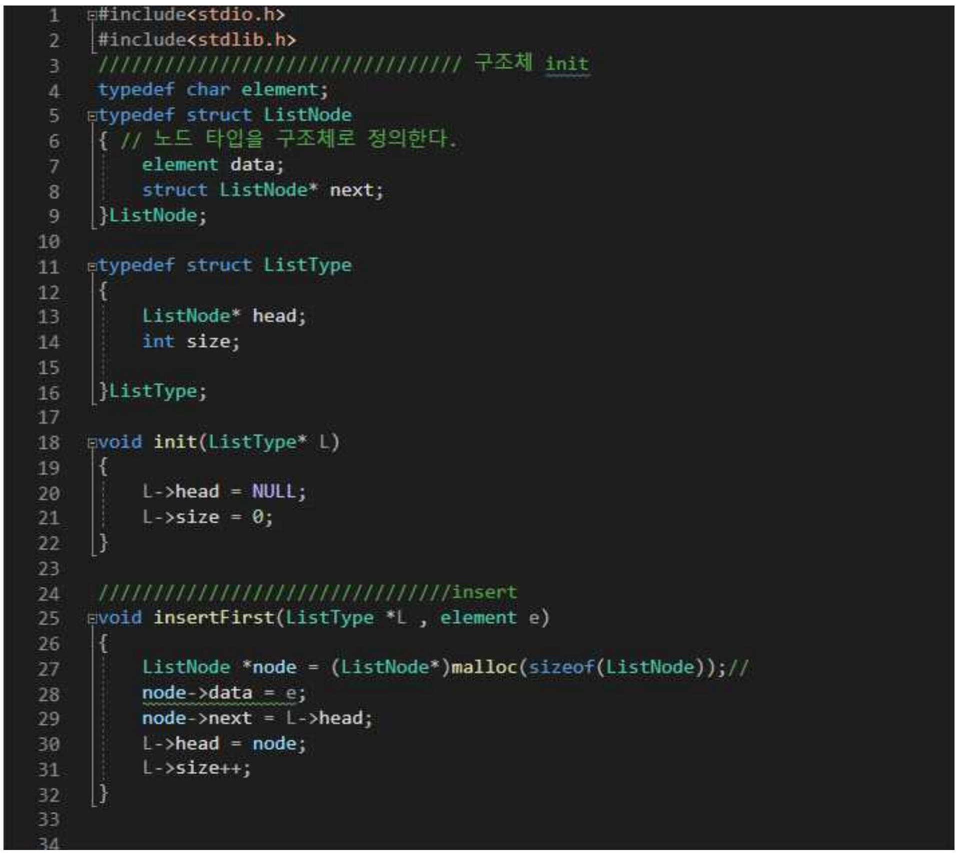
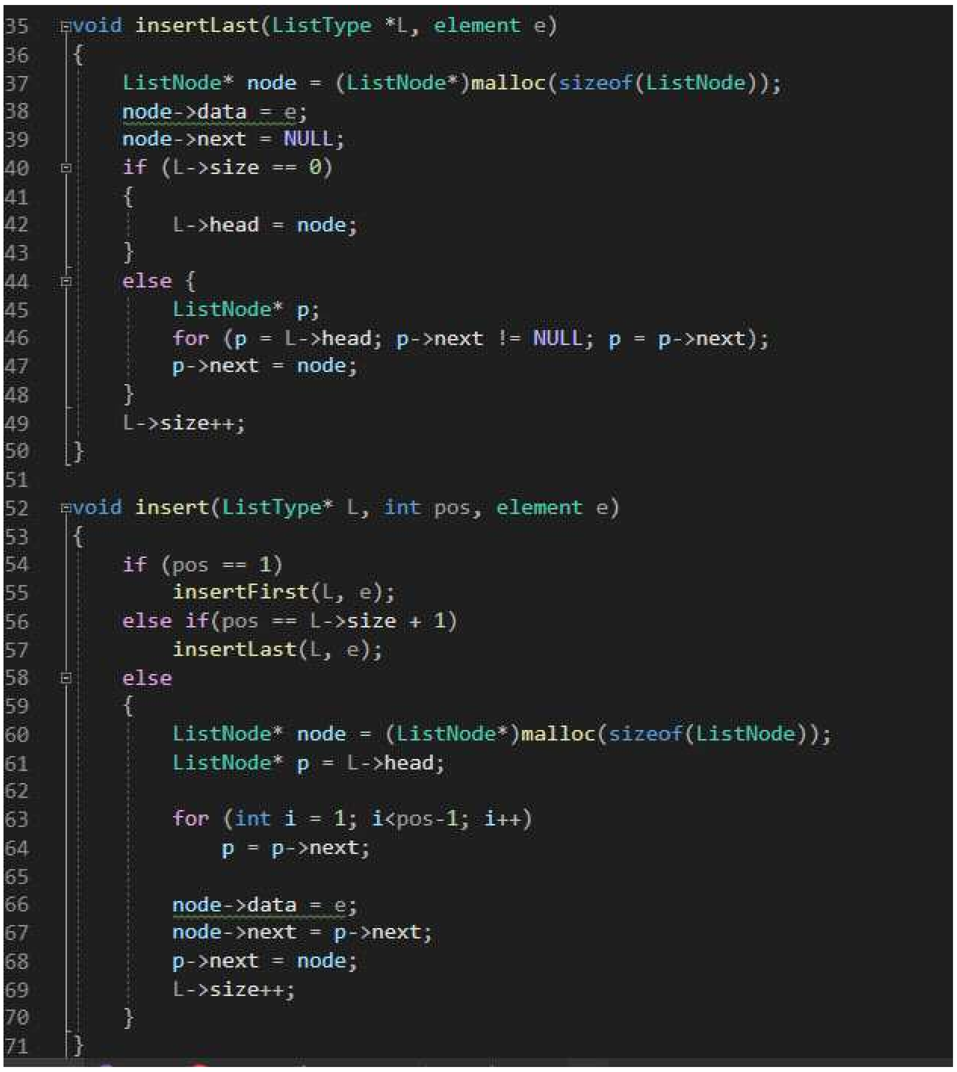
연결리스트 delete실습-20203152최정민

