

# Doubly Linked List

Dr. Anto Satriyo Nugroho, M.Eng

Email: [asnugroho@gmail.com](mailto:asnugroho@gmail.com)

Web: <http://asnugroho.net/lecture/ds.html>

# Beberapa Jenis Struktur Data

## 1. Array

1. Linear List
2. Stack
3. Queue

## 2. List

1. Linked List
2. Circular List
3. Doubly Linked List
4. Multi list structure

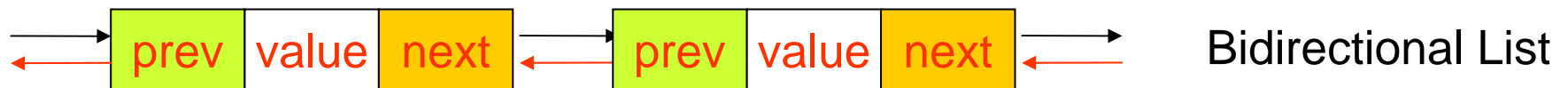
## 3. Tree Structure

# Outline

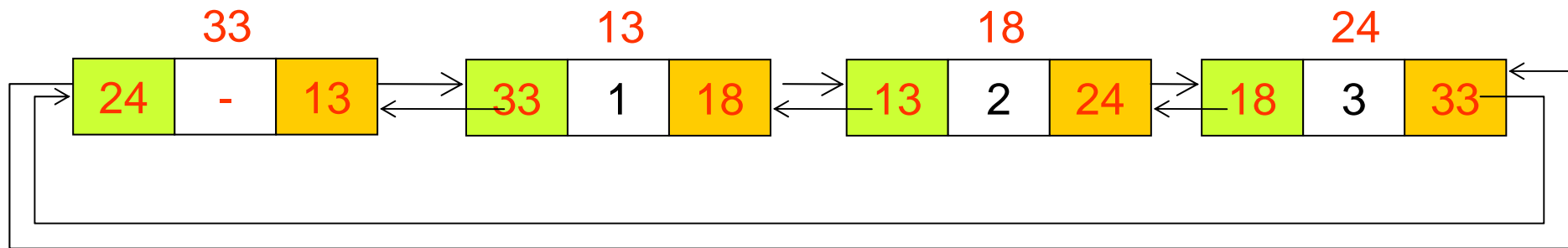
- Review doubly-linked list
- Latihan

# Apakah doubly-linked list itu ?

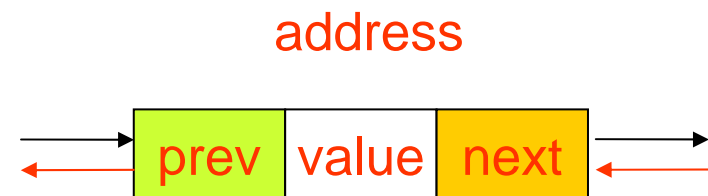
- Semua sel yang terdapat pada list disambungkan dengan pointer, sedangkan tiap sel memiliki TIGA komponen : value, pointer ke sel sebelumnya dan pointer ke sel berikutnya
- Dengan memiliki dua buah pointer ini, maka doubly-linked list dapat diakses dengan DUA arah : ke arah depan dan ke belakang



# Doubly-linked List



```
struct CELL {  
    struct CELL *prev;  
    struct CELL *next;  
    MYDATA value;  
};
```



Misalnya int, char, float, long, double, dsb

# Kelebihan dan kelemahan

## Kelebihan

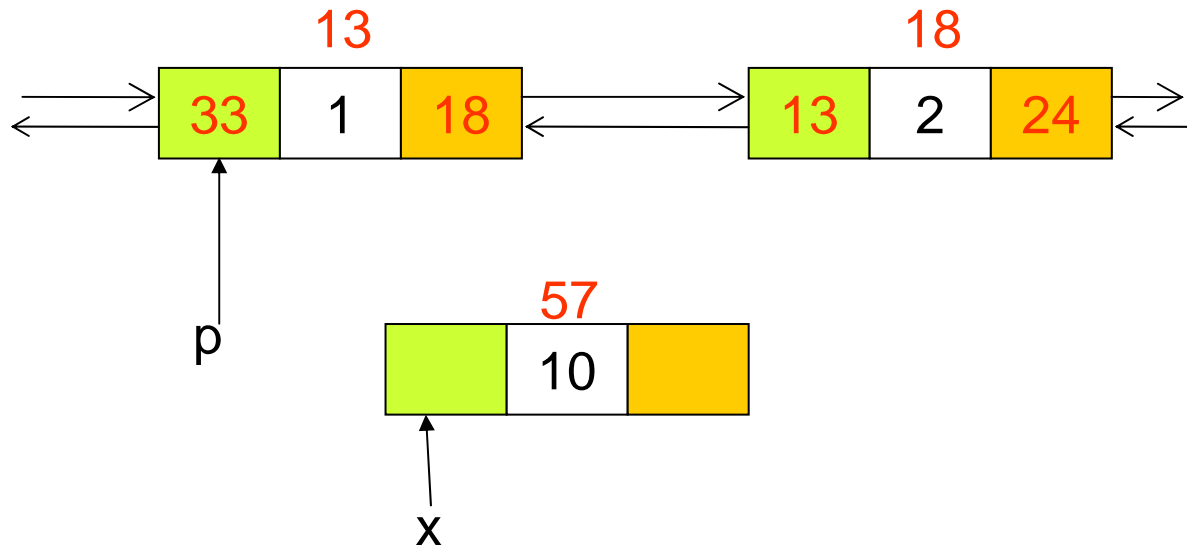
1. List bisa diakses dengan dua arah : ke depan maupun ke belakang
2. Penambahan dan penghapusan data menjadi mudah, karena pada tipe ini kita dapat menambahkan sel baru sesudah maupun sebelum sebuah sel. Demikian juga dalam hal menghapus sel.

