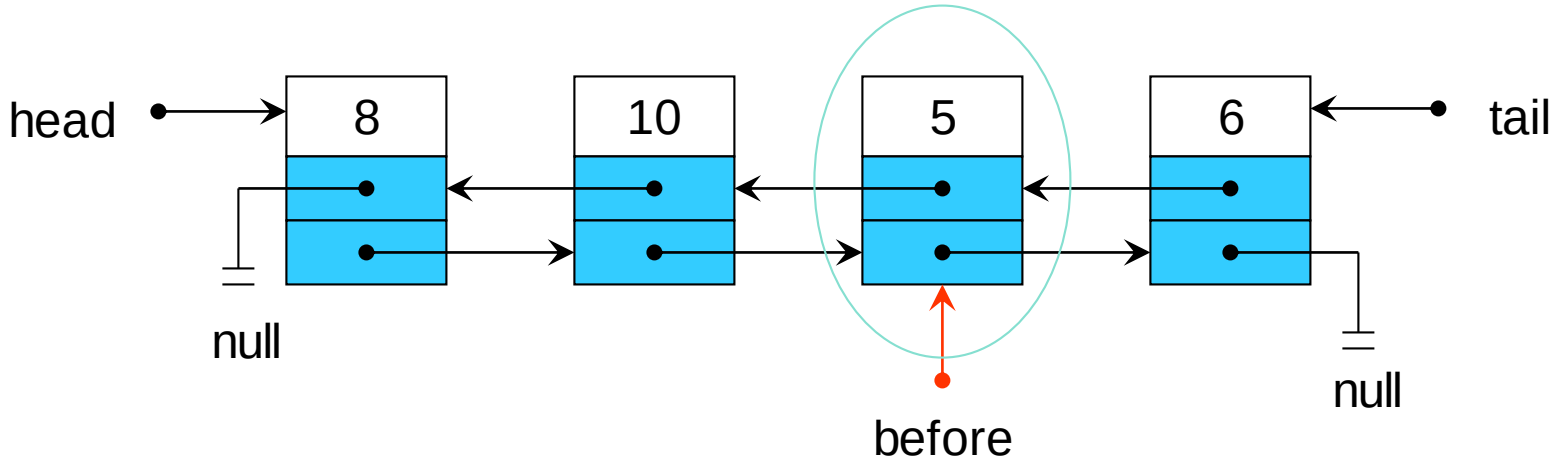


Linked List:



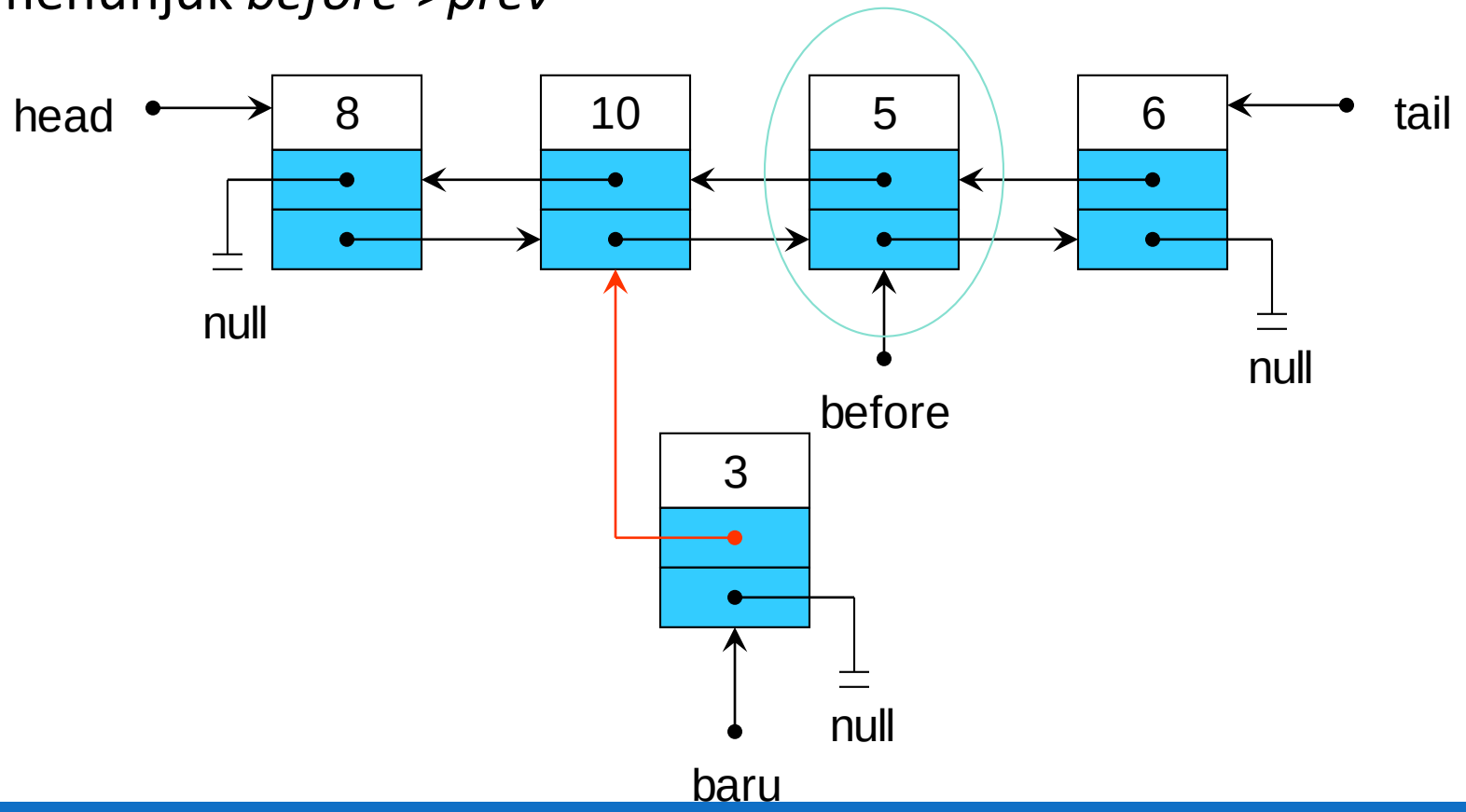
Sisip Sebelum Simpul x (misal x=5)

1. *Before* diarahkan menunjuk ke posisi simpul 5, *before* dapat dimulai dari *head* atau *tail*