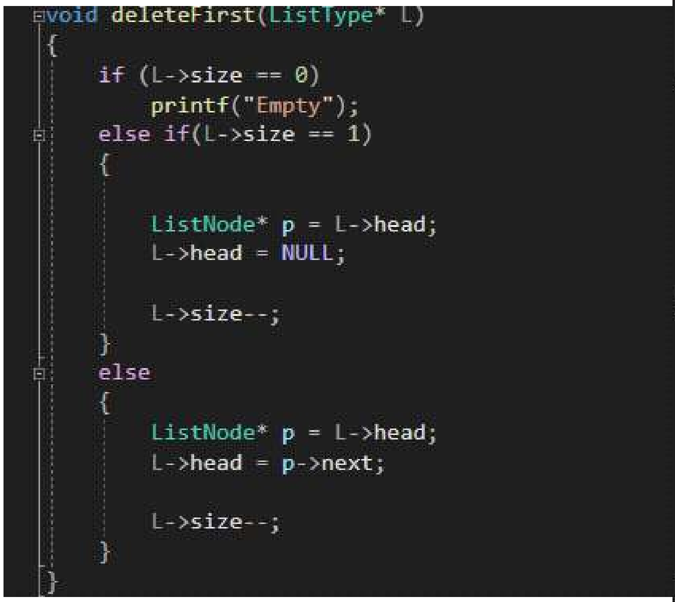
코드



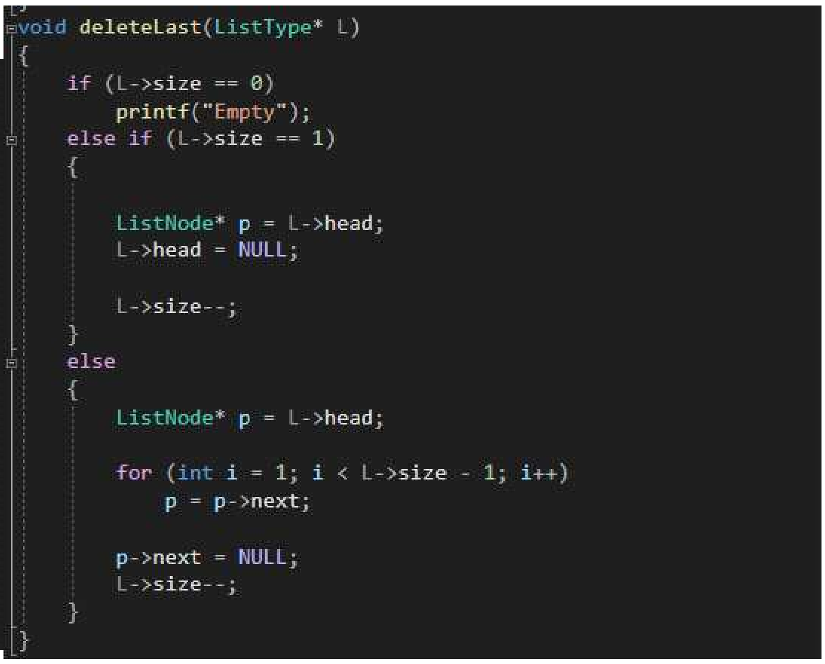
```c
#include<stdio.h>
#include<stdlib.h>
///////////////////////////////// 구조체 init
typedef char element;
typedef struct ListNode
{ // 노드 타입을 구조체로 정의한다.
    element data;
    struct ListNode* next;
}ListNode;

typedef struct ListType
{
    ListNode* head;
    int size;

}ListType;

void init(ListType* L)
{
    L->head = NULL;
    L->size = 0;
}

/////////////////////////////////insert
void insertFirst(ListType *L , element e)
{
    ListNode *node = (ListNode*)malloc(sizeof(ListNode));//
    node->data = e;
    node->next = L->head;
    L->head = node;
    L->size++;
}

