

자료구조

L04 Lists

2022년 1학기

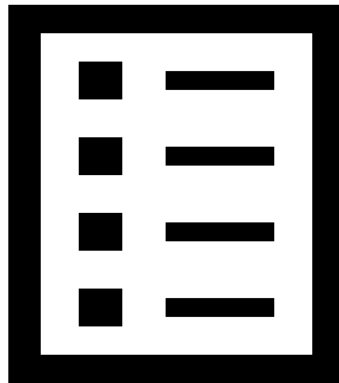
국민대학교 소프트웨어학부

Summary

- ❖ 리스트 (Lists)
- ❖ 배열(Array) 기반 리스트 구현
- ❖ 연결(Link) 기반 리스트 구현
- ❖ Freelist

리스트 (Lists)

- 유한하고 순서가 있는 데이터의 나열
- 중요한 개념: 리스트 속의 데이터들은 위치 **Position**가 있음
- 표기법: $\langle a_0, a_1, \dots, a_{n-1} \rangle$
- 어떤 연산 **Operation**이 필요할까?



리스트에 필요한 연산, 리스트 ADT

- 특정 위치에 아이템을 넣을(insert, append) 수 있다.
- 특정 위치의 아이템을 읽고(getValue), 변경(update)할 수 있다.
- 특정 위치의 아이템을 삭제(remove)할 수 있다.
- 리스트 내의 **아이템 개수(length)**를 알 수 있다.
- 리스트를 **비울(clear)** 수 있다.

```
public interface List<E> {  
    public void clear();  
    public void insert(int pos, E item);  
    public void append(E item);  
    public void update(int pos, E item);  
    public E getValue(int pos);  
    public E remove(int pos);  
    public int length();  
}
```

리스트 ADT 예제

- **List L = $\langle 12, 32, 15 \rangle$**
- **L.insert(1, 99) $\Rightarrow \langle 12, 99, 32, 15 \rangle$**
- **L.remove(3) $\Rightarrow \langle 12, 99, 32 \rangle$**
- **L.update(0, 61) $\Rightarrow \langle 61, 99, 32 \rangle$**
- **L.length() $\Rightarrow 3$**
- **L.clear() $\Rightarrow \langle \rangle$**

Summary

- ❖ 리스트 (Lists)
- ❖ 배열(Array) 기반 리스트 구현
- ❖ 연결(Link) 기반 리스트 구현
- ❖ Freelist

배열(Array) 기반 리스트, 넣기

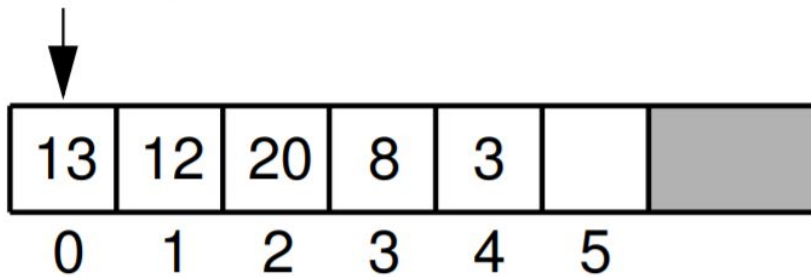
- **List L** = $\langle 13, 12, 20, 8, 3 \rangle$
- **L.insert(0, 23)** $\Rightarrow \langle 23, 13, 12, 20, 8, 3 \rangle$
- 배열로 리스트를 어떻게 표현할까?

13	12	20	8	3		
0	1	2	3	4	5	

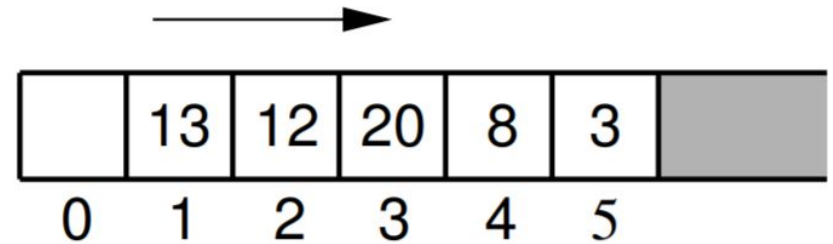
- 23을 배열의 맨 앞에 넣으려면...?