## Kelemahan

1. Sebuah data memiliki dua buah pointer, sehingga memerlukan space yang lebih besar

# Cara menambahkan sel baru

Tambahkan sel baru yang ditunjuk oleh pointer x, sesudah suatu sel yang ditunjuk oleh pointer p

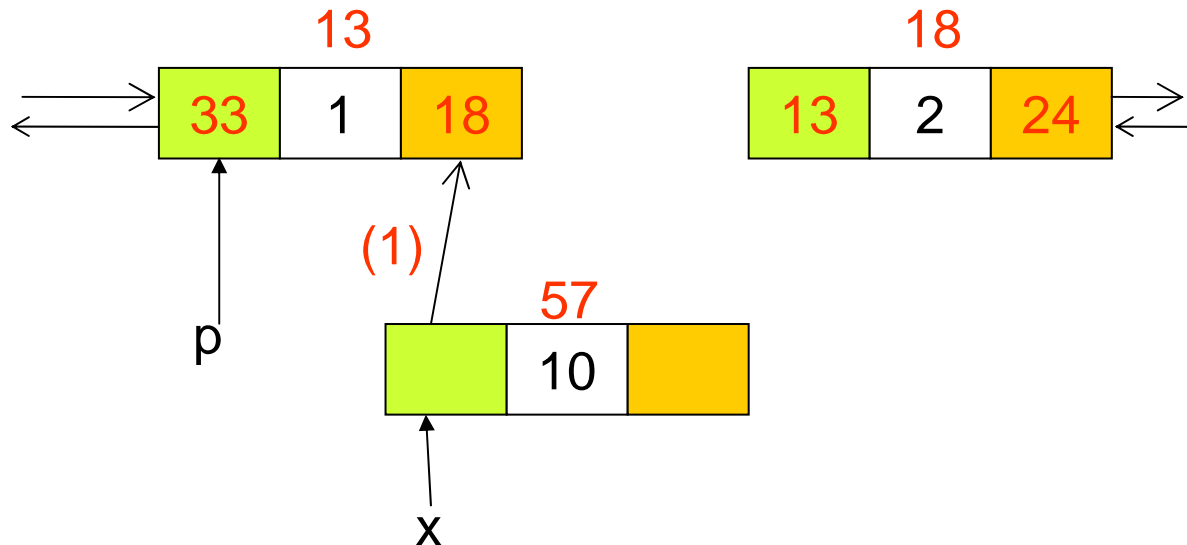


```
x->prev=p;  
x->next=p->next;  
p->next->prev=x;  
p->next=x;
```

- (1) Yang diubah bukan hanya
- (2) pointer next
- (3) tetapi juga pointer prev
- (4)

# Cara menambahkan sel baru

Tambahkan sel baru yang ditunjuk oleh pointer x, sesudah suatu sel yang ditunjuk oleh pointer p

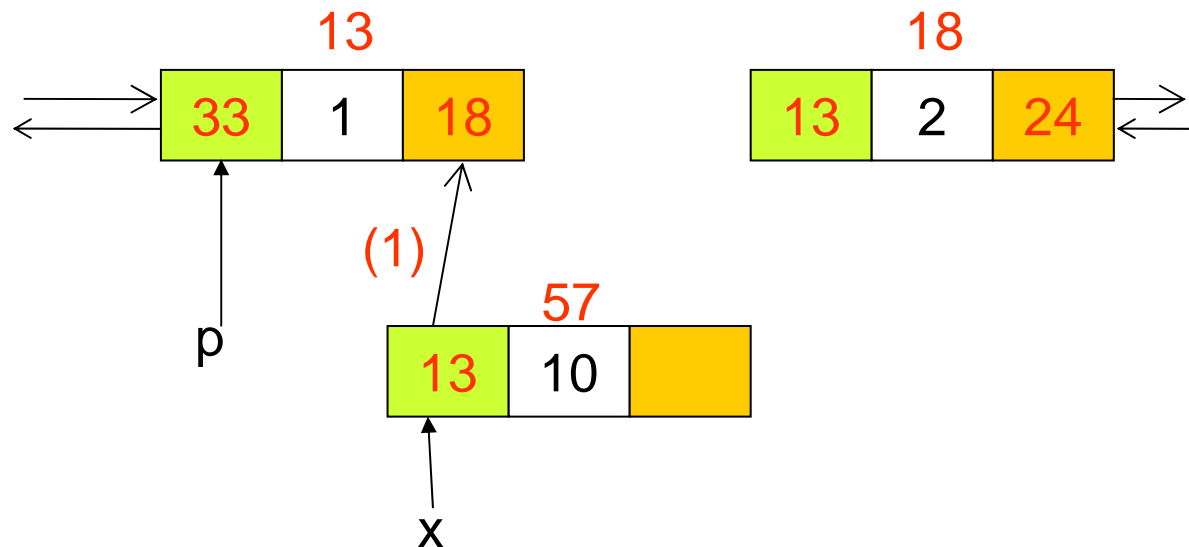


```
x->prev=p;  
x->next=p->next;  
p->next->prev=x;  
p->next=x;
```

- (1) Yang diubah bukan hanya
- (2) pointer next
- (3) tetapi juga pointer prev
- (4)

# Cara menambahkan sel baru

Tambahkan sel baru yang ditunjuk oleh pointer x, sesudah suatu sel yang ditunjuk oleh pointer p