```



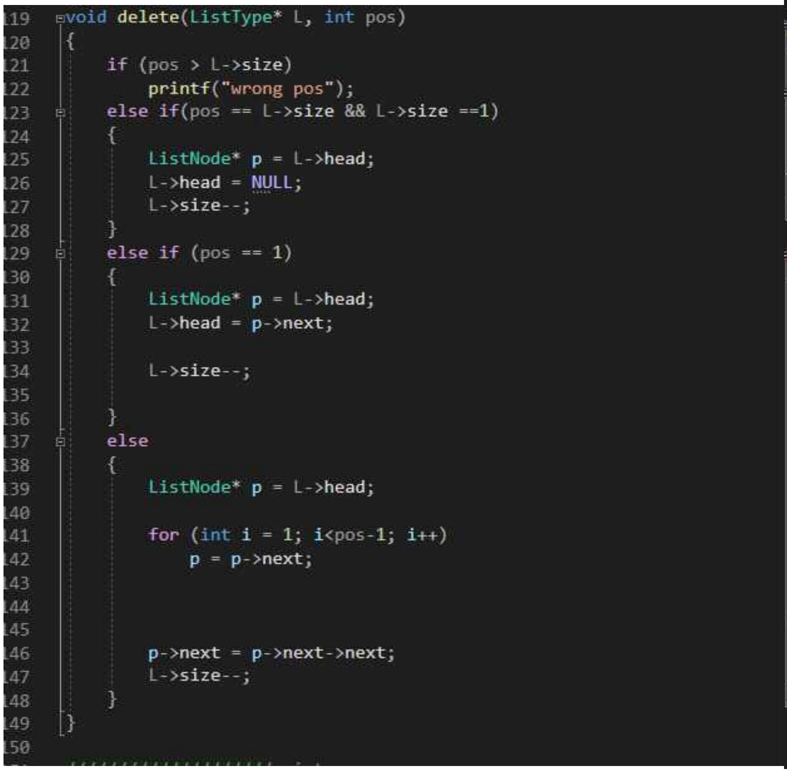
```c
void insertLast(ListType *L, element e)
{
    ListNode* node = (ListNode*)malloc(sizeof(ListNode));
    node->data = e;
    node->next = NULL;
    if (L->size == 0)
    {
        L->head = node;
    }
    else {
        ListNode* p;
        for (p = L->head; p->next != NULL; p = p->next);
        p->next = node;
    }
    L->size++;
}

void insert(ListType* L, int pos, element e)
{
    if (pos == 1)
        insertFirst(L, e);
    else if(pos == L->size + 1)
        insertLast(L, e);
    else
    {
        ListNode* node = (ListNode*)malloc(sizeof(ListNode));
        ListNode* p = L->head;

        for (int i = 1; i<pos-1; i++)
            p = p->next;

        node->data = e;
        node->next = p->next;
        p->next = node;
        L->size++;
    }
}
```



