**자료구조 실습 보고서**

**실습 6. 단일연결리스트**

**2016년 4월 15일**

**학번: 201404051**

**이름: 정 용 석**

**1. 실습 문제 소개**

**6-2) 단일 연결 리스트 삭제1**

기존에 작성한 리스트의 구성 기능에서 사용자가 입력한 값을 가지는 노드를 찾아서 삭제하는 searchDeleteNode(L, x)의 함수를 추가하는 문제이다. 리스트 L에 연결되어 있는 모든 노드를 앞에서부터 순차적으로 탐색하여 x값을 가지는 노드를 찾아서 알맞게 삭제해야 한다.

**6-3) 단일 연결 리스트 삭제 2**

실습 6-2에 이어서 리스트 가장 첫 노드와 가장 마지막 노드를 삭제하는 deleteFirstNode(L)와 deleteLastNode(L)의 함수를 추가하여 메뉴에 삽입하는 문제이다. 작성된 알고리즘이 주어지지 않기 때문에, 개인적으로 구상한 알고리즘을 코드 화하여 작성하여야 하고, 각 노드 삭제 이 후, 메모리 누수가 일어나지 않도록 비어 주는 것도 잊으면 안된다.

**2. 소스 코드**

**6-2) 단일 연결 리스트 삭제1**

node\* searchDeleteNode(node \*L, int x) //리스트 L에서 x의 값을 가지는 노드 삭제

{

node \*p; node \*pre;

//리스트 L이 비어있을 때,

if (L == NULL)

return L;

//리스트 L의 첫 노드에 x값이 있을 때,

if (L->data == x)

{

p = L;

L = L->link;

returnNode(p);

return L;

}

//pre는 L의 첫 노드를, P는 L의 두번째 노드를 가리킨다.

pre = L;

p = L->link;

while (p != NULL) //첫 노드부터 끝 노드까지 탐색

{

/\* p가 가리키는 노드에 x값이 있을 시

pre를 p가 가리키는 다음 노드에 연결시킨 후,

p가 가리키는 노드 삭제, 리턴 L \*/

if (p->data == x)

{

pre->link = p->link;

returnNode(p);

return L;

}

pre = p;

p = p->link;

}

return L;

}

**6-3) 단일 연결 리스트 삭제 2**

node\* deleteFirstNode(node \*L) //리스트 L의 첫번째 노드 삭제

{

//List가 비어있을 때, 리턴 L

if (L == NULL)

return L;

/\*L의 첫번째 노드를 temp가 가리키고,

L를 두번쨰 노드와 연결시킨후,

L의 첫번째 노드 삭제, 리턴 L \*/

node\* temp = L;

L = temp->link;

returnNode(temp);

return L;

}

node\* deleteLastNode(node\*L) //리스트 L의 마지막 노드 삭제

{

node \*p = L;

//리스트 EMPTY

if (L == NULL)

return L;

//리스트에 노드 1개

if (L->link == NULL){

returnNode(L->link);

L = NULL;

return L;

}

//리스트에 노드 2개 이상

while (p != NULL){

/\* 리스트 L를 탐색하면서, p의 다음다음 노드가 마지막 노드일 시, deleteNode(L, p)\*/

if (p->link->link == NULL)

deleteNode(L, p);

p = p->link;

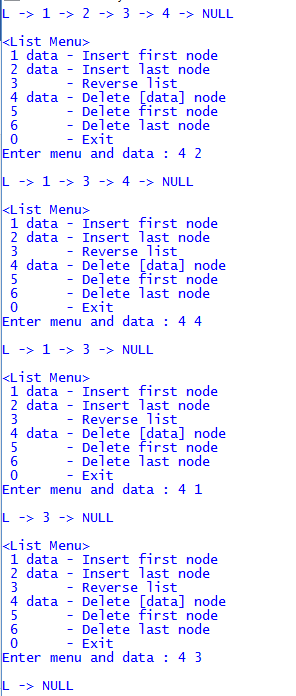
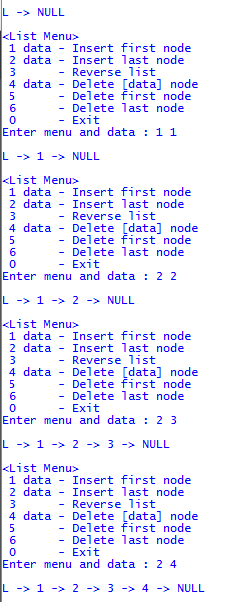
}

return L;