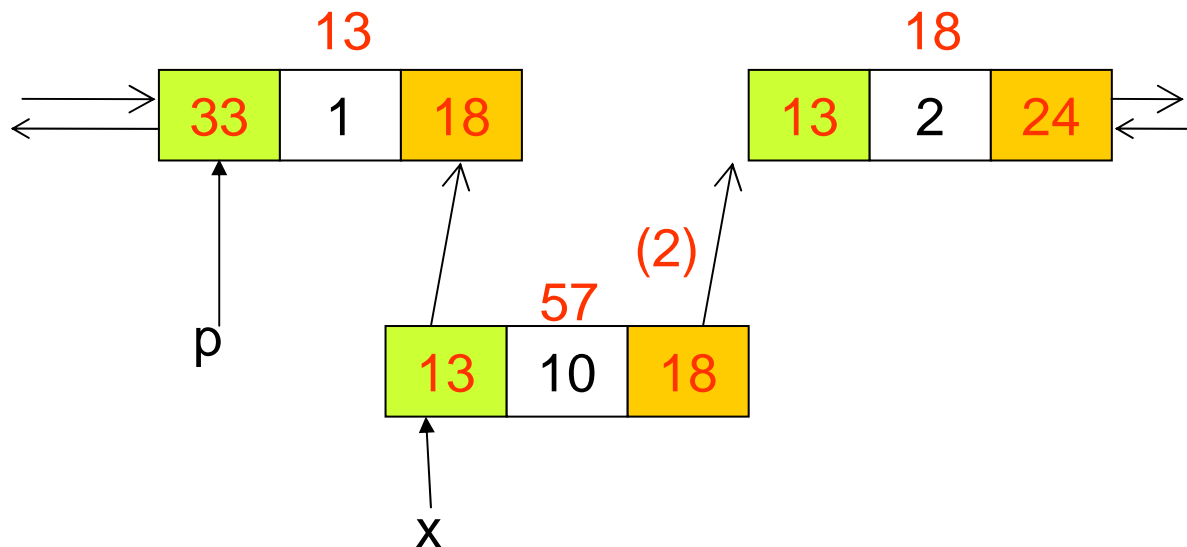


```
x->prev=p;  
x->next=p->next;  
p->next->prev=x;  
p->next=x;
```

- (1) Yang diubah bukan hanya
- (2) pointer next
- (3) tetapi juga pointer prev
- (4)

# Cara menambahkan sel baru

Tambahkan sel baru yang ditunjuk oleh pointer x, sesudah suatu sel yang ditunjuk oleh pointer p

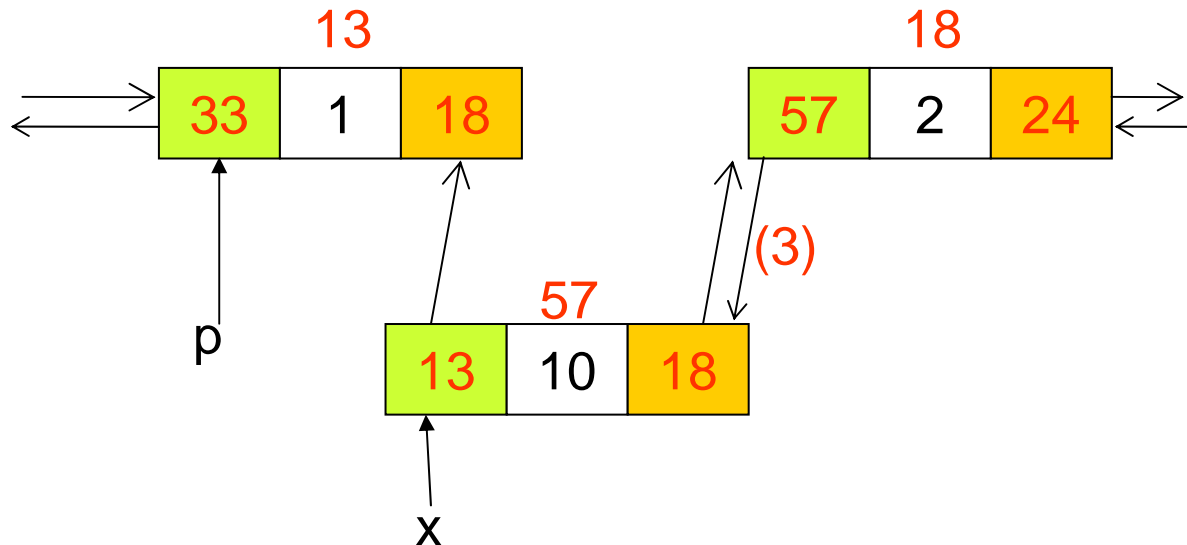


```
x->prev=p;  
x->next=p->next;  
p->next->prev=x;  
p->next=x;
```

- (1) Yang diubah bukan hanya
- (2) pointer next
- (3) tetapi juga pointer prev
- (4)

# Cara menambahkan sel baru

Tambahkan sel baru yang ditunjuk oleh pointer x, sesudah suatu sel yang ditunjuk oleh pointer p



`x->prev=p;`

`x->next=p->next;`

`p->next->prev=x;`

`p->next=x;`

(1)

(2)