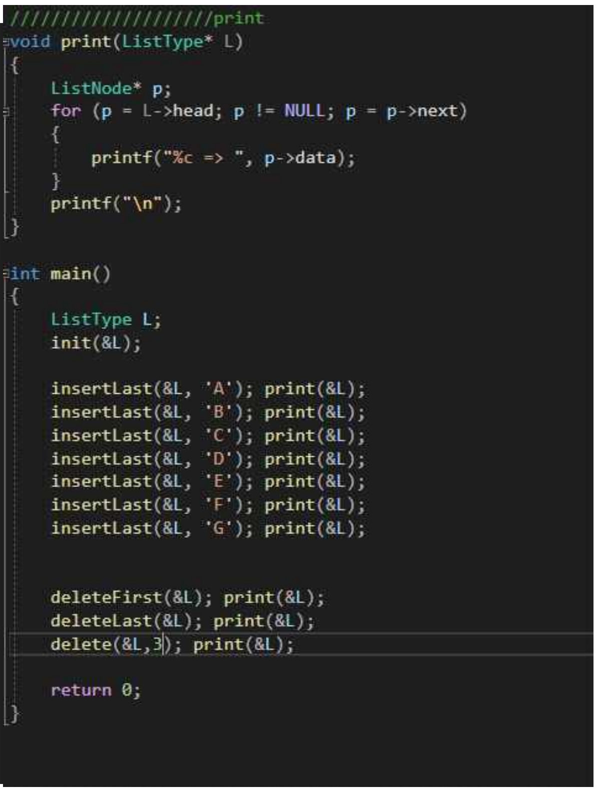
```c
void deleteFirst(ListType* L)
{
    if (L->size == 0)
        printf("Empty");
    else if(L->size == 1)
    {

        ListNode* p = L->head;
        L->head = NULL;

        L->size--;
    }
    else
    {
        ListNode* p = L->head;
        L->head = p->next;

        L->size--;
    }
}
```



```c
void deleteLast(ListType* L)
{
    if (L->size == 0)
        printf("Empty");
    else if (L->size == 1)
    {

        ListNode* p = L->head;
        L->head = NULL;

        L->size--;
    }
    else
    {
        ListNode* p = L->head;

        for (int i = 1; i < L->size - 1; i++)
            p = p->next;

        p->next = NULL;
        L->size--;
    }
}
```

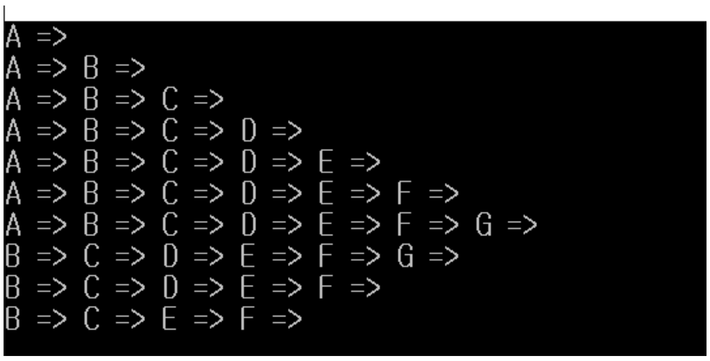


```c
void delete(ListType* L, int pos)
{
    if (pos > L->size)
        printf("wrong pos");
    else if(pos == L->size && L->size ==1)
    {
        ListNode* p = L->head;
        L->head = NULL;
        L->size--;
    }
    else if (pos == 1)
    {
        ListNode* p = L->head;
        L->head = p->next;

        L->size--;

    }
    else
    {
        ListNode* p = L->head;

        for (int i = 1; i<pos-1; i++)
            p = p->next;



        p->next = p->next->next;
        L->size--;
    }
}
```