배열(Array) 기반 리스트, 넣기

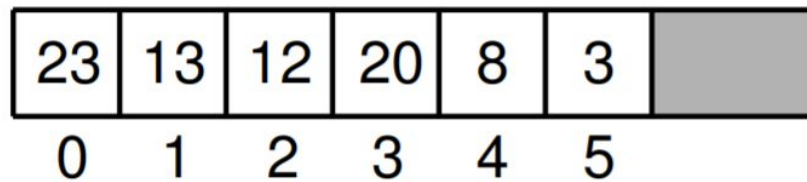
Insert 23:



(a)



(b)



(c)

배열(Array) 기반 리스트 구현

- ArrayList의 자료구조 및 생성자

```
public class ArrayList<E> implements List<E> {  
  
    private static final int defaultSize = 10;  
    private int listSize;  
    private E[] data;  
  
    public ArrayList() {  
        this(defaultSize);  
    }  
  
    public ArrayList(int size) {  
        listSize = 0;  
        data = (E[]) new Object[size];  
    }  
}
```

배열(Array) 기반 리스트 구현

- ArrayList의 clear, update, getValue, length, append

```
public void clear() { listSize = 0; }
```

```
public void update(int pos, E item) { data[pos] = item; }
```

```
public E getValue(int pos) { return data[pos]; }
```

```
public int length() { return listSize; }
```

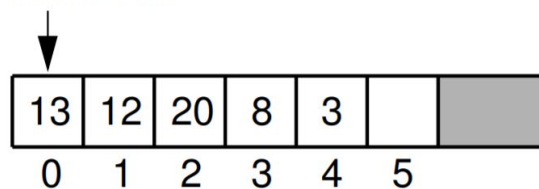
```
public void append(E item) { data[listSize++] = item; }
```

배열(Array) 기반 리스트 구현

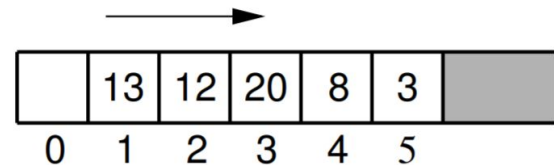
- ArrayList의 insert

```
public void insert(int pos, E item) {  
    for (int i=listSize; i > pos; i--)  
        data[i] = data[i-1];  
  
    data[pos] = item;  
    listSize++;  
}
```

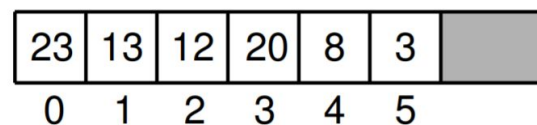
Insert 23:



(a)



(b)

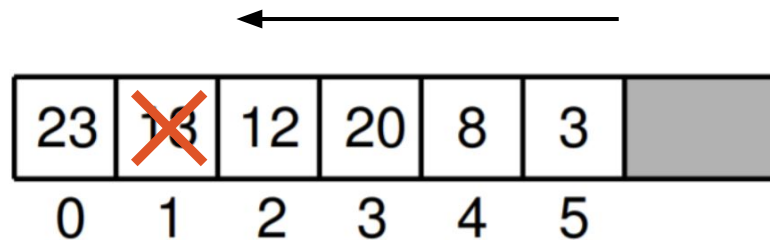


(c)

배열(Array) 기반 리스트 구현

- ArrayList의 **remove**

```
public E remove(int pos) {  
    E ret = data[pos];  
  
    for(int i=pos; i<listSize-1; i++)  
        data[i] = data[i+1];  
  
    listSize--;  
  
    return ret;  
}
```