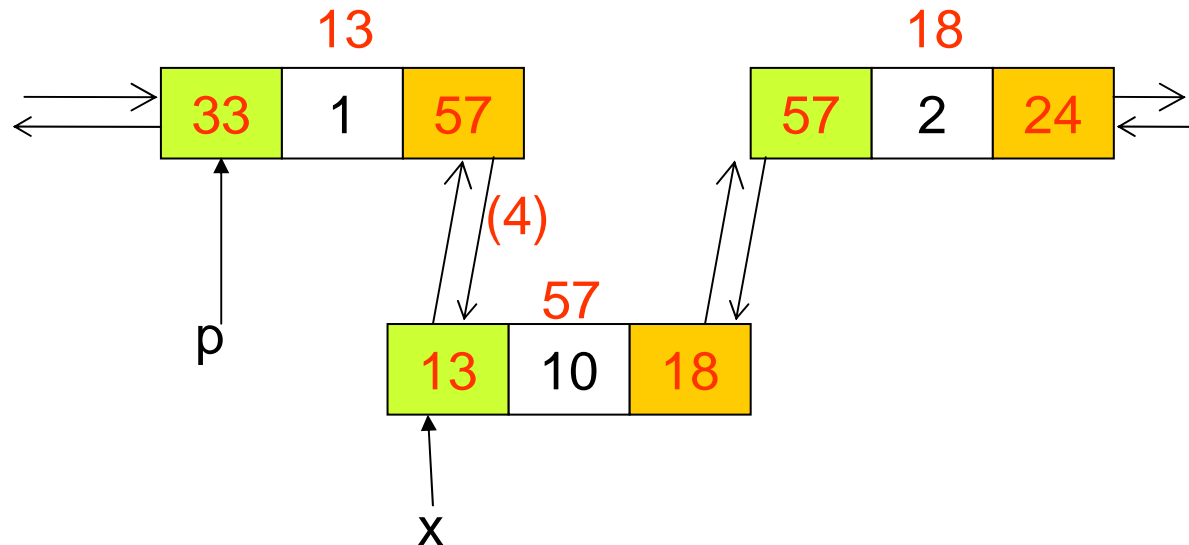
(3)

(4)

Yang diubah bukan hanya pointer next tetapi juga pointer prev

# Cara menambahkan sel baru

Tambahkan sel baru yang ditunjuk oleh pointer x, sesudah suatu sel yang ditunjuk oleh pointer p

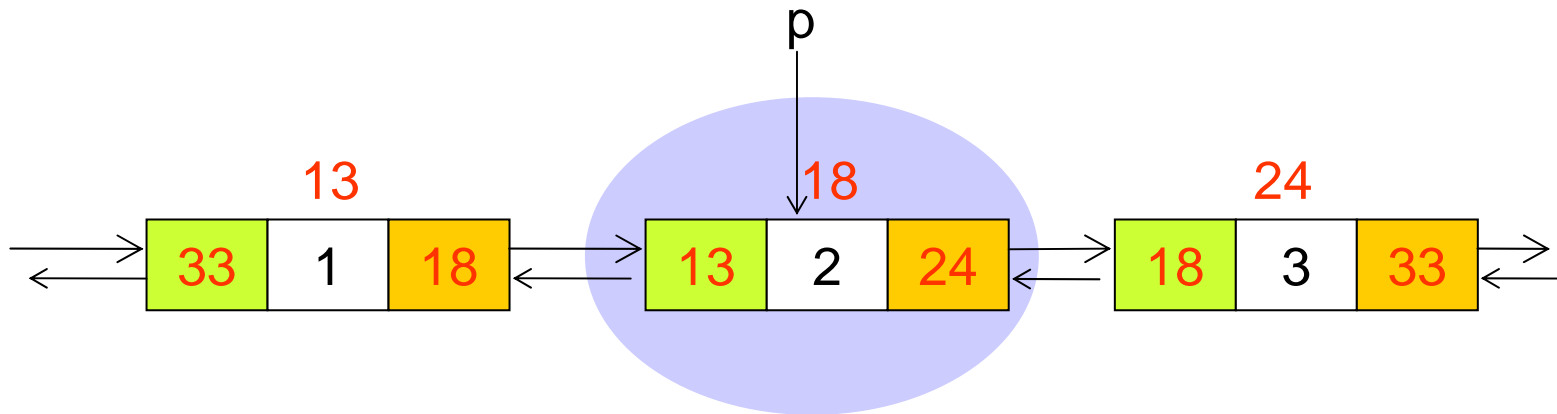


```
x->prev=p;  
x->next=p->next;  
p->next->prev=x;  
p->next=x;
```

- (1) Yang diubah bukan hanya
- (2) pointer next
- (3) tetapi juga pointer prev
- (4)

# Cara menghapus sebuah sel

Hapuslah sel yang ditunjuk pointer p



```
p->prev->next=p->next; (1)
```

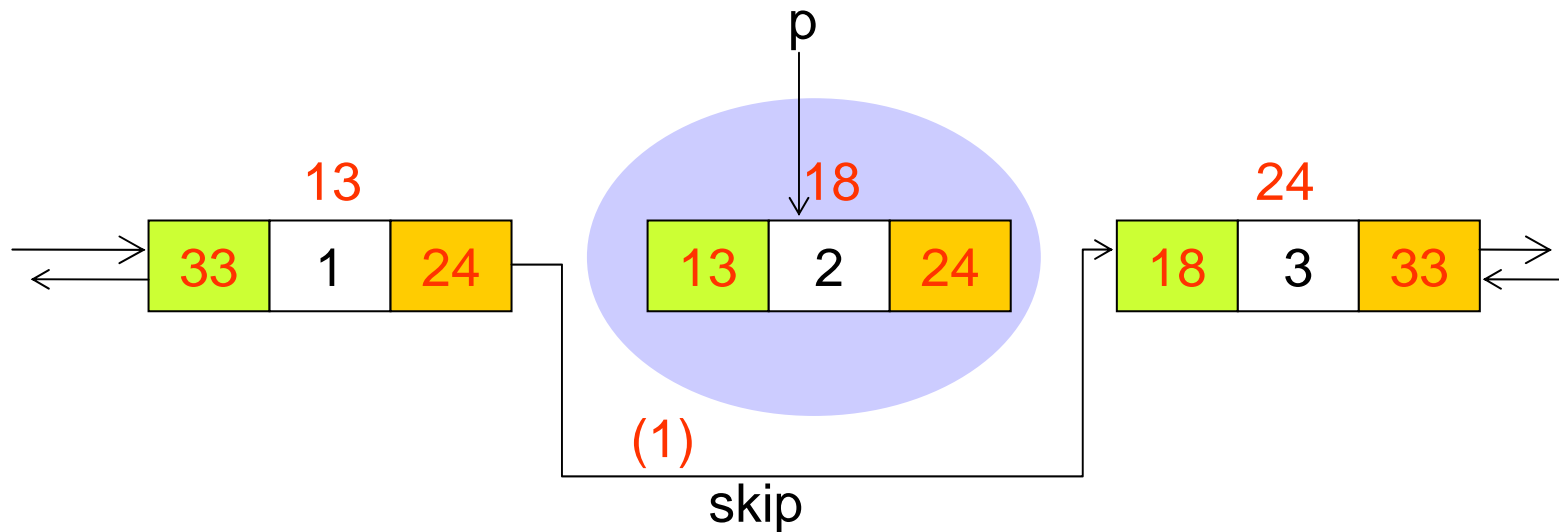
```
p->next->prev=p->prev; (2)
```

```
free(p);
```

