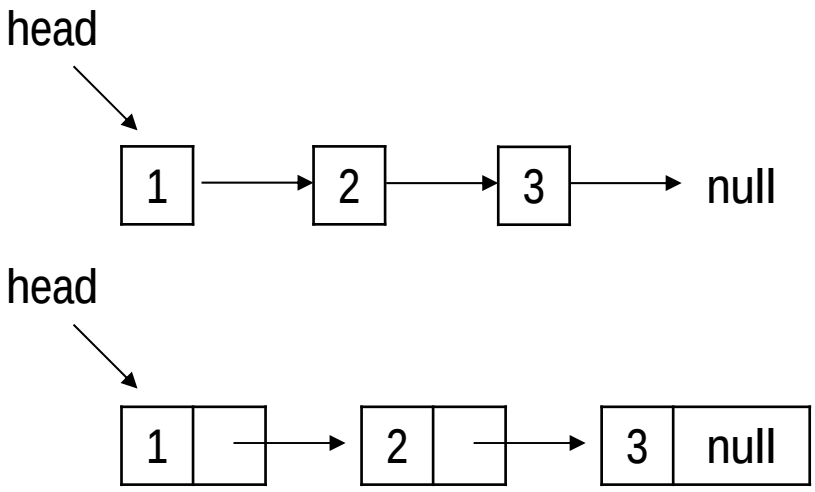
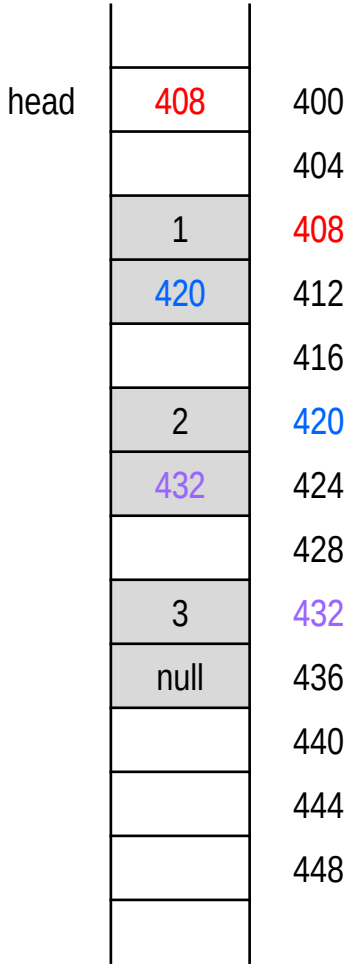
연결리스트

연결리스트: 개념적 도식

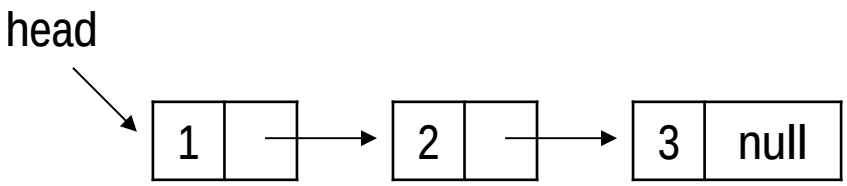

head
1
2
3
null
head
1
2
3
null


head
408
400
404
1
408
420
412
416
2
420
432
424
428
3
432
null
436
440
444
448

연결리스트: 생성

```
public class Node {
        int    data;
        Node       next;
        public Node(int data) {
                this.data=data;
        }
}
```

```
public class Test {
        public static void main(String[] args) {
                Node       head=null;
                Node       n1=new Node(1);
                Node       n2=new Node(2);
                Node       n3=new Node(3);
                head=n1;
                n1.next=n2;
                n2.next=n3;
        }
}
```