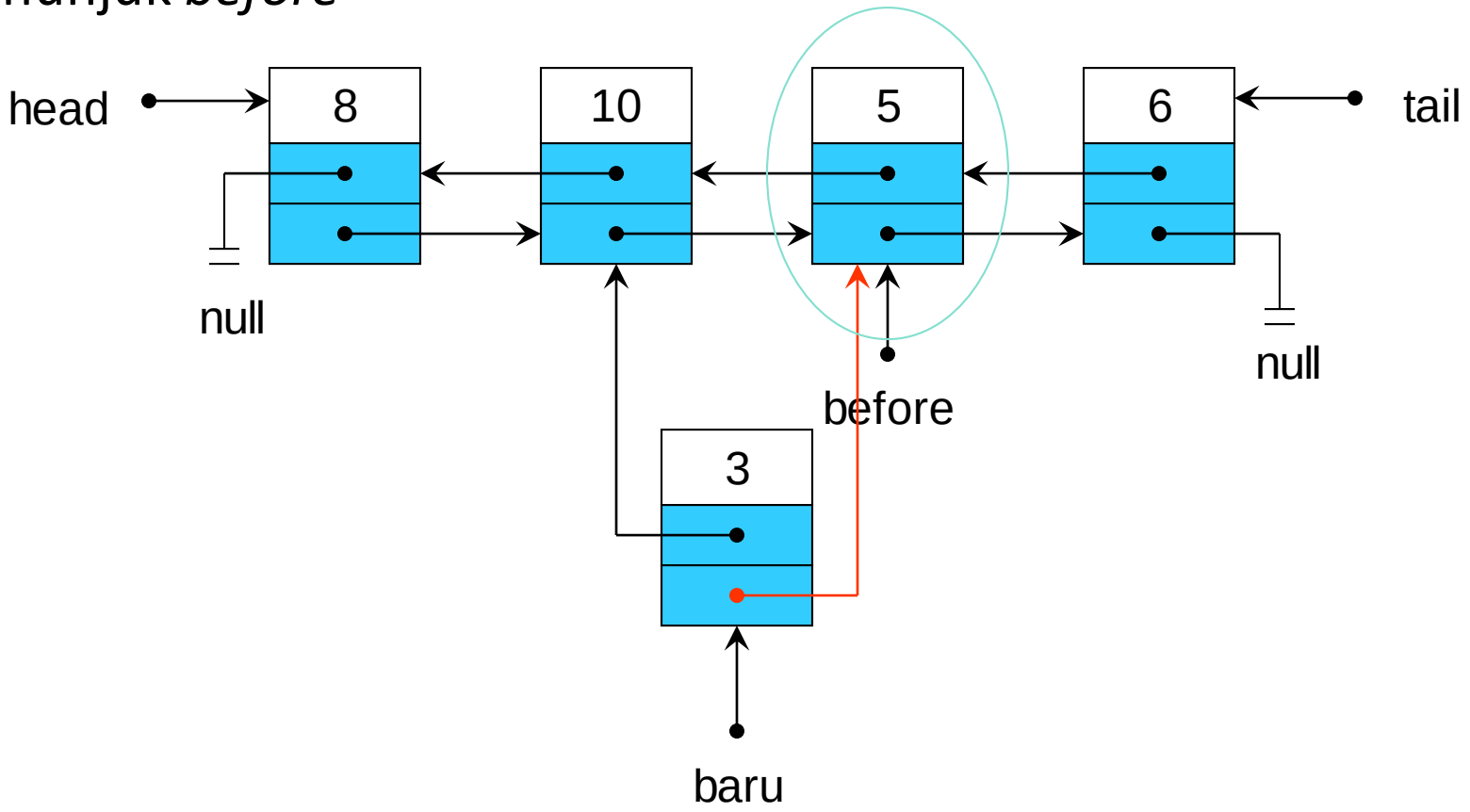
Sisip Sebelum Simpul x (misal x=5)

2. *baru->prev* menunjuk *before->prev*



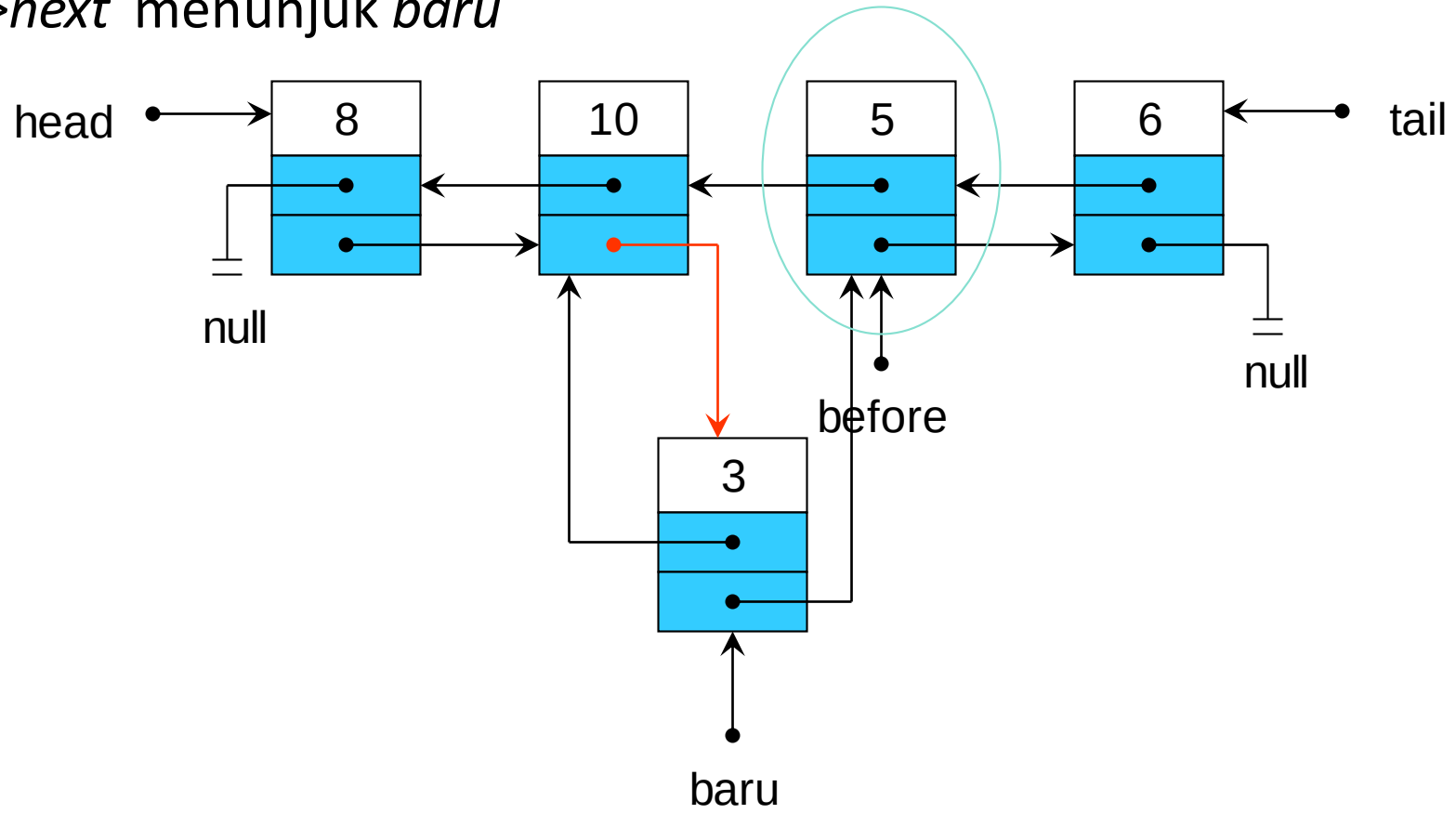
Sisip Sebelum Simpul x (misal x=5)