Yang diubah bukan hanya pointer next  
tetapi juga pointer prev

# Cara menghapus sebuah sel

Hapuslah sel yang ditunjuk pointer p



```
p->prev->next=p->next; (1)
```

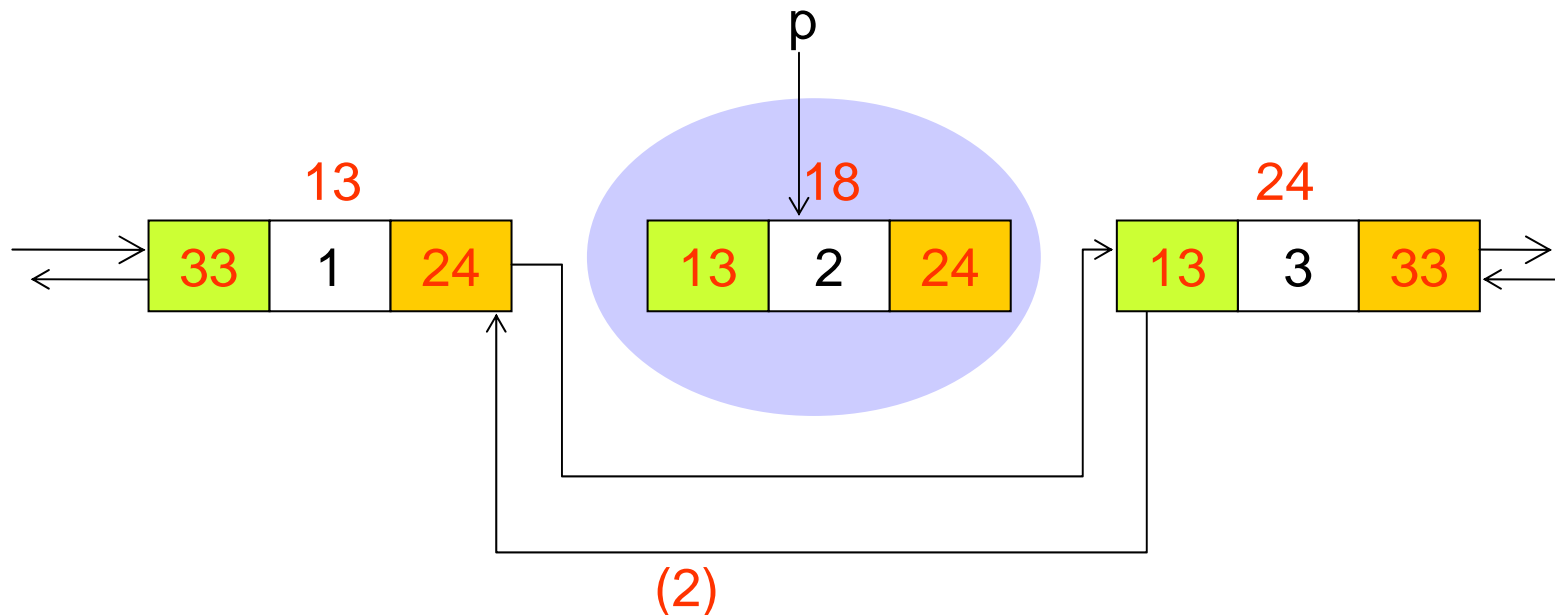
```
p->next->prev=p->prev; (2)
```

```
free(p);
```

Yang diubah bukan hanya pointer next  
tetapi juga pointer prev

# Cara menghapus sebuah sel

Hapuslah sel yang ditunjuk pointer p



```
p->prev->next=p->next; (1)
```

```
p->next->prev=p->prev; (2)
```

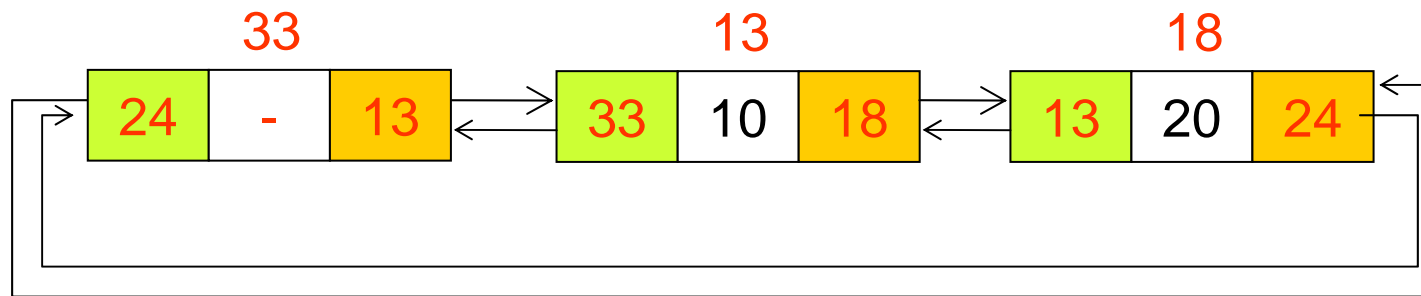
```
free(p);
```

Yang diubah bukan hanya pointer next  
tetapi juga pointer prev

# Latihan pemrograman

1. Downloadlah doublyll.c dari <http://asnugroho.net/lecture/ds.html>
2. Selesaikan fungsi `cell_insert()`
3. Fungsi `cell_insert()` adalah menambahkan databaru pada sebuah doubly linked list, dimana data harus dalam kondisi terurutkan. Dimulai dari data dengan nilai kecil, dan semakin ke belakang semakin besar.