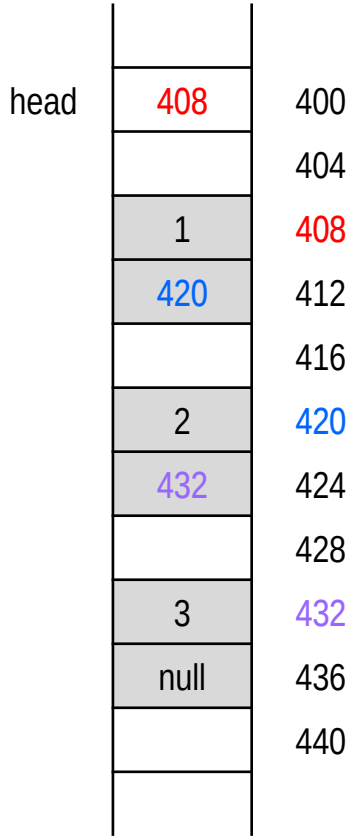


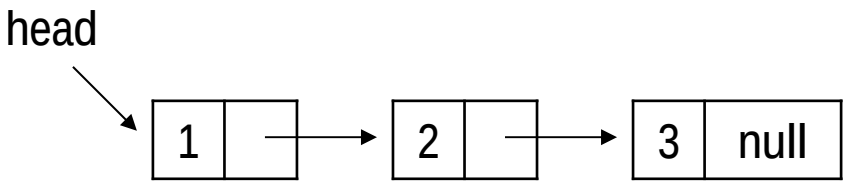
head	408	400
		404
	1	408
	420	412
		416
	2	420
	432	424
		428
	3	432
	null	436
		440
n1	408	444
n2	420	448
n3	432	452

연결리스트: 생성

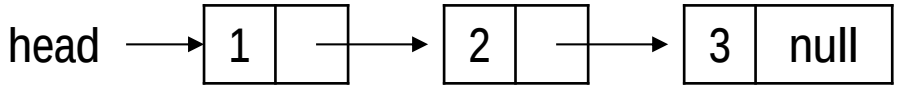
```
public class Node {
        int    data;
        Node       next;
        public Node(int data) {
                this.data=data;
        }
}
```

```
public class Test {
        public static void main(String[] args) {
                Node       head=null;
                head=new Node(1);
                head.next=new Node(2);
                head.next.next=new Node(3);
        }
}
```