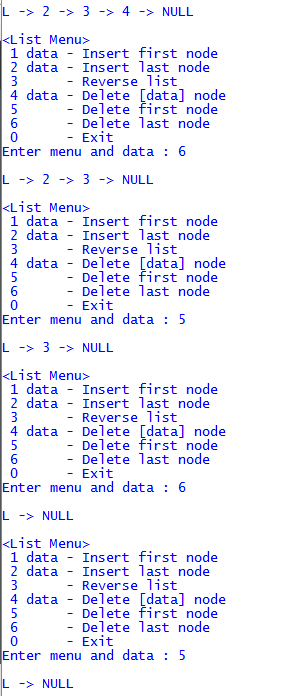
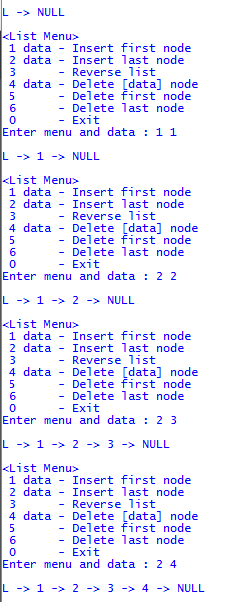
}

**3. 테스트 결과**

**6-2) 단일 연결 리스트 삭제1**



**6-3) 단일 연결 리스트 삭제 2**

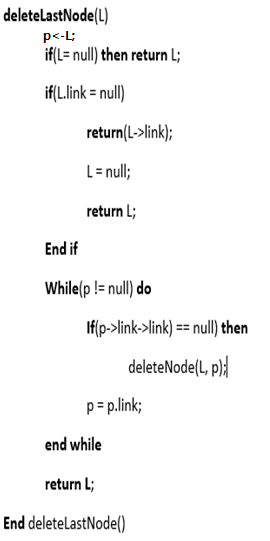
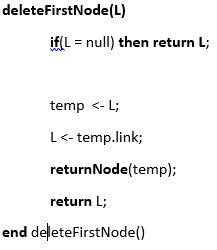


**4. 작성자 코멘트**

**6-2) 단일 연결 리스트 삭제1**

사실 searchDeleteNode의 함수 같은 경우는 알고리즘과 Pseudo코드가 강의 자료에 전부 나와있었기에, 딱히 어렵고 쉽다는 느낌보다는 그냥 있는 그대로 작성 및 테스트 하였다. 알고리즘도 그다지 어렵지 않기 때문에, Pseudo 코드가 주어지지 않았어도, 작성할 수 있을 것 같다.

**6-3) 단일 연결 리스트 삭제 2**



위는 내가 구현한 deleteFirstNode와 deleteLastNode의 Pseudo코드이다. deleteFirstNode와 같은 경우는 굉장히 간단하다. 리스트 L의 첫 번째 노드를 삭제하고 두 번째를 첫번째로 끌고 오는 형식으로, 첫 번째 노드를 temp가 가리키게 하고, L을 두번째 노드와 연결시키고 temp를 삭제하여 마무리한다.

deleteLastNode같은 경우는 deleteNode와 리스트 탐색 알고리즘을 이용하여 구현하였다. 마찬가지로 리스트가 비어있을 때와 노드가 1개만 있을 때의 예외처리를 해주고, 탐색 알고리즘을 이용하여 리스트의 처음부터 끝까지 p가 움직이면서 p->link->link가 NULL인 경우를 확인하는데, 이는 마지막 노드를 가리키게 p가 지속적으로 마지막 노드를 가리키게 하기 위함이다. 이렇게 p의 다음다음 노드가 마지막일 경우, deleteNode를 이용하여 리스트 L에서 p가 가리키는 노드를 다음다음 노드, 즉 NULL에 연결하고 p가 가리키는 다음 노드를 삭제한다.

deleteLastNode의 알고리즘을 생각하면서 느낀 점은 바로 효율성이다. 여기서 구현된 리스트 같은 경우는 시작 포인터 L만 존재할 뿐, 마지막을 칭하는 인위적인 포인터는 존재하지않는다. 이는 deleteLastNode를 호출할 때마다, 리스트의 모든 노드를 탐색해야하는 번거로움이 생긴다. 리스트가 적으면 큰 문제는 없겠지만, 리스트가 수십 만개에 이르게 된다면, 이는 엄청난 시간적 효율 저하를 일으킬 것이다. 따라서 이에 대한 대처방법을 생각한 것이 항상 처음을 가리키는 L포인터 이외에도 마지막을 가리키는 Last 포인터를 선언하여, 프로그램 실행 간 지속적으로 tracking하여 마지막 노드를 삭제시킬 시 사용할 수 있으면 어떨까 한다. 물론 실행에 옮기지는 않았지만 충분히 가능할 것 같다.