3. *baru->next* menunjuk *before*



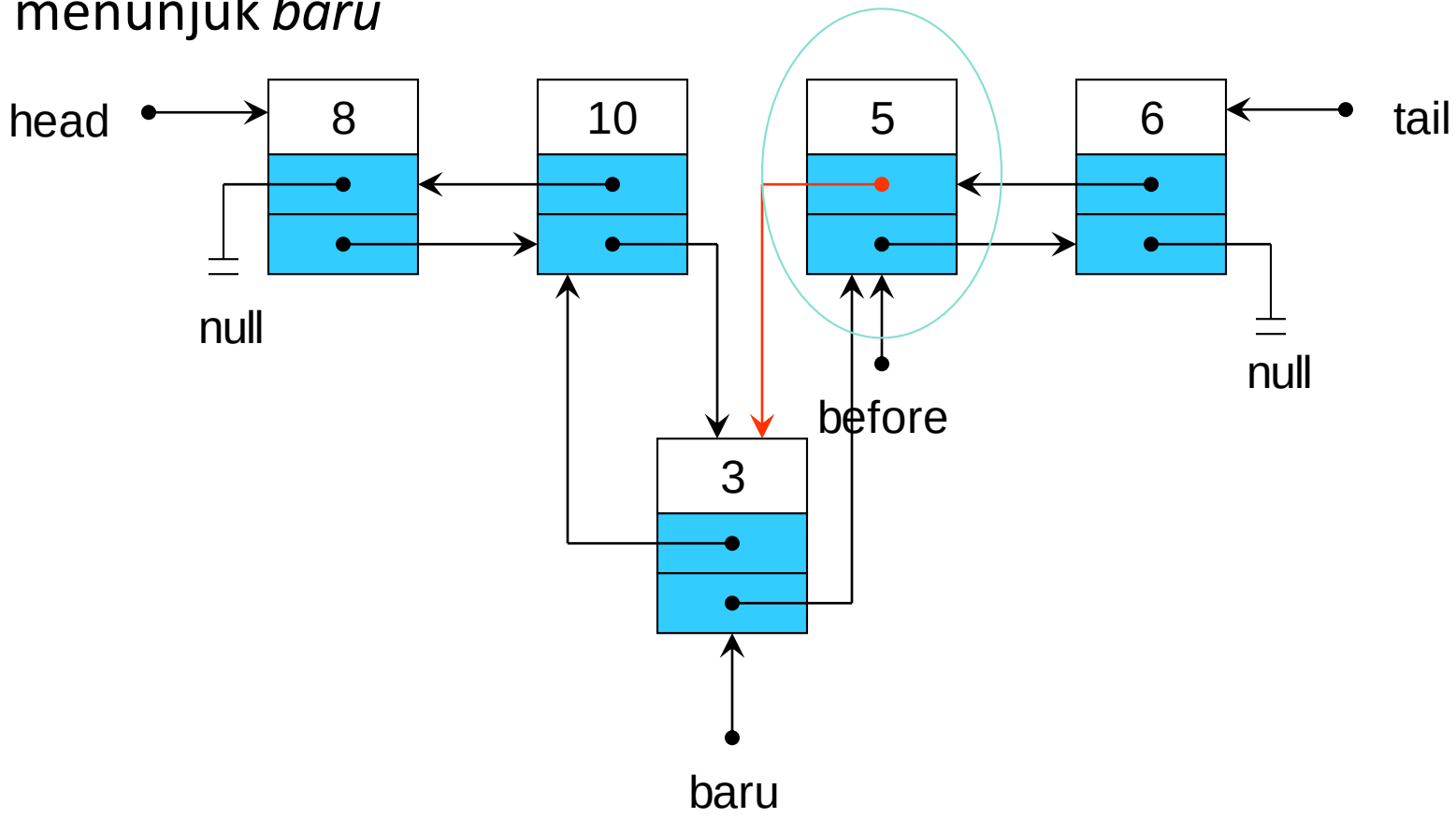
Sisip Sebelum Simpul x (misal x=5)

4. *before->prev->next* menunjuk *baru*



Sisip Sebelum Simpul x (misal x=5)

5. *before->prev* menunjuk *baru*



Sisip Sebelum Simpul Tertentu

```
DNode *before = head;
while (before->data!= x)
   before = before->next;
baru->prev = before->prev;
baru->next = before;
before->prev->next = baru;
before->prev = baru;
```

Sisip Sebelum Simpul Tertentu

```
DNode *before = tail;
while (before->data!= x)
   before = before->prev;
baru->prev = before->prev;
baru->next = before;
before->prev->next = baru;
before->prev = baru;
```

Operasi Menghapus Simpul

- Operasi menghapus simpul terdiri dari:
  - Hapus simpul awal
  - Hapus simpul akhir
  - Hapus simpul tertentu

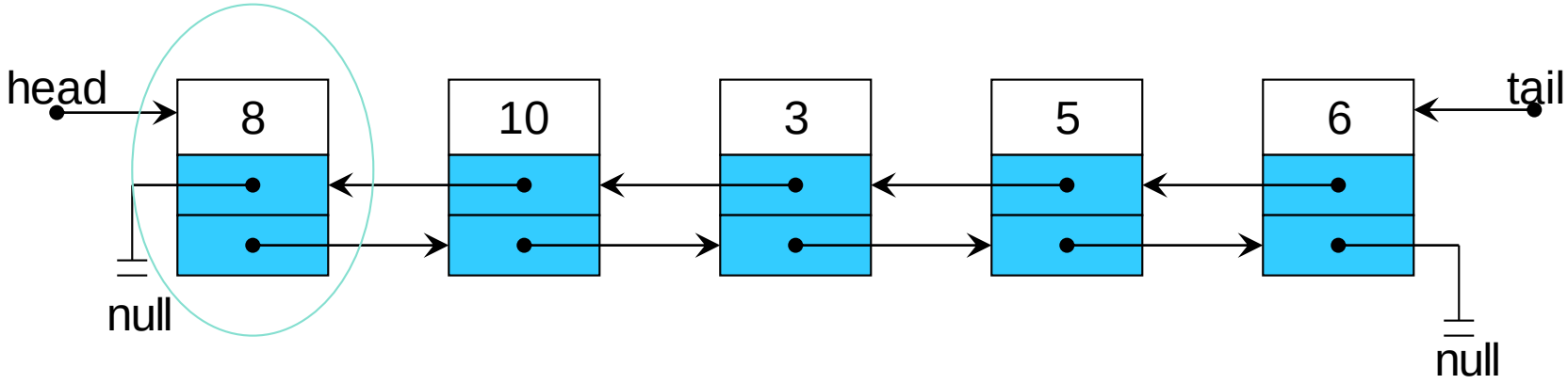
Fungsi free_DNode

- Sebelum menghapus simpul, buat fungsi untuk membebaskan alokasi memori dengan fungsi *free*

```
void free_DNode(DNode *p)
{
   free(p);
   p=NULL;
}
```