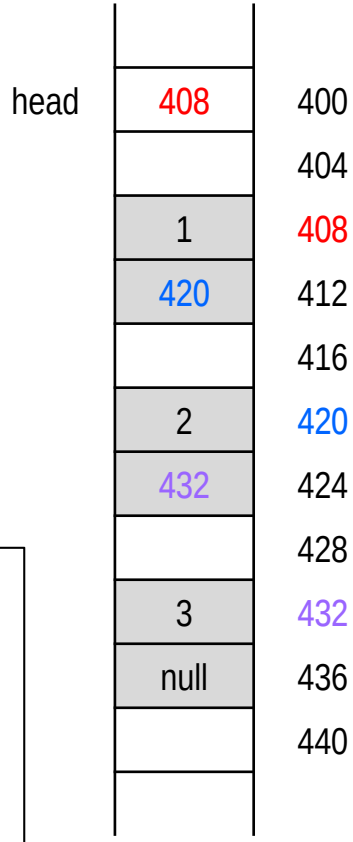


연결리스트: 출력

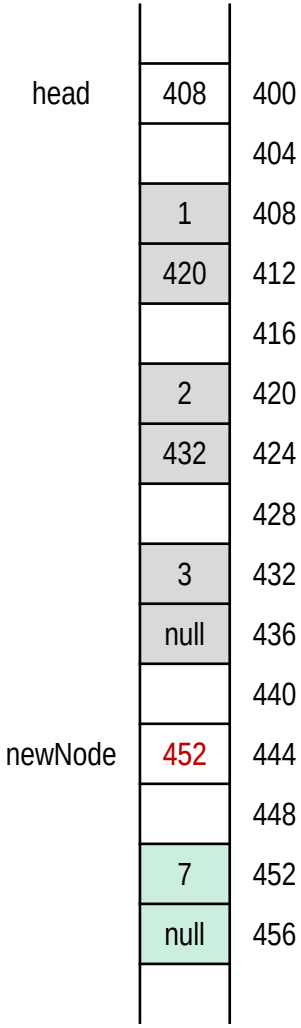


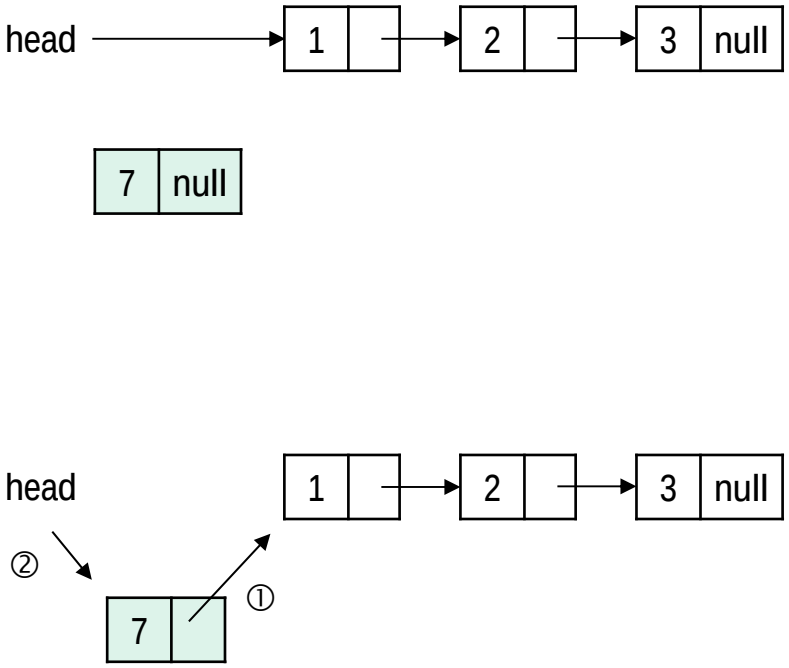

```
public class Node {
        int    data;
        Node       next;
        public Node(int data) {
                this.data=data;
        }
}
```

```
public class Test {
        public static void main(String[] args) {
                Node       head=null;
                head=new Node(1);
                head.next=new Node(2);
                head.next.next=new Node(3);
                for (Node p=head; p != null; p=p.next) System.out.print(p.data+"->");
        }
}
```