배열(Array) 기반 리스트, 장단점

- 장점

- 특정 위치의 데이터 조회가 빠르다

- 단점

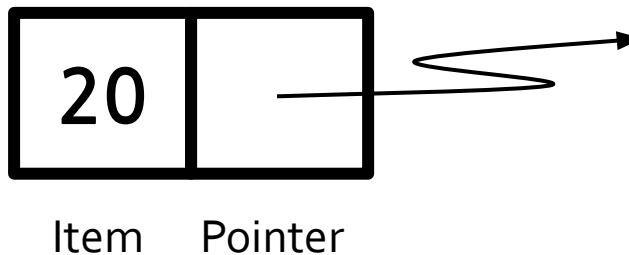
- 데이터의 추가, 삭제의 비용이 크다
- 크기가 고정되어 있다
 - 넣으려는 데이터가 더 많으면... Error
 - 넣으려는 데이터가 너무 적으면... 메모리 낭비
 - 해결법?

Summary

- ❖ 리스트 (Lists)
- ❖ 배열(Array) 기반 리스트 구현
- ❖ 연결(Link) 기반 리스트 구현
- ❖ Freelist

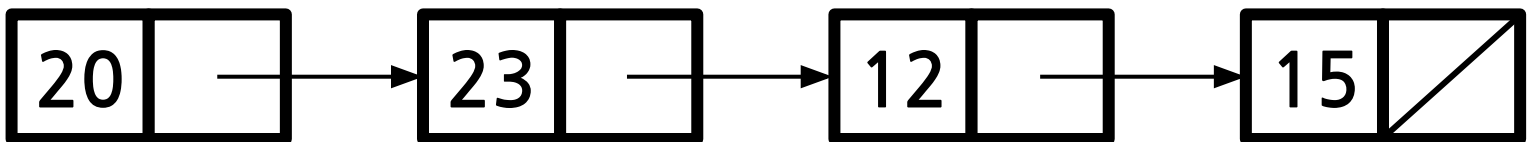
연결(Link)과 연결리스트(Linked List)

- 연결은 데이터 **Item**와 포인터 **Pointer**로 구성된다.



- 리스트를 연결을 이용해 표현하는 방법은?

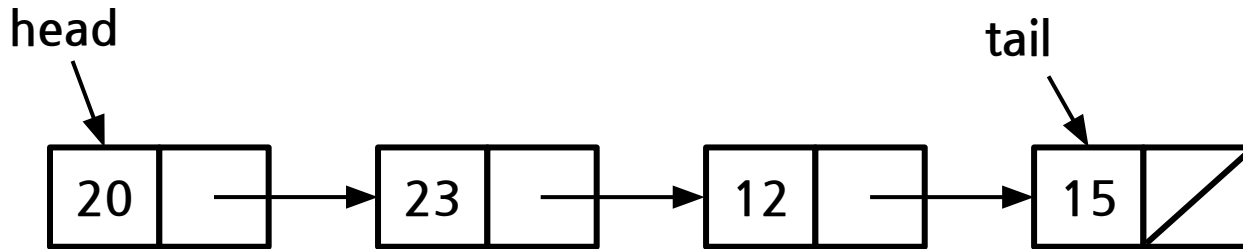
<20, 23, 12, 15>



연결 Class

```
class Link<E> {  
    private E item;  
    private Link<E> next;  
  
    public Link(E item, Link<E> next) {  
        this.item = item;  
        this.next = nextval;  
    }  
  
    Link<E> next() { return next; }  
  
    Link<E> setNext(Link<E> next) { return this.next = next; }  
  
    E item() { return item; }  
  
    E setItem(E item) { return this.item = item; }  
}
```