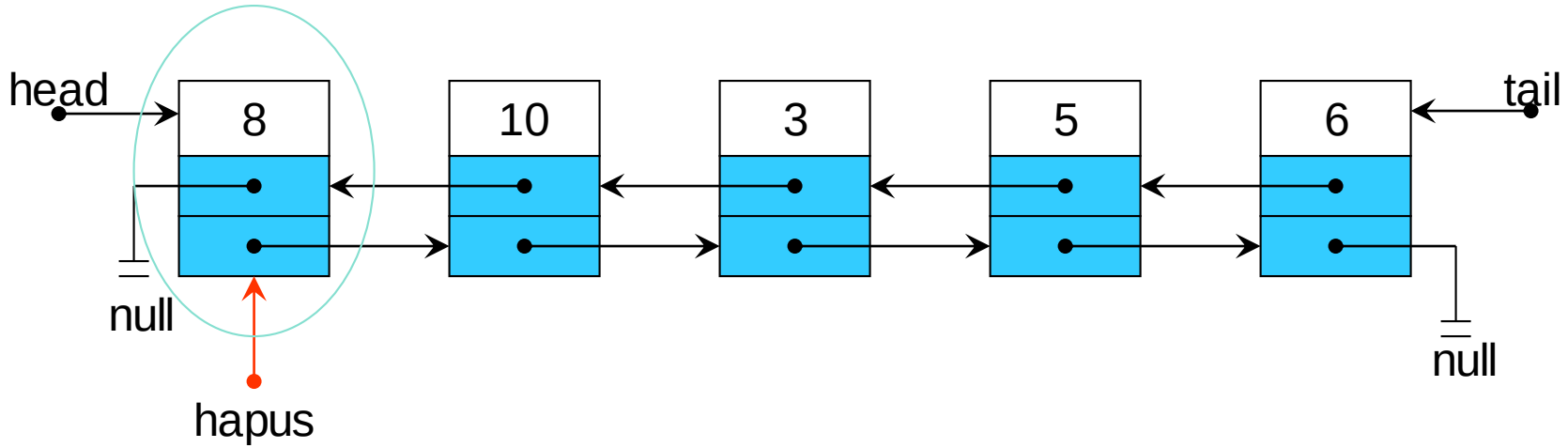
Hapus Simpul Awal

Linked List



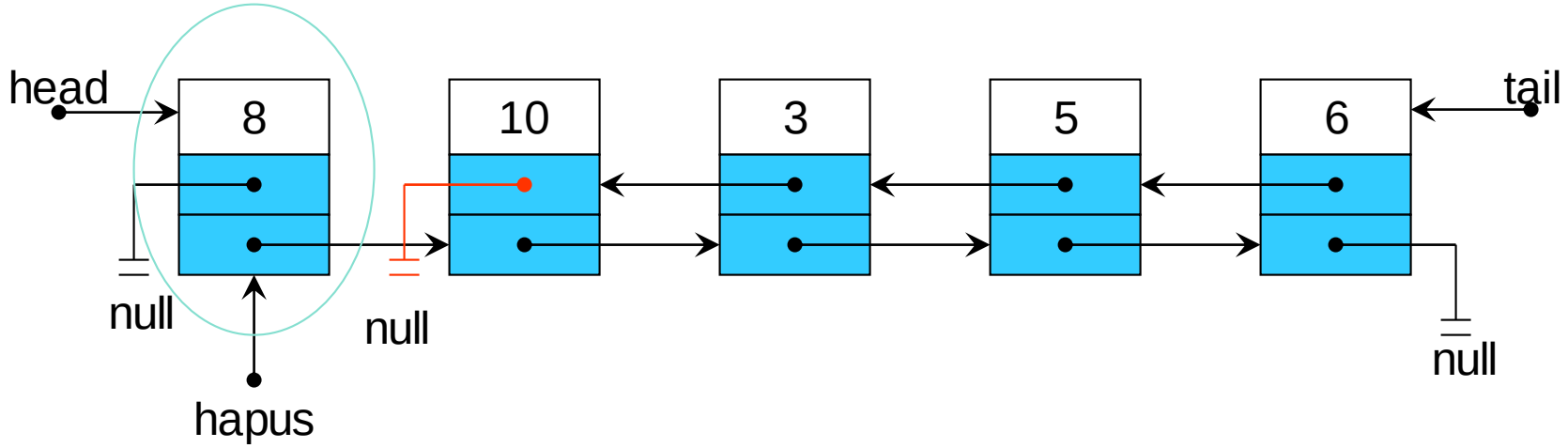
Hapus Simpul Awal

1. *hapus* menunjuk *head*



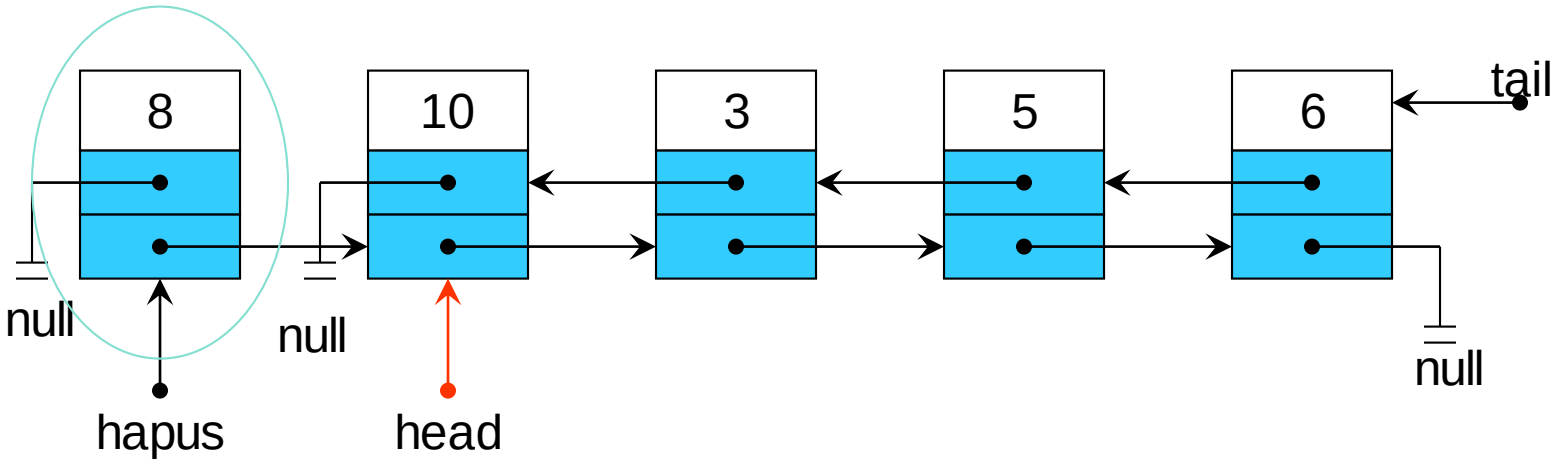
Hapus Simpul Awal

2. *head->next-> prev* menunjuk *NULL*