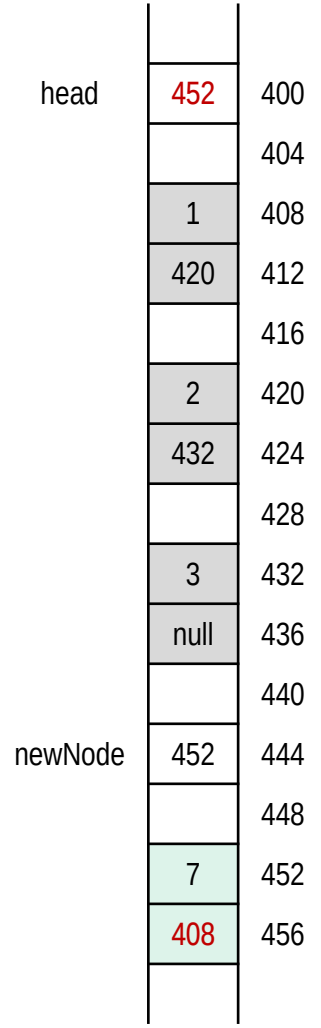


연결리스트: 첫 위치 자료 삽입

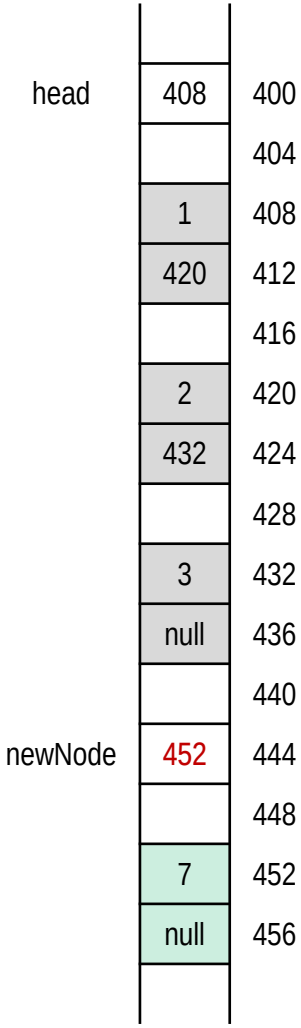


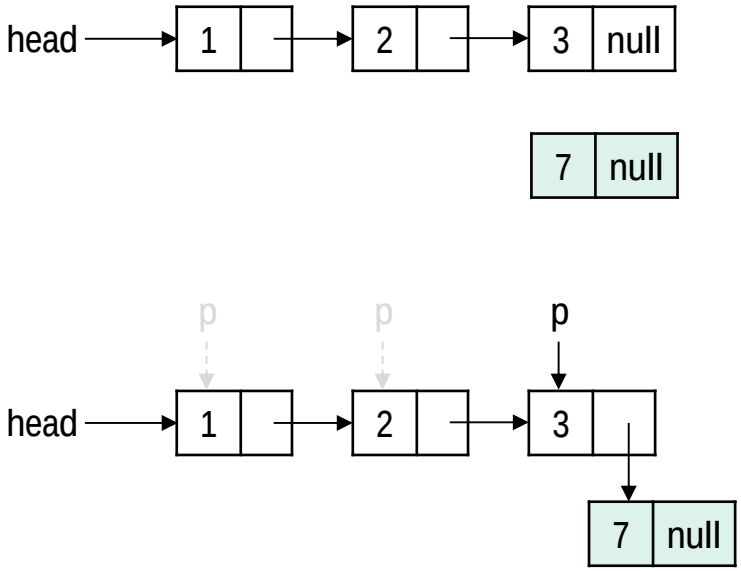


Node newNode=new Node(7);
newNode.next=head;
head=newNode;

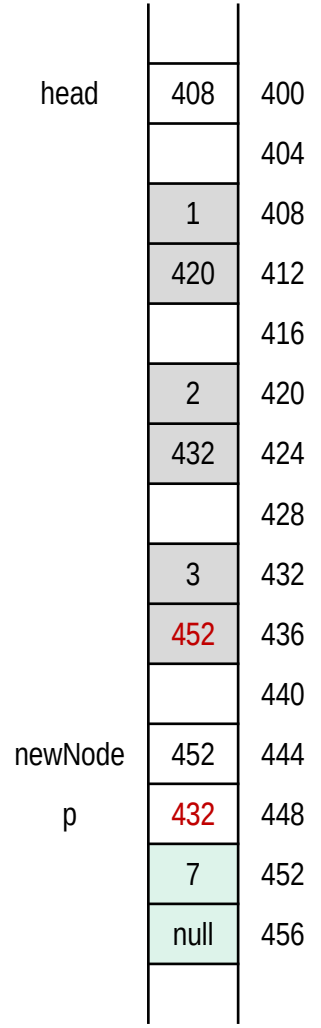


연결리스트: 끝 위치 자료 삽입



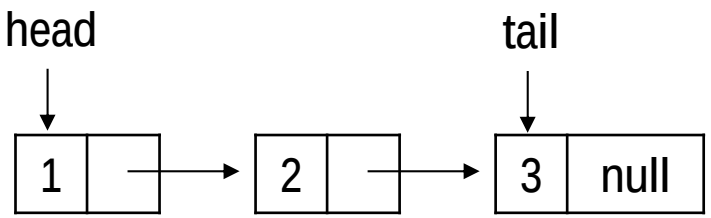


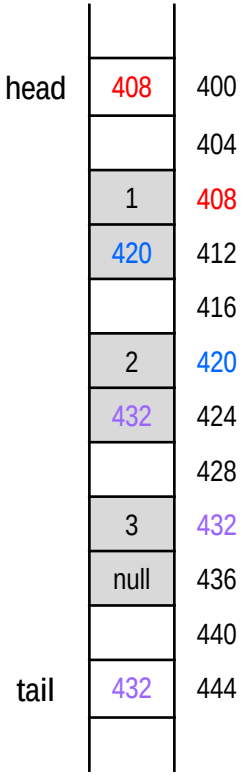

```
Node   newNode=new Node(7);
if(head!=null){
        Node   p=head;
        while(p.next!=null) p=p.next;
        p.next=newNode;
}
else head=newNode;
```