연결 in Java

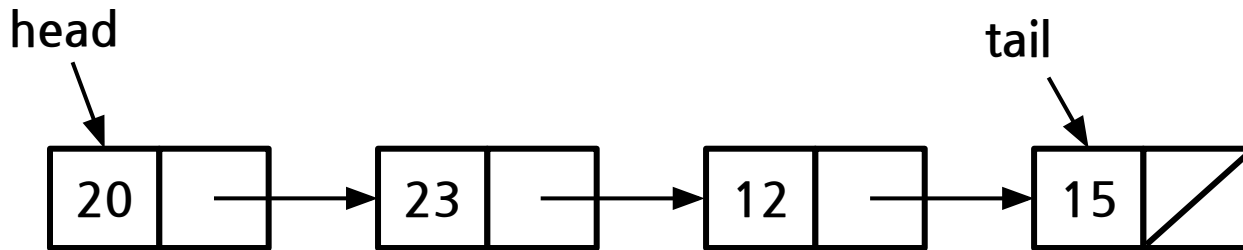


```
Link<Integer> l15 = new Link<>(15, null);  
Link<Integer> l12 = new Link<>(12, l15);  
Link<Integer> l23 = new Link<>(23, l12);  
Link<Integer> l20 = new Link<>(20, l23);  
Link<Integer> head = l20;  
Link<Integer> tail = l15;
```

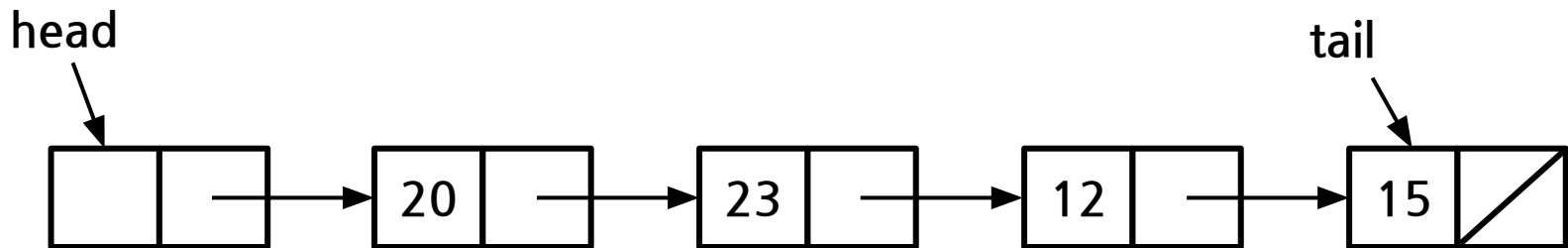
```
System.out.println(head.next().next().item());
```

연결리스트 구현 이슈

- head를 따로 둘까? 말까?



VS



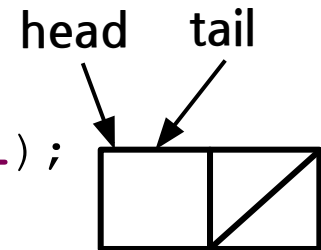
Linked List Class

```
public class LinkedList<E> implements List<E>{
```

```
    private Link<E> head, tail;  
    int size;
```

```
    public LinkedList() {  
        head = tail = new Link<>(null, null);  
        size = 0;  
    }
```

```
    @Override  
    public void clear() {  
        head.setNext(null);  
        size = 0;  
    }
```