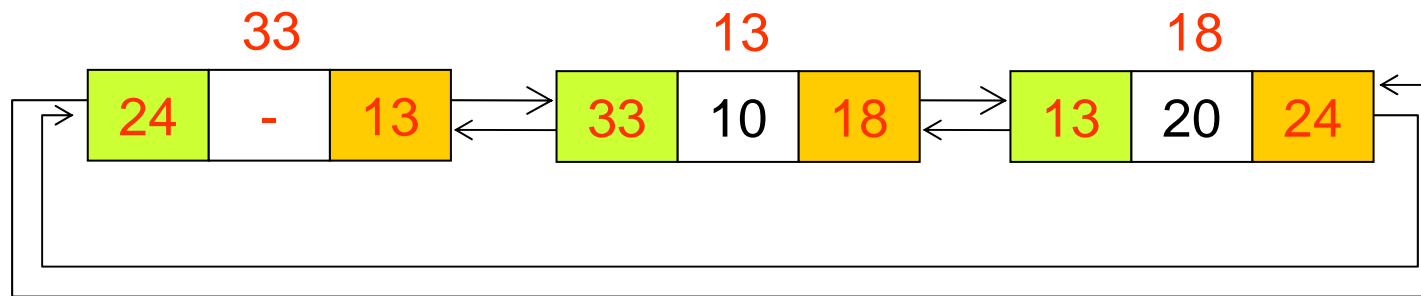
# Doubly-linked List



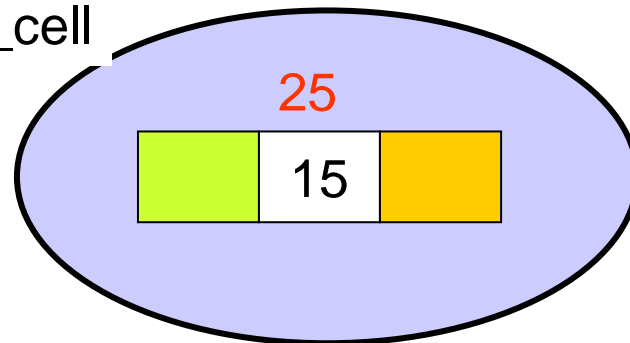
```
struct CELL {  
    struct CELL *prev;  
    struct CELL *next;  
    int value;  
};
```