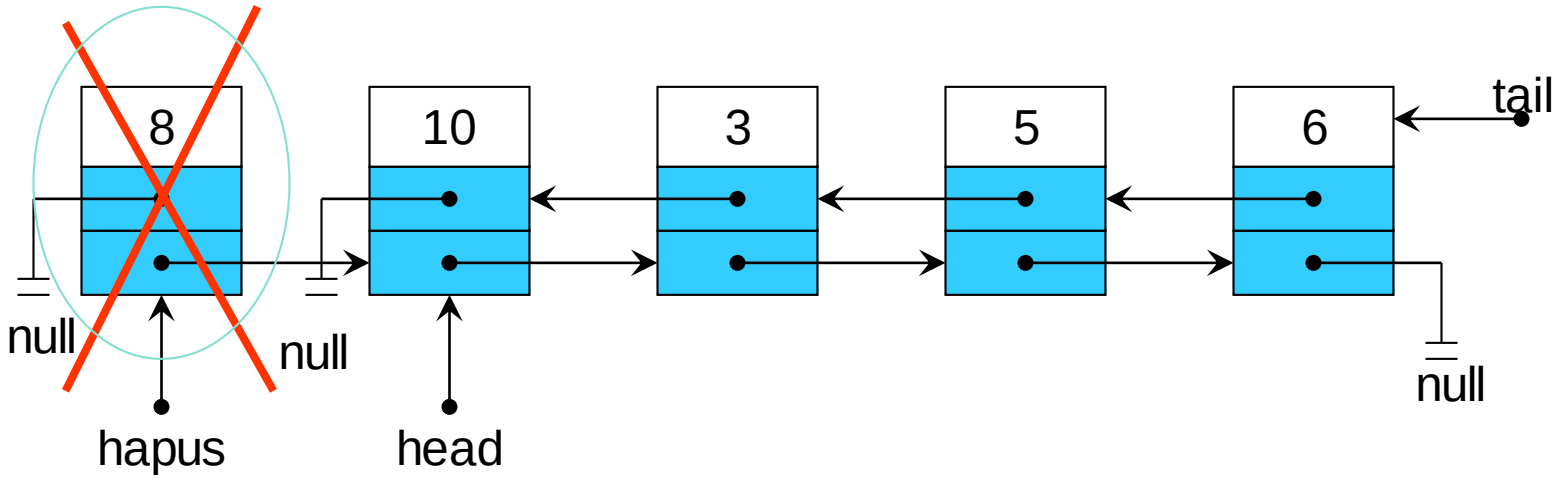
Hapus Simpul Awal

3. *head* menunjuk *hapus->next*



Hapus Simpul Awal

4. *free_DNode(hapus)*

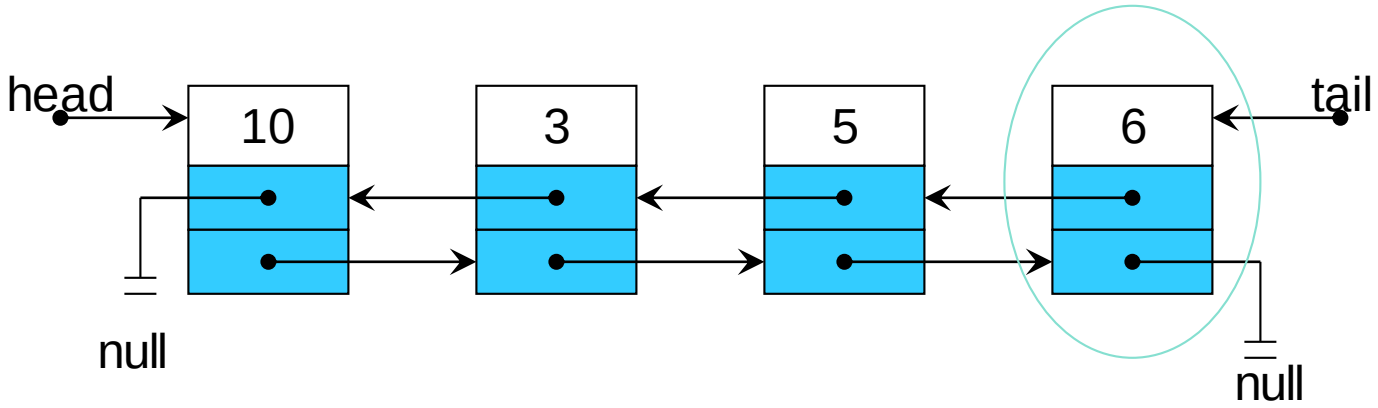


Hapus Simpul Awal

```
Dnode hapus = head;
head->next->prev = NULL;
head = hapus->next;
free_DNode(hapus);
```