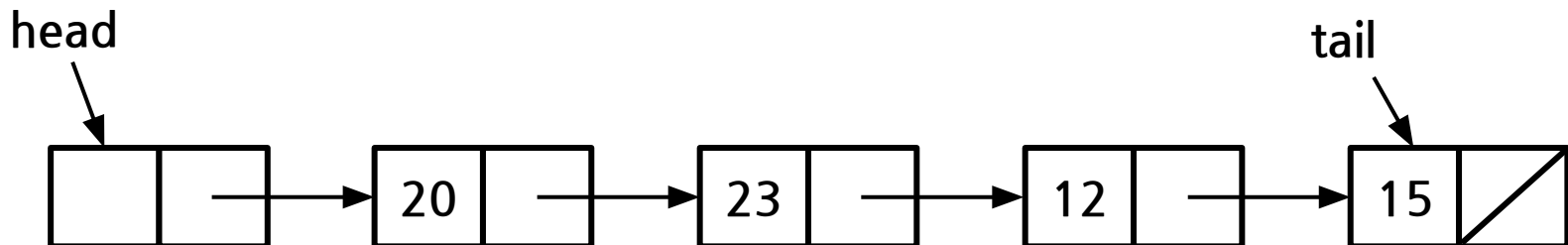
Linked List Class: update/getValue

@Override

```
public void update(int pos, E item) {  
    Link<E> curr = head;  
    for(int i=0; i<pos; i++) curr = curr.next();  
    curr.setItem(item);  
}
```

@Override

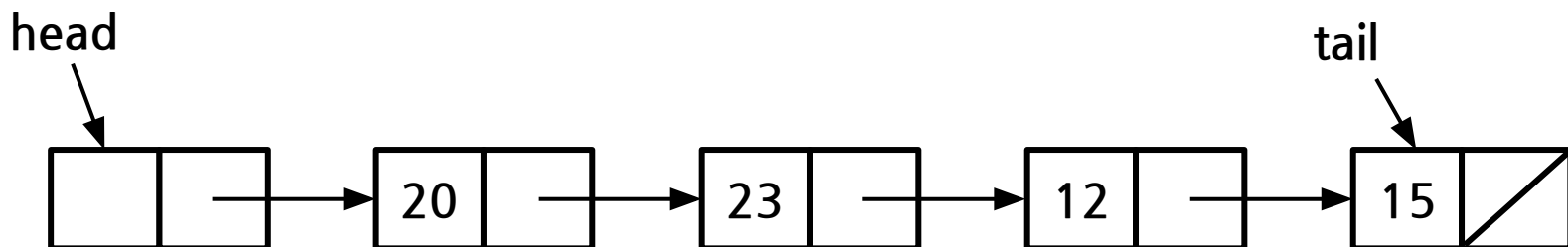
```
public E getValue(int pos) {  
    Link<E> curr = head;  
    for(int i=0; i<pos; i++) curr = curr.next();  
    return curr.item();  
}
```