# Doubly-linked List

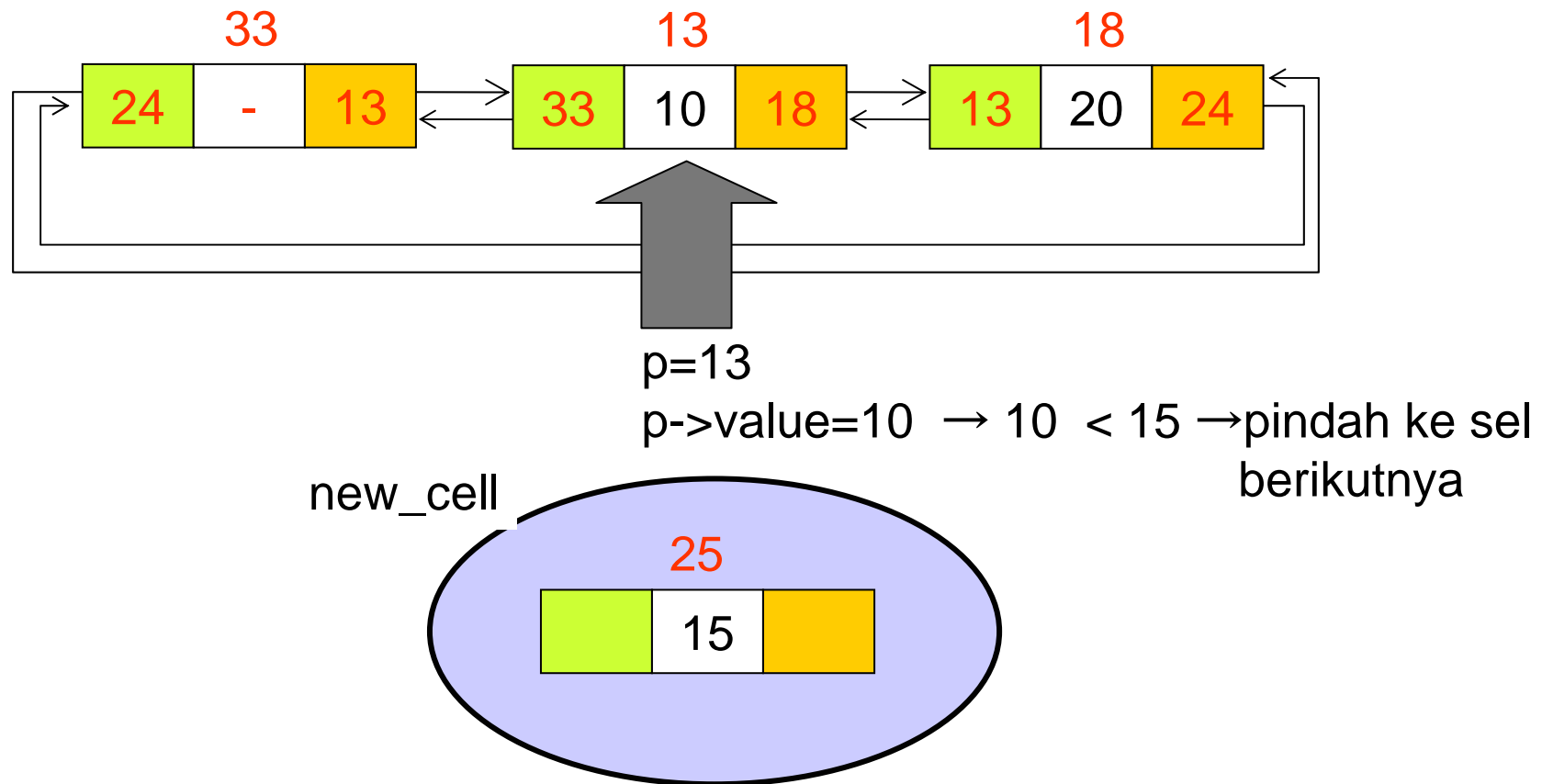


new\_cell

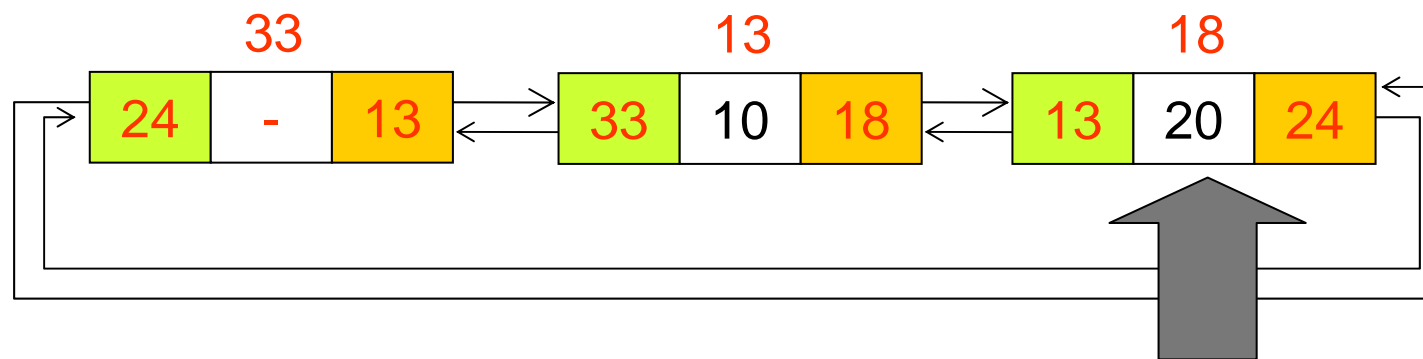


Sel ini akan ditambahkan ke list

# Doubly-linked List



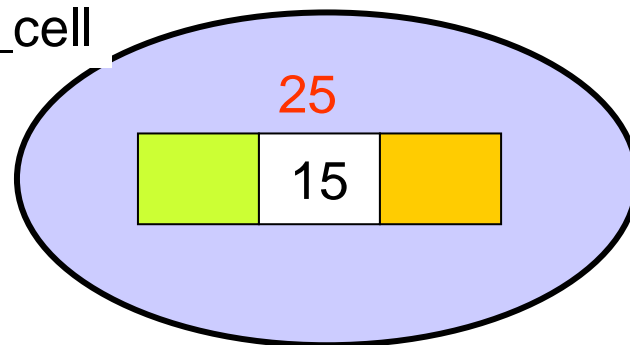
# Doubly-linked List



p=18

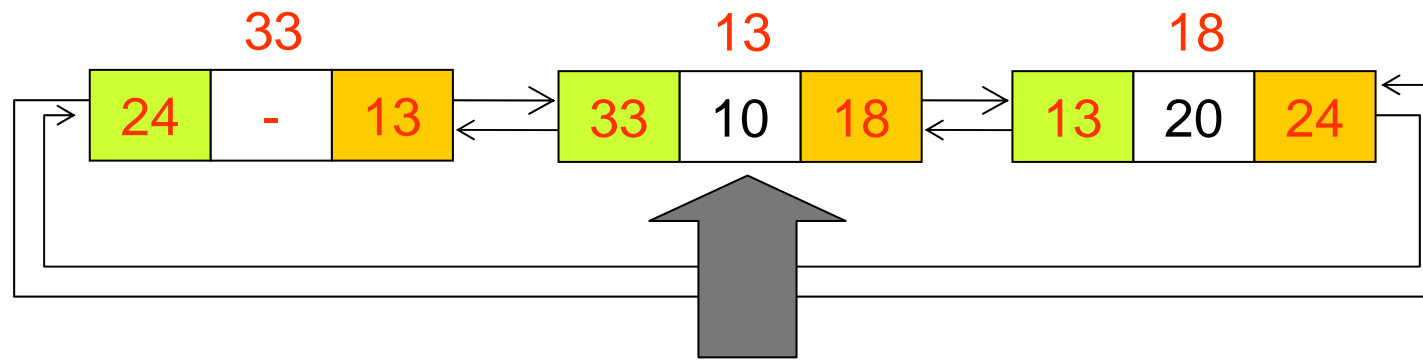
p->value=20 → 20 > 15

new\_cell



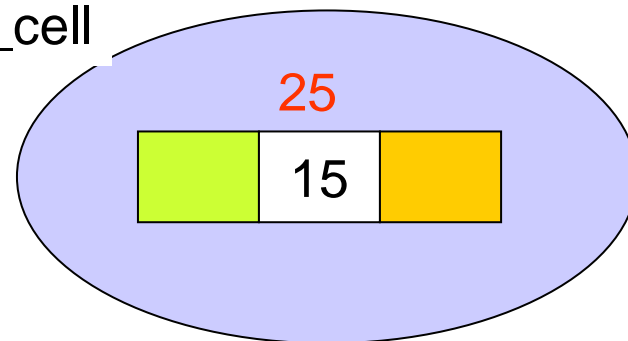
pindah ke sel sebelumnya

# Doubly-linked List

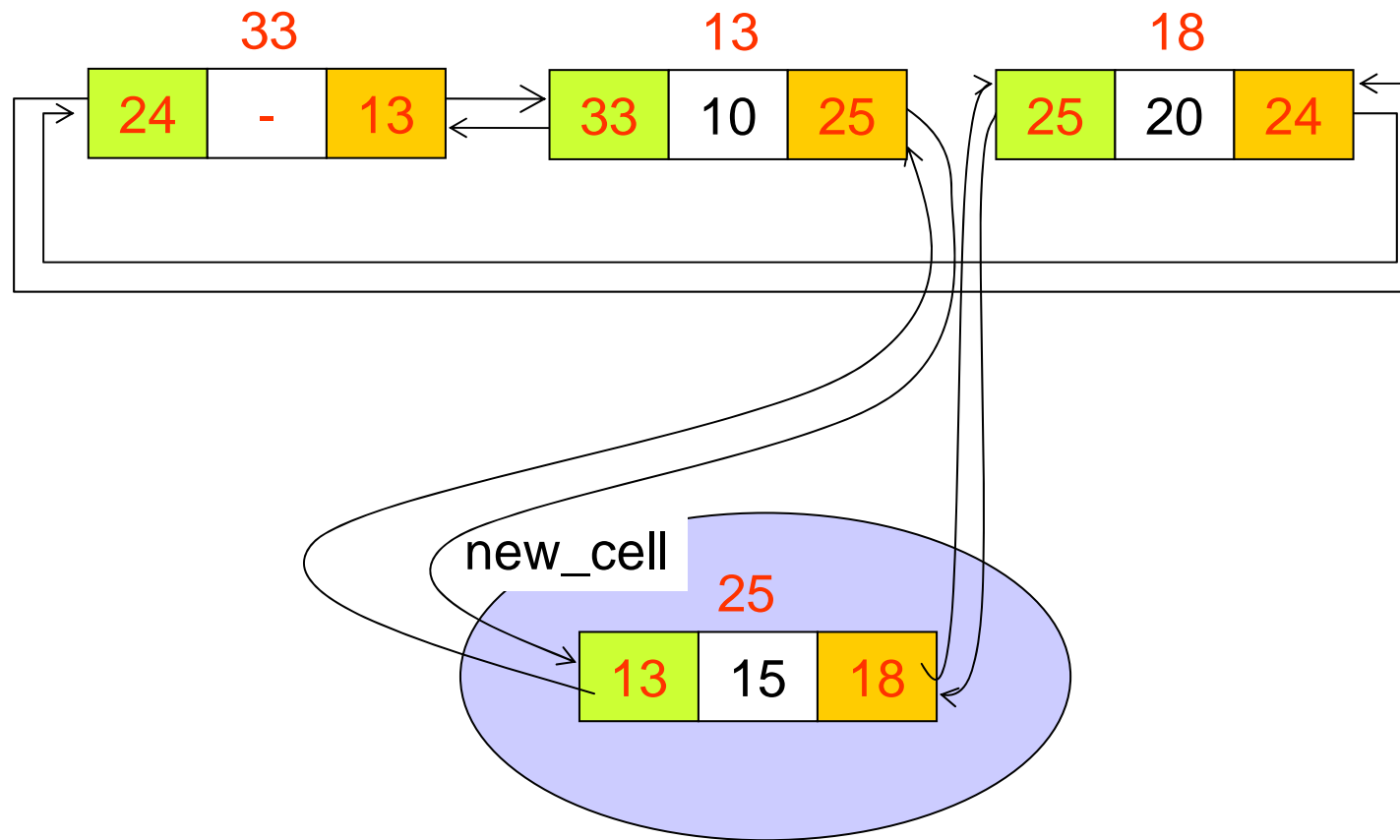


p=13  
tambahkan sel baru sesudah sel yang ditunjuk

new\_cell



# Doubly-linked List



# Doubly-linked List

```