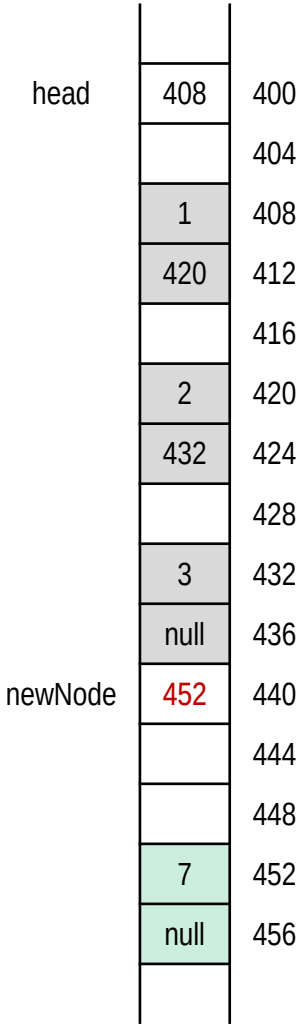
연결리스트: 끝 위치 자료 삽입

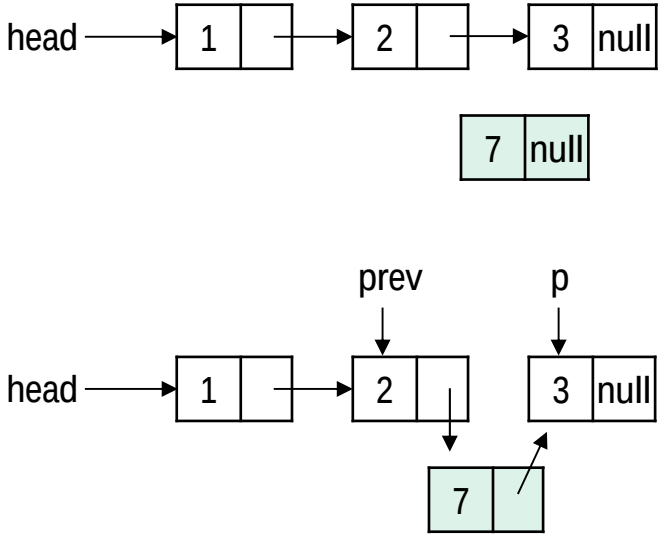
- 끝 위치 자료 삽입
  - Head 링크만 존재하는 경우 비효율적: O(n)
    - 연결리스트의 시작 노드부터 다음 노드로의 link를 따라 마지막 노드까지 이동해야 함
  - Tail 링크가 유지되는 경우: O(1)