Linked List Class: length/append

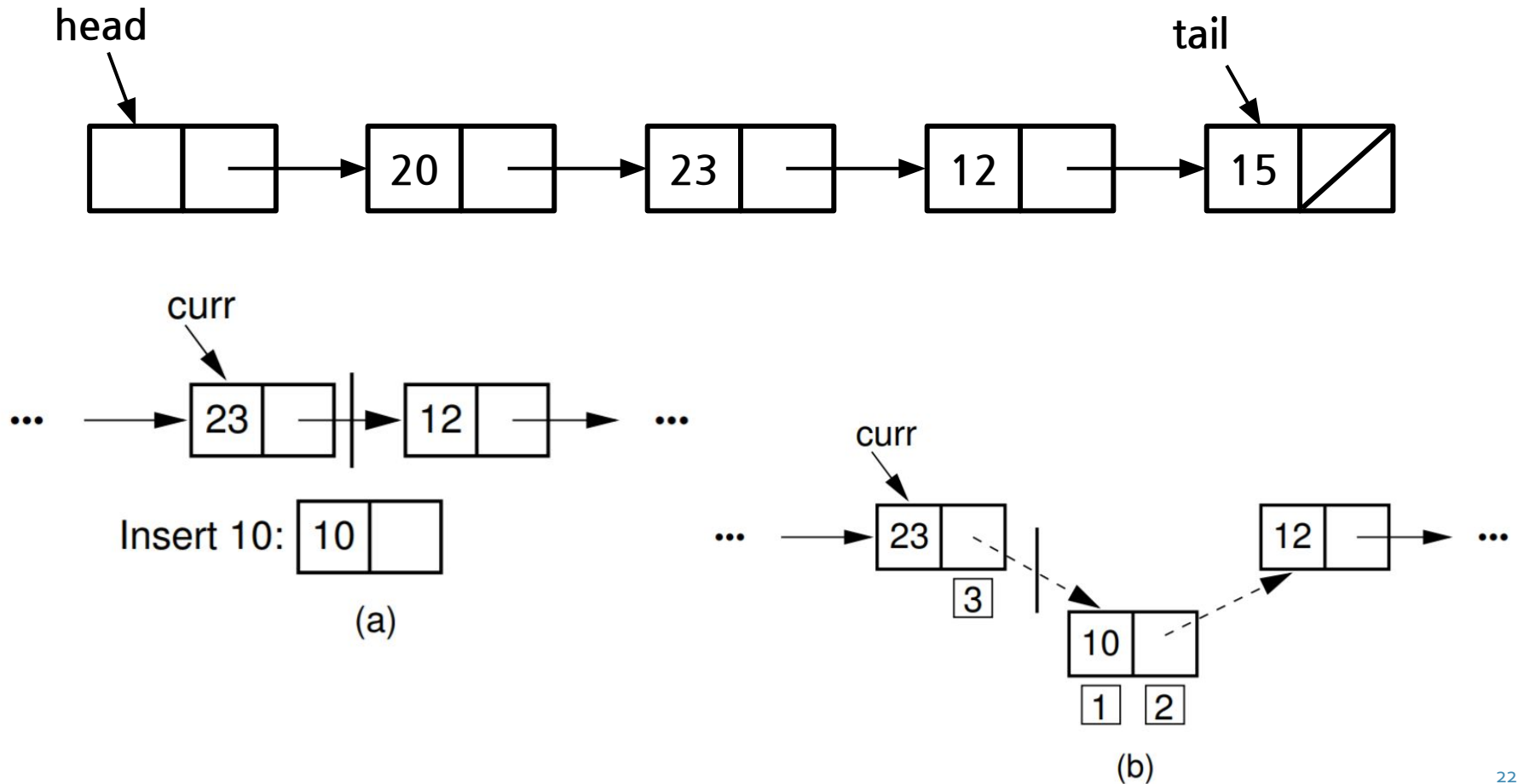
```
@Override  
public int length() {  
    return size;  
}
```

```
@Override  
public void append(E item) {  
    tail = tail.setNext(new Link<>(item, null));  
    size++;  
}
```



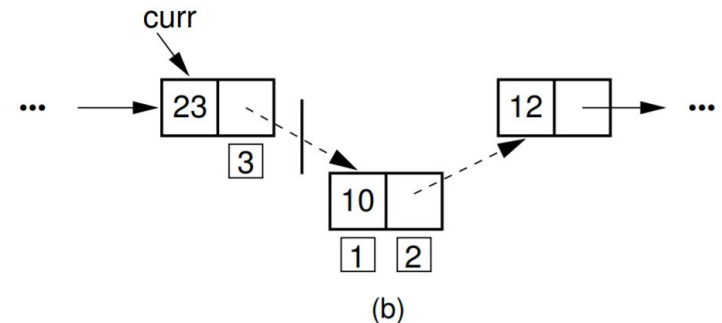
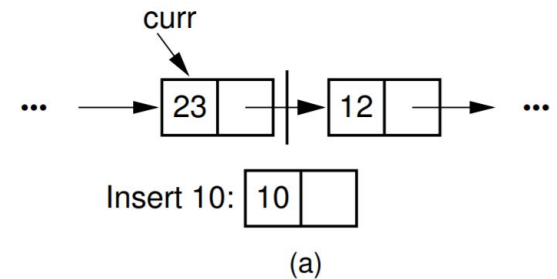
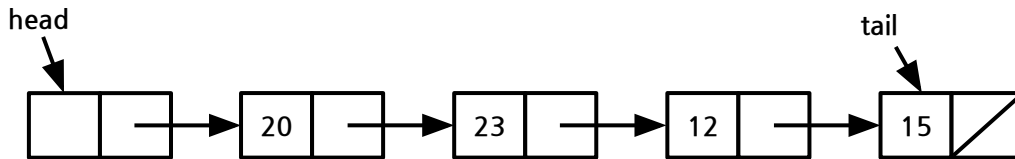
Linked List Class: insert