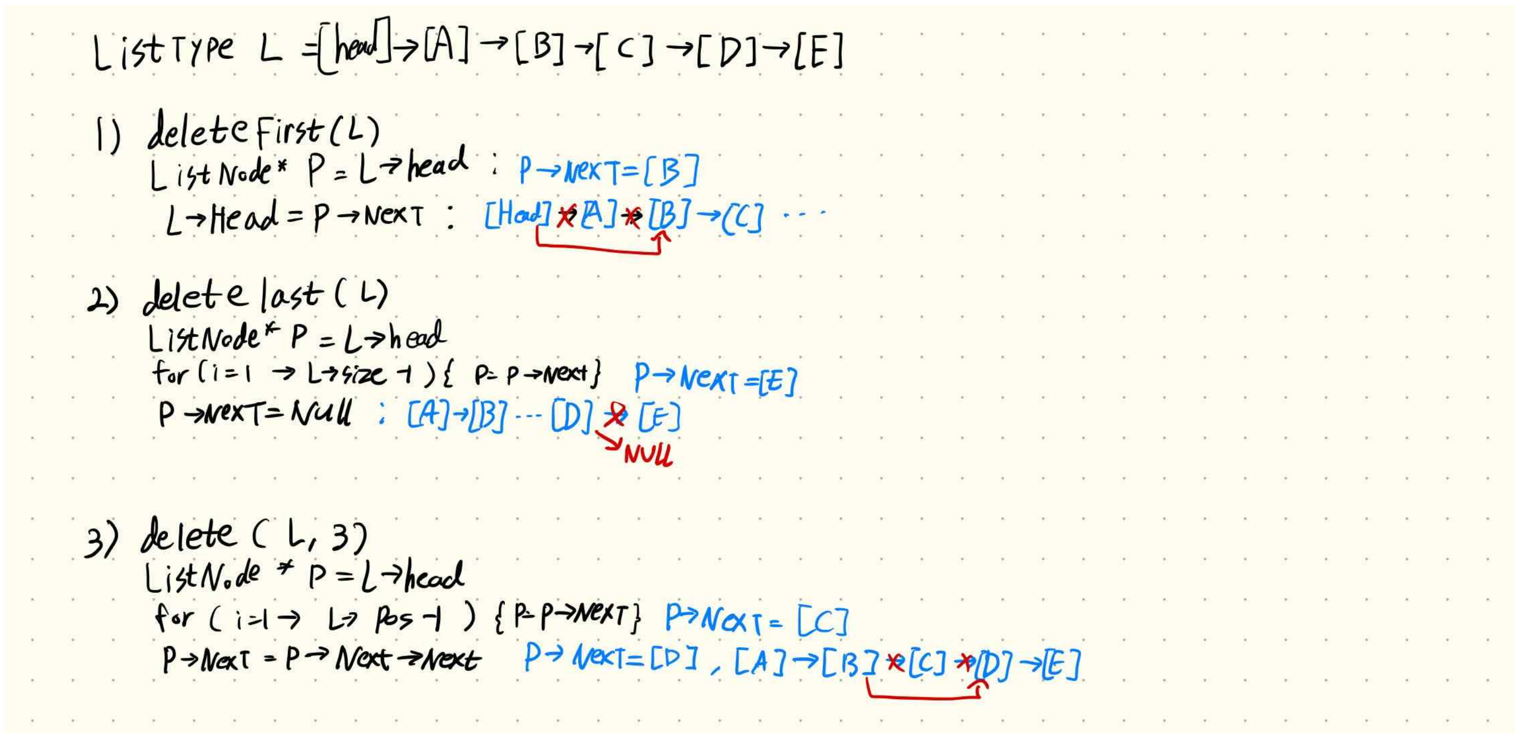
```c
////////////////////print
void print(ListType* L)
{
    ListNode* p;
    for (p = L->head; p != NULL; p = p->next)
    {
        printf("%c => ", p->data);
    }
    printf("\n");
}

int main()
{
    ListType L;
    init(&L);

    insertLast(&L, 'A'); print(&L);
    insertLast(&L, 'B'); print(&L);
    insertLast(&L, 'C'); print(&L);
    insertLast(&L, 'D'); print(&L);
    insertLast(&L, 'E'); print(&L);
    insertLast(&L, 'F'); print(&L);
    insertLast(&L, 'G'); print(&L);


    deleteFirst(&L); print(&L);
    deleteLast(&L); print(&L);
    delete(&L,3); print(&L);

    return 0;
}
```

실행결과



```
A =>
A => B =>
A => B => C =>
A => B => C => D =>
A => B => C => D => E =>
A => B => C => D => E => F =>
A => B => C => D => E => F => G =>
B => C => D => E => F => G =>
B => C => D => E => F =>
B => C => E => F =>
```

실행과정



ListType L = [head]→[A]→[B]→[C]→[D]→[E]

1) deleteFirst(L)
ListNode* P = L→head : P→NexT=[B]
L→Head = P→NexT : [Head] ✗ [A] ✗ [B]→[C] · · ·

2) delete last(L)
ListNode* P = L→head
for(i=1 → L→size-1){ P=P→Next} P→NexT=[E]
P→NexT=Null : [A]→[B]···[D] ✗ [E]
↘NULL

3) delete(L, 3)
ListNode * P = L→head
for(i=1 → L→pos-1){P=P→NexT} P→NexT=[C]
P→NexT = P→Next→Next P→NexT=[D], [A]→[B] ✗ [C] ✗ [D]→[E]