void cell_insert(int a)
{
    cell *p, *new_cell;
```

① Mencari posisi dimana new\_cell harus dimasukan

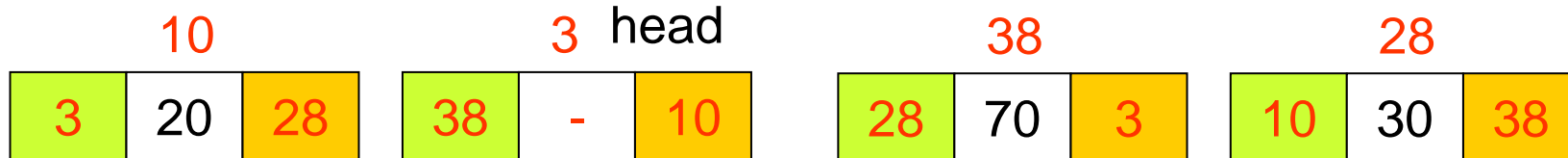
② Alokasi memory untuk new\_cell memakai malloc

③ Tambahkan new\_cell pada posisi yang ditentukan

```
}
```

# Latihan pemrograman

1. Gambarkan bidirectional list berikut



2. Tambahkan sel berikut setelah sel yang berisi value "20"



3. Gambarkan kondisi bidirectional list, setelah sel yang berisi value "70" dihapus. Kerjakan soal ini terhadap list yang diperoleh dari no.2 di atas.