- 23과 12 사이에 10 넣기



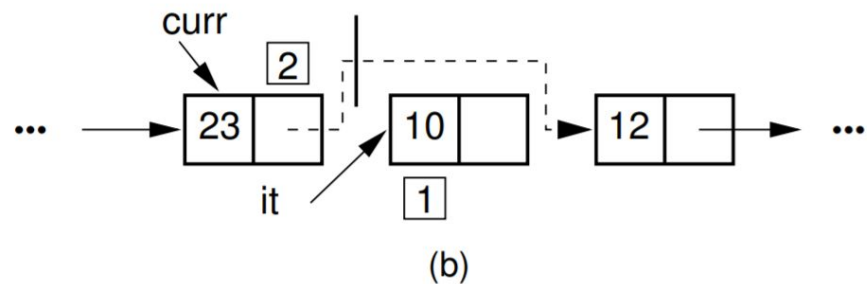
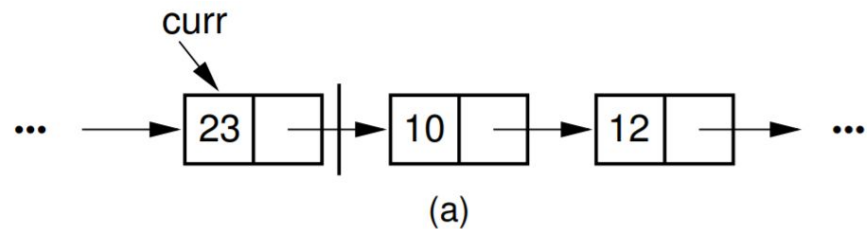
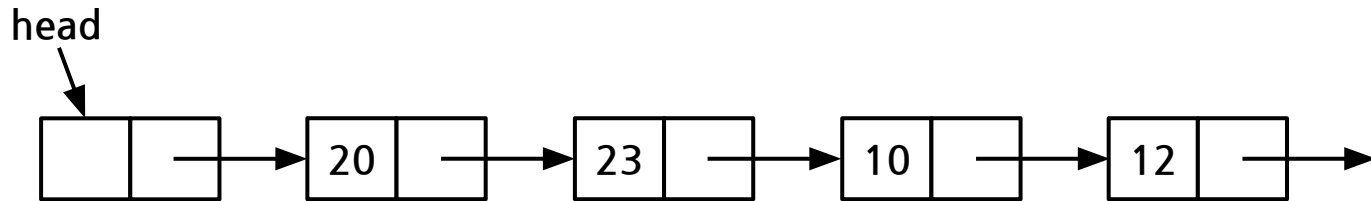
Linked List Class: insert

```
public void insert(int pos, E item) {  
    Link<E> curr = head;  
    for(int i=0; i<pos; i++)  
        curr = curr.next();  
  
    curr.setNext(new Link<>(item, curr.next()));  
  
    if(curr == tail)  
        tail = curr.next();  
  
    size++;  
}
```