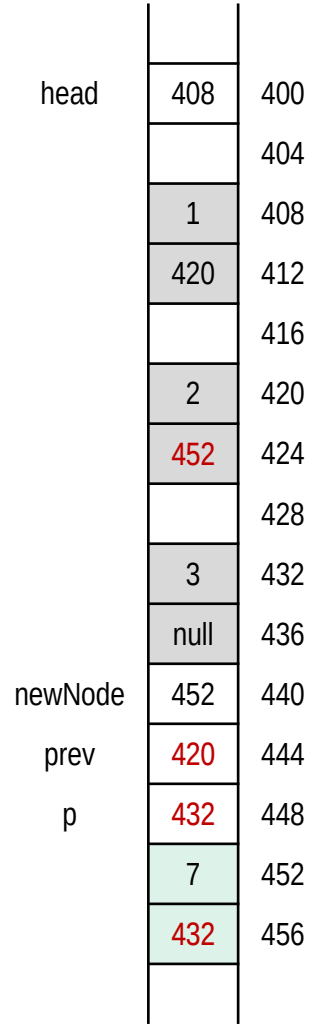


연결리스트: 임의 위치 자료 삽입

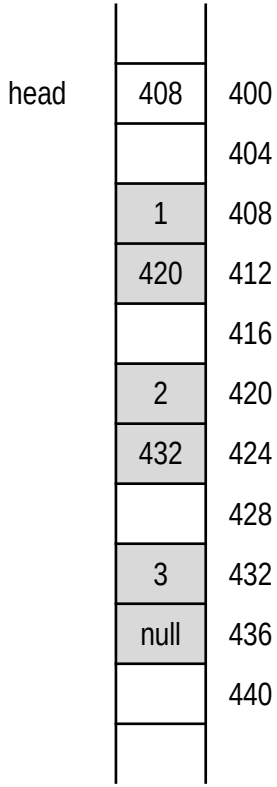


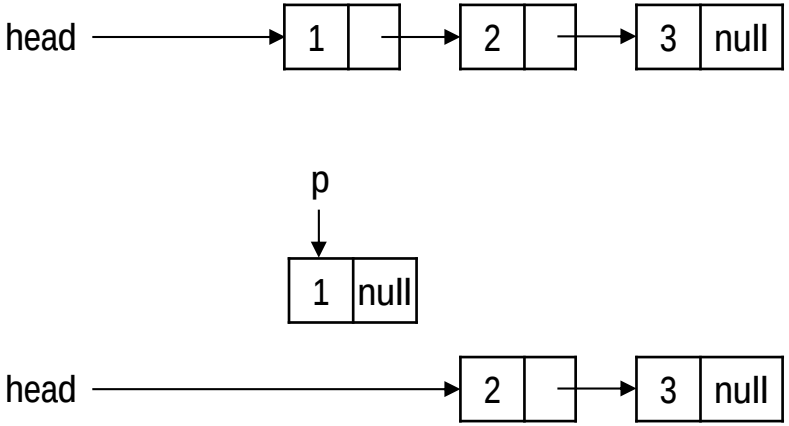



```
int     searchValue=3;
Node        newNode=new Node(7);
Node        p=head, prev=null;
while(p!=null && p.data!=searchValue){ prev=p; p=p.next; }
if(p==null) System.out.println(searchValue+" does not exist");
else{
        if(prev!=null) prev.next=newNode;
        else  head=newNode;
        newNode.next=p;
}
```

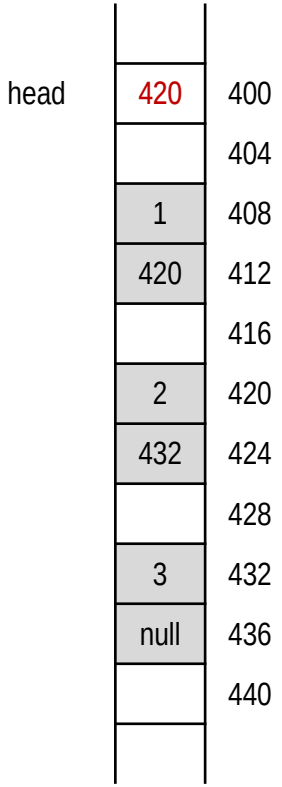


연결리스트: 첫 위치 자료 삭제

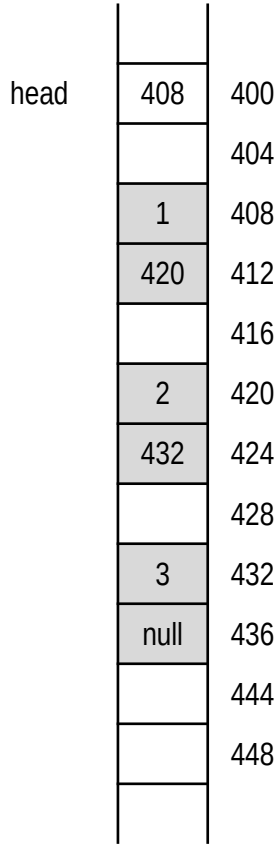


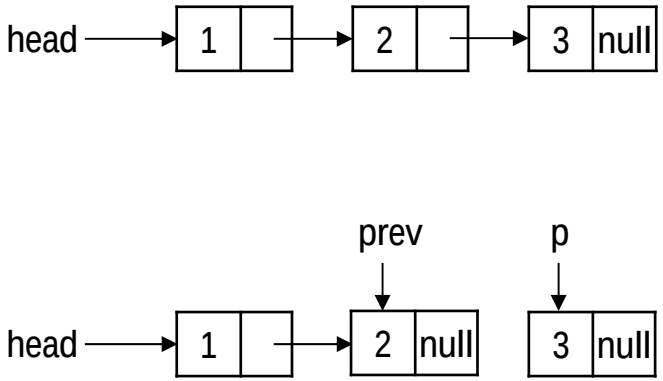



```
if(head!=null){
        Node p=head;
        head=head.next;
        p.next=null;
}
```