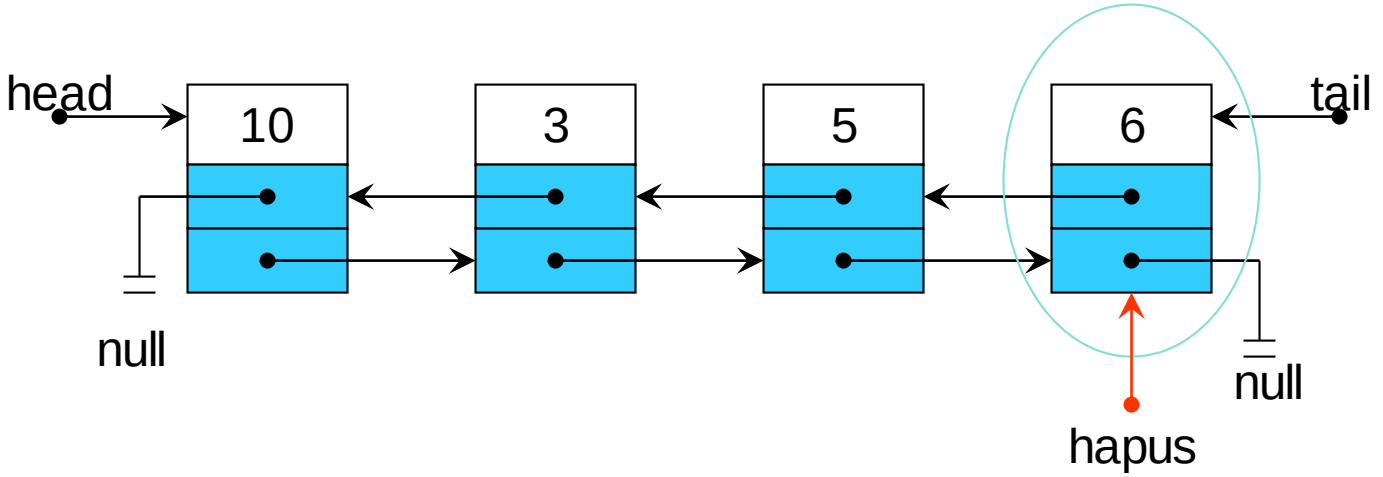
Hapus Simpul Akhir

Linked List



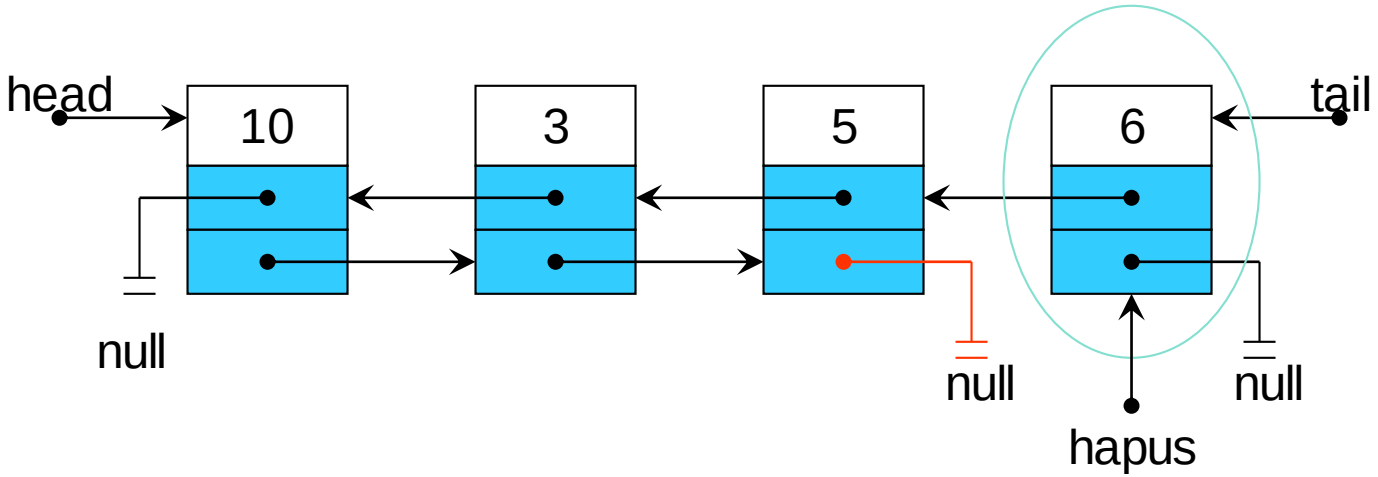
Hapus Simpul Akhir

1. *hapus* menunjuk *tail*



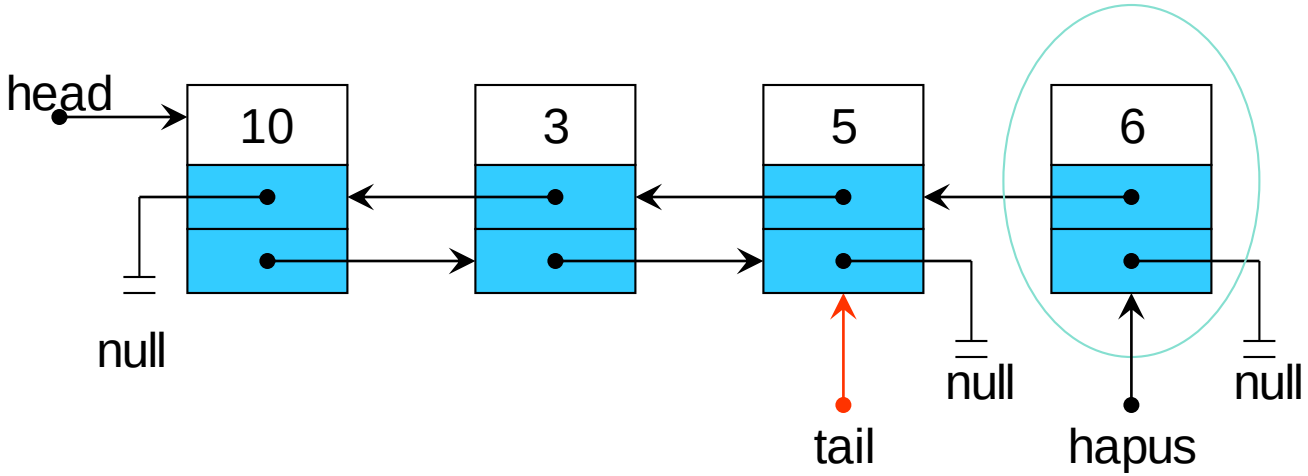
Hapus Simpul Akhir

2. *tail->prev->next* menunjuk *null*



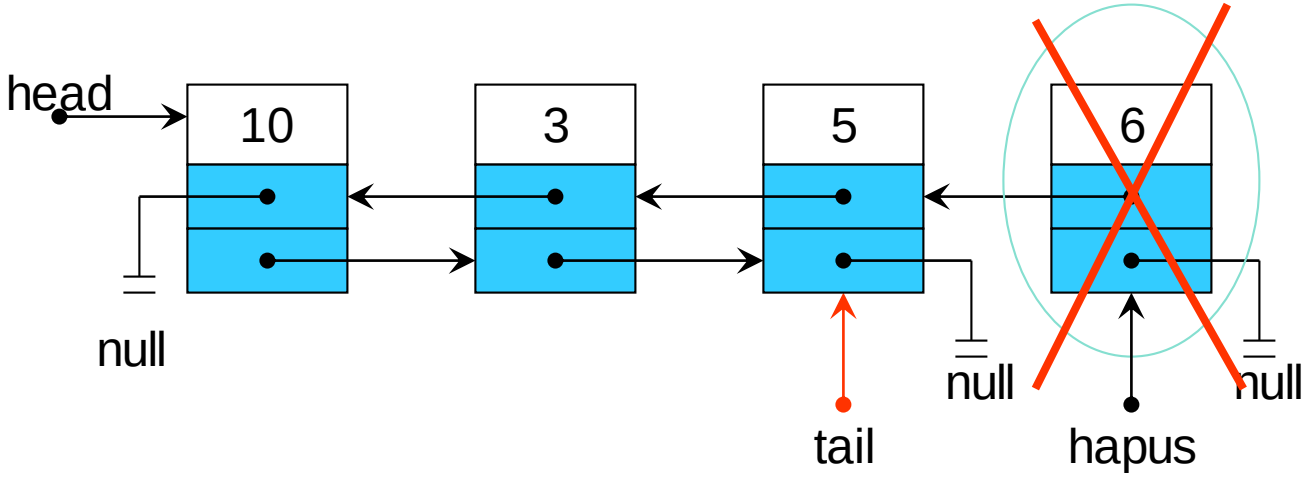
Hapus Simpul Akhir

3. *tail* menunjuk *hapus->prev*



Hapus Simpul Akhir

4. *free_DNode(hapus)*

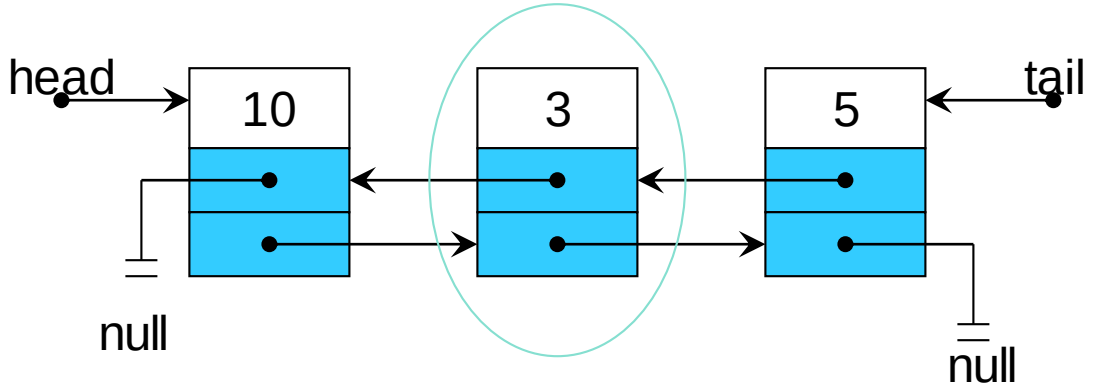


Hapus Simpul Akhir

```
Dnode hapus = tail;
tail->prev->next = NULL;
tail = hapus->prev;
free_DNode(hapus);
```