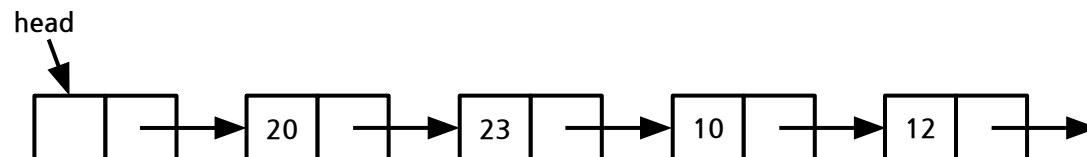
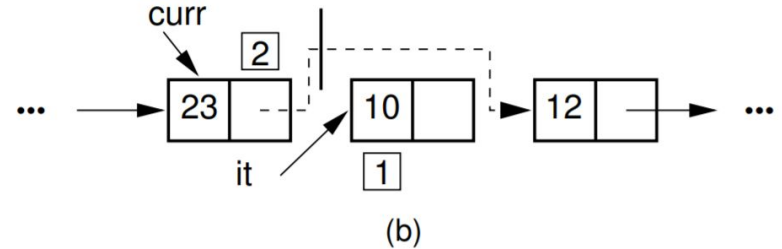
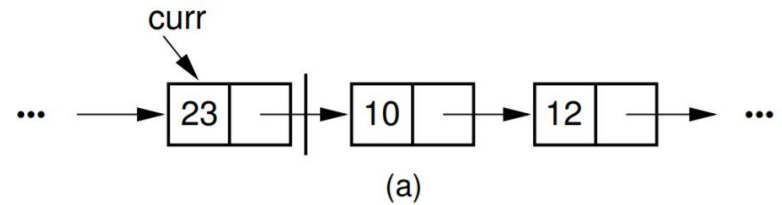
Linked List Class: remove

- 2번 index에 있는 값 (10) 지우기



Linked List Class: remove

```
public E remove(int pos) {  
    Link<E> curr = head;  
    for(int i=0; i<pos; i++)  
        curr = curr.next();  
  
    E ret = curr.next().item();  
  
    if(curr.next() == tail)  
        tail = curr;  
  
    curr.setNext(curr.next().next());  
  
    size--;  
  
    return ret;  
}
```



Comparison of Implementations

	Array Based List	Linked List
넣기/빼기		
맨 앞에 넣기/빼기		
맨 뒤에 넣기/빼기		
메모리 오버헤드 Space overhead		

When should we use an array based list or a linked list?

Space Comparison

Break-even point:

$$DE = n(P + E)$$
$$n = \frac{DE}{P + E}$$

D: 최대 데이터 수 (array 크기)

n: 데이터 개수 (linked-list에 있는 데이터 개수)

E: item 하나에 필요한 메모리 양

P: pointer 하나에 필요한 메모리 양

Outline

- ❖ Introduction to Lists
- ❖ Array-based List Implementation
- ❖ Link-based List Implementation
- ❖ **Freelist**

Freelists

System **new** and garbage collection are slow.

- Add freelist support to the Link class.

Link Class Extensions

```
static Link<?> freelist = null;

static <E> Link<E> get(E item, Link<E> next) {
    if (freelist == null)
        return new Link<>(item, next);

    Link<E> ret = (Link<E>) Link.freelist;
    freelist = freelist.next();

    ret.setItem(item);
    ret.setNext(next);

    return ret;
}

void release() {
    item = null;
    next = (Link<E>) Link.freelist;
    freelist = this;
}
```

Using Freelist

`new Link<E>(item, next);` → `Link.get(item, next);`

```
public E remove(int pos) {
    Link<E> curr = head;
    for(int i=0; i<pos; i++)
        curr = curr.next();

    E ret = curr.next().item();
    if(curr == tail.next()) tail = curr;

    curr.setNext(curr.next().next());

    size--;
    return ret;
}
```

```
public E remove(int pos) {
    Link<E> curr = head;
    for(int i=0; i<pos; i++)
        curr = curr.next();

    E ret = curr.next().item();
    if(curr == tail.next()) tail = curr;

    Link<E> tmp = curr.next();
    curr.setNext(curr.next().next());
    tmp.release();

    size--;
    return ret;
}
```

Outline

- ❖ Introduction to Lists
- ❖ Array-based List Implementation
- ❖ Link-based List Implementation
- ❖ Freelist

Questions?