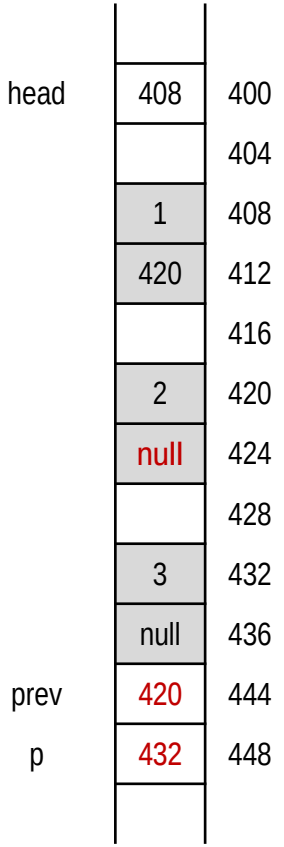


연결리스트: 끝 위치 자료 삭제

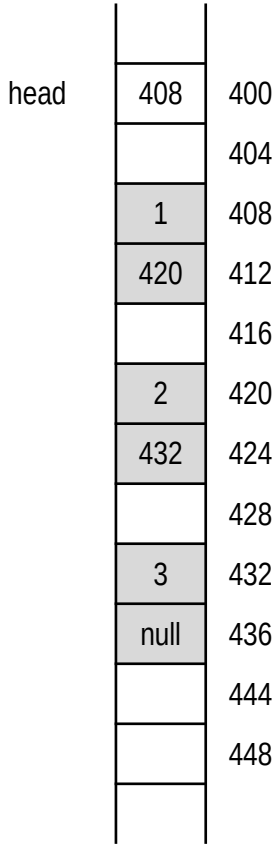


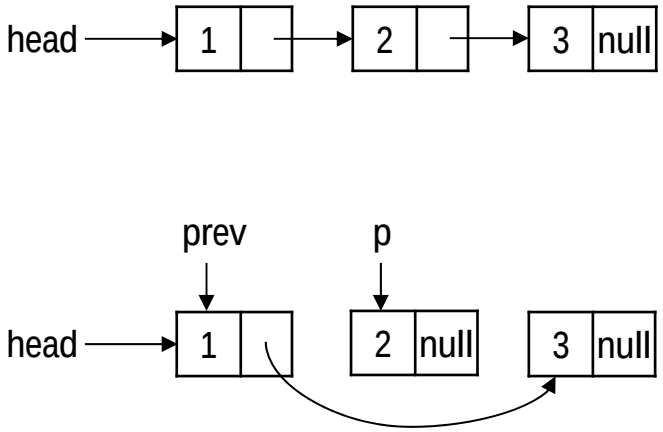



```
if(head!=null){
        Node    p=head, prev=null;
        while(p.next!=null){ prev=p; p=p.next; }
        if(prev!=null) prev.next=null;
        else head=null;
}
```

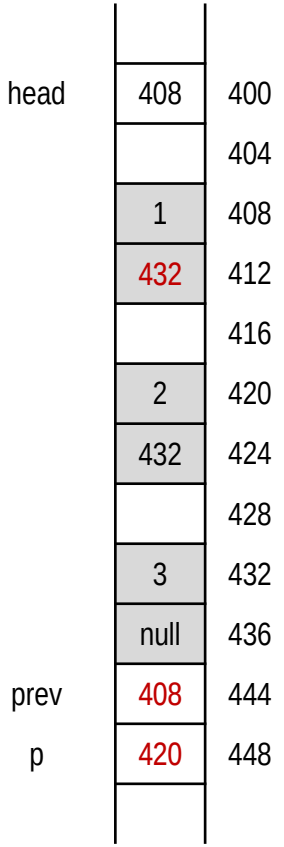


연결리스트: 임의 위치 자료 삭제






```
int     searchValue=2;
Node    p=head, prev=null;
while(p!=null && p.data!=searchValue){ prev=p; p=p.next; }
if(p==null) System.out.println(searchValue+" does not exist");
else{
        if(prev!=null) prev.next=p.next;
        else  head=p.next;
        p.next=null;
}
```



References

- C로 쓴 자료구조론 (Fundamentals of Data Structures in C, Horowitz et al.). 이석호 역. 사이텍미디어. 1993.
- 쉽게 배우는 알고리즘: 관계 중심의 사고법. 문병로. 한빛아카데미. 2013.
- C언어로 쉽게 풀어 쓴 자료구조. 천인국 외 2인. 생능출판사. 2017.