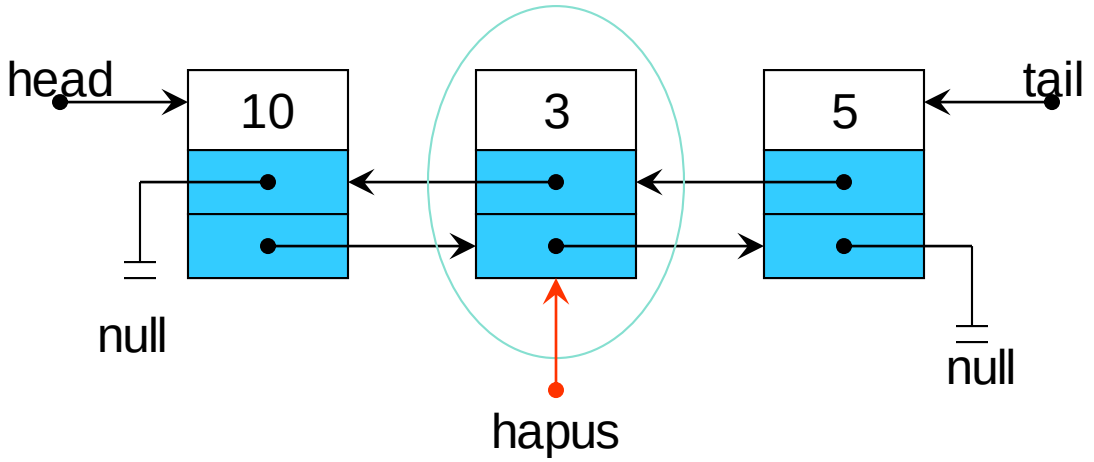
Hapus Simpul Tertentu (misal x=3)

Linked List



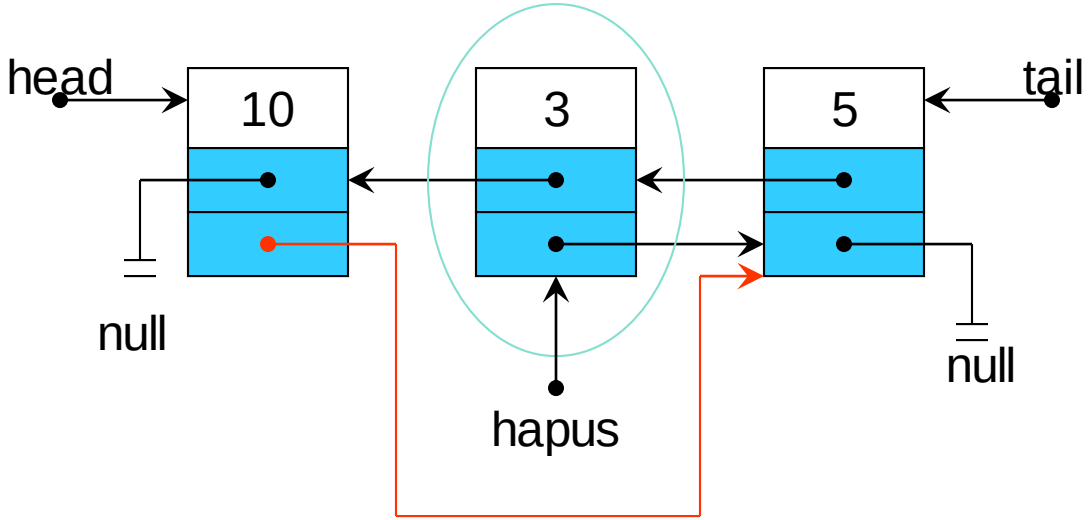
Hapus Simpul Tertentu (misal x=3)

1. *hapus* menunjuk simpul x=3



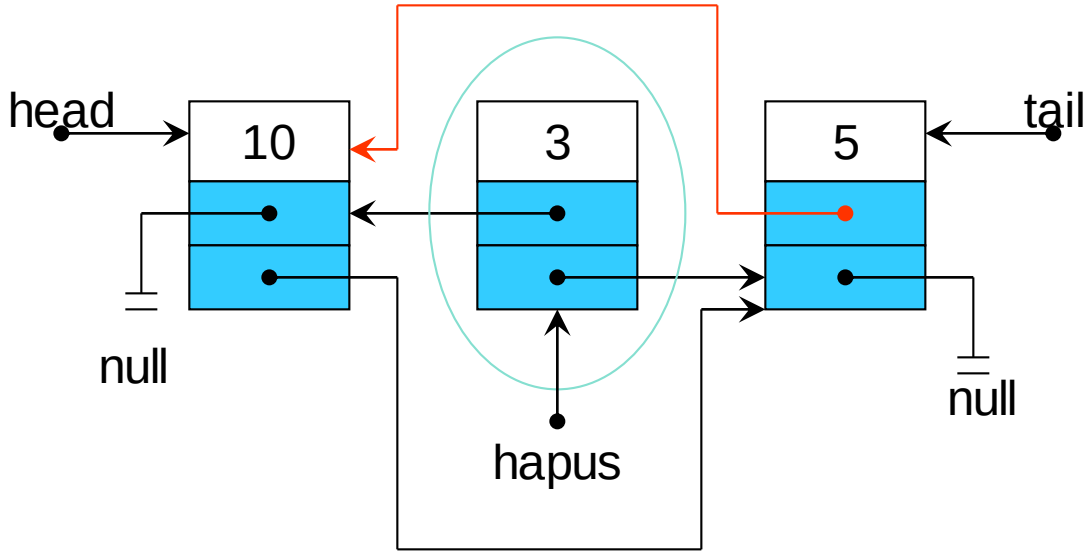
Hapus Simpul Tertentu (misal x=3)

2. *hapus->prev->next* menunjuk *hapus->next*



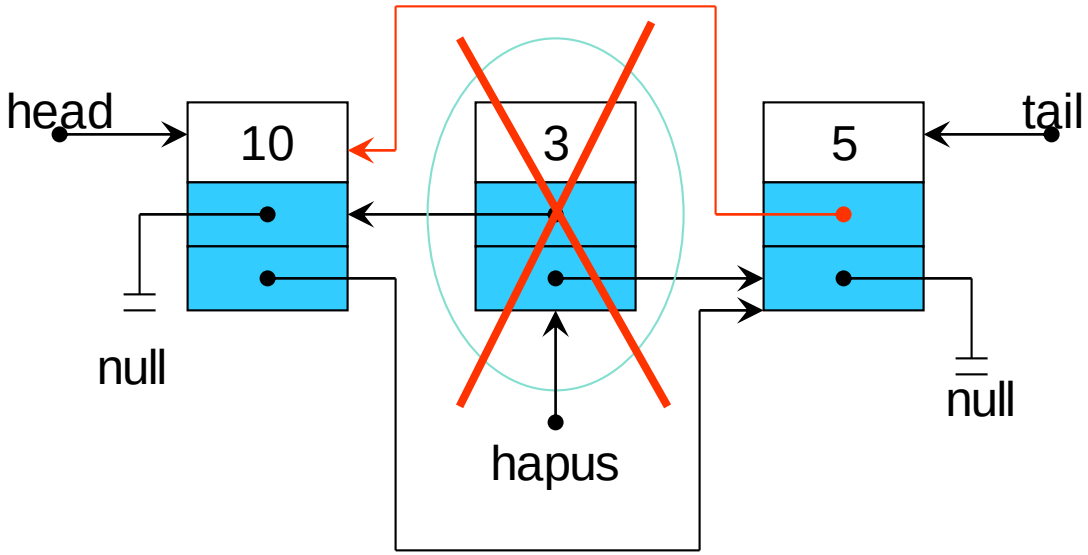
Hapus Simpul Tertentu (misal x=3)

3. *hapus->next->prev* menunjuk *hapus->prev*



Hapus Simpul Tertentu (misal x=3)

4. *free_DNode(hapus)*



Hapus Simpul Tertentu

```
Dnode hapus = head;
while (hapus->data != x)
   hapus = hapus->next;
hapus->prev->next = hapus->next;
hapus->next->prev = hapus->prev;
free_DNode(hapus);
```

Hapus Simpul Tertentu

```
Dnode hapus = tail;
while (hapus->data != x)
   hapus = hapus->prev;
hapus->prev->next = hapus->next;
hapus->next->prev = hapus->prev;
free_DNode(hapus);
```

Implementasi Queue dengan Double Linked List

- Queue adalah penyimpanan data dengan konsep First In First Out (FIFO)
- Terdapat dua penunjuk yaitu *front* dan *rear*
- Fungsi Enqueue menambah simpul pada posisi *rear*
- Fungsi Dequeue menghapus simpul pada posisi *front*

Deklarasi Queue dengan Double Linked List

```
typedef struct simpul DNode;
typedef int itemtype;
struct simpul {
   itemtype item;
   DNode *next;
   DNode *prev;
};
typedef struct {
   DNode *front;
   DNode *rear;
   int count;
} Queue;
```

Fungsi pada Queue

- Inisialisasi → *front* dan *rear* menunjuk NULL, *count*=0
- Kosong → jika *rear* = NULL atau *front* = NULL
- Enqueue → buat simpul, sisip akhir list (posisi *rear*)
- Dequeue → hapus awal list (posisi *front*)

Fungsi Inisialisasi

```
void inisialisasi (Queue *q)
{
   q->front = NULL;
   q->rear = NULL;
   q->count = 0;
}
```

Fungsi Kosong

```
int Kosong (Queue *q)
{
   return(q->rear==NULL)
}
```

Fungsi Enqueue

```
void Enqueue (Queue *q, itemtype x)
{
    DNode *baru = (DNode *) malloc (sizeof(DNode));
    if(baru==NULL) {
        printf("Alokasi gagal\n");
        exit(1);
    }
    else {
        baru->item = x;
        baru->next = NULL;
        baru->prev = q->rear;
        q->rear->next = baru;
        q->rear = baru;
        q->count++;
    }
}
```

Fungsi Dequeue

```
itemtype Dequeue (Queue *q)
{
    DNode *hapus;
    itemtype temp;
    if(Kosong(q)) {
        printf("Queue kosong\n");
        return ' ';
    }
    else {
        temp = q->front->item;
        hapus = q->front;
        q->front = hapus->next;
        q->front->prev = NULL;
        free(hapus);
        q->count--
        return temp;
    }
}
```

Rangkuman

- Simpul pada double linked list terdiri dari bagian data, pointer *next* dan pointer *prev*
- Operasi pada double linked list terdiri dari operasi cetak, sisip dan hapus
- Operasi cetak dapat dilakukan dari *head* ke *tail* atau dari *tail* ke *head*
- Operasi sisip terdiri dari sisip awal list, sisip akhir list, sisip setelah simpul tertentu, sisip sebelum simpul tertentu
- Operasi hapus terdiri dari hapus awal list, hapus akhir list dan hapus simpul tertentu
- Implementasi Queue dengan double linked list, pada operasi Enqueue dengan sisip akhir list dan pada operasi Dequeue dengan hapus awal list

Latihan

1. Buatlah double linked list dengan data bertipe integer yang dapat melakukan operasi sisip secara terurut dan hapus simpul tertentu dengan ketentuan sebagai berikut :
   a) Operasi sisip, buat sebuah simpul baru,
      - Jika data simpul baru < data pada head, maka gunakan sisip awal list
      - Jika pencarian data simpul baru mencapai NULL (data belum ada), sisip akhir list
      - Jika data simpul baru = data pada simpul tertentu maka berikan pesan simpul sudah ada (duplikat)
      - Lainnya sisip sebelum simpul tertentu
   b) Operasi hapus,
      - Jika posisi simpul yang dihapus pada head, gunakan hapus awal list
      - Jika posisi simpul yang dihapus pada tail, gunakan hapus akhir list
      - Lainnya hapus simpul tengah

Latihan

2. Implementasikan Queue dengan double linked list, buatlah menu Enqueue, Dequeue dan Tampil
3. Buatlah double linked list dengan simpul berupa data mahasiswa yang terdiri dari NRP, Nama dan Kelas. Buatlah operasi Sisip secara terurut, Hapus data mahasiswa tertentu dan Update data (nama dan kelas saja)

Project

Implementasikan sebuah multiple list sistem akademik. Data terdiri dari mahasiswa dan nilai mahasiswa. Data mahasiswa terdiri dari NRP, Nama dan Prodi. Sedangkan data nilai terdiri dari Kode MK, Nama MK dan Nilai. Buatlah hubungan antara mahasiswa dan nilai, dimana satu mahasiswa mengambil beberapa mata kuliah. Fungsi yang dibuat terdiri dari:

1. Sisip mahasiswa secara terurut
2. Hapus mahasiswa
3. Update mahasiswa (Nama dan Prodi)
4. Sisip MK per mahasiswa
5. Hapus MK per mahasiswa
6. Tampilkan data mahasiswa dan nilai rata-rata

Deklarasi mahasiswa dan mata kuliah sebagai berikut:

```
typedef struct simpulMhs Mahasiswa;
typedef struct simpulMK MataKuliah;
struct simpulMhs {
    int NRP;
    char Nama[30];
    char Prodi[15];
    Mahasiswa *nextMhs;
    Mahasiswa *prevMhs;
    MataKuliah *next;
};
struct simpulMK {
    char KodeMK[6];
    char NamaMK[15];
    int Nilai;
    MataKuliah *nextMK;
};
```