Data Structure Linked Lists
List ADT

- Data : n개의 element로 구성된 순서 있는 모임 (linkedList)
- Operations :
  - linkedList* initList() : 공백 리스트 생성
  - void insertFirst(linkedList* L, element x) : 리스트의 첫 번째 노드로 삽입
  - void insertLast(linkedList* L, element x) : 리스트의 마지막 노드로 삽입
  - void insert(linkedList* L, listNode* pre, element x) : 리스트 L 중간에 x(data) 노드 삽입 (pre: 삽입할 위치의 앞 노드)
  - int delete(linkedList* L, listNode* p) : 리스트에서 p 노드 삭제
  - listNode* search(linkedList* L, element x) : data로 x가 저장되어 있는 노드를 검색
  - int getLength(linkedList* L) : 리스트의 길이(항목의 개수)를 구함
  - void displayList(linkedList* L) : 리스트의 모든 요소를 표시
  - void clear(linkedList* L) : 리스트의 모든 요소 삭제

Doubly Linked Lists
Insertion I Algorithm

Algorithm	void insert(linkedList* L, listNode* pre, element x)

```
insert(L, pre, x)
   newNode.data ← x
   if (L = NULL) then
      newNode.rlink ← NULL
      newNode.llink ← NULL
      L ← newNode
   else if (pre = NULL) then
      newNode.llink ← NULL
      newNode.rlink ← L
      L ← newNode
   else
      newNode.rlink ← pre.rlink
      pre.rlink ← newNode
      newNode.llink ← pre
      if (newNode.rlink ≠ NULL) then
         newNode.rlink.llink ← newNode
   endif
length++
end insert()
```

공백리스트에 삽입

리스트의
맨 처음 노드로 삽입

리스트의
중간에 삽입

삽입 자리에
다음 노드가 있는 경우

Doubly Linked Lists
Deletion Algorithm

Algorithm | **int delete(linkedList* L, listNode* p)**

```
delete(L, p)
  if (L = NULL) then return FALSE
  if (p = NULL) then return FALSE
  if (p.llink = NULL) then
     L ← p.rlink
  else
     p.llink.rlink ← p.rlink
  endif
  if (p.rlink ≠ NULL) then
     p.rlink.llink ← p.llink
  endif
  delete p
  length—
  return TRUE
end delete()
```

공백리스트인 경우

삭제할 노드가 없는 경우

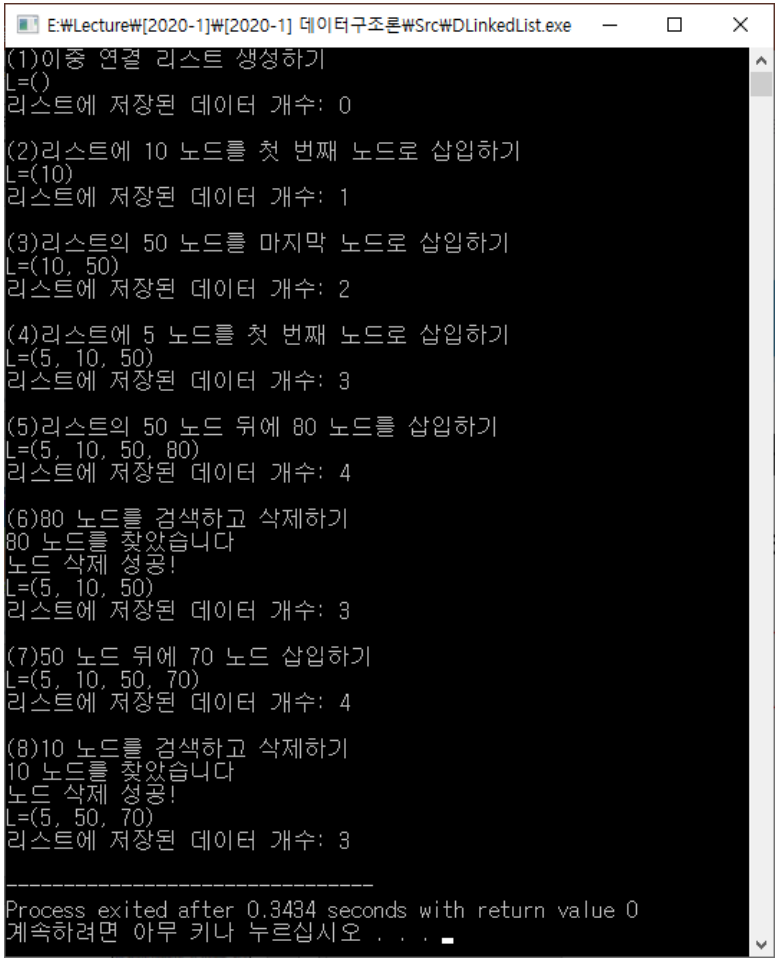
첫 번째 노드가 삭제되는 경우

삭제할 노드가
마지막 노드가 아닌 경우

Doubly Linked Lists

HW#4.1. Doubly Linked List 구현

- 아래와 같이 실행되도록 main()함수 구성
- Linked List ADT의 모든 연산 구현
- DLinkedList.h 및 DLinkedListMain.c제공
- DLinkedList.c 완성하여 제출



Doubly Linked Lists
Polynomial ADT

- Data : 지수(e_i), 계수(a_i)의 순서쌍 <e_i, a_i>의 집합으로 표현된 다항식
- Operations : A, B는 다항식, coef는 계수, exp는 지수
  - polyList* initList() : 공백 다항식 리스트 생성
  - void appendTerm(polyList* PL, float coef, int exp) : 다항식 PL에 계수가 coef, 지수가 exp인 노드 항 삽입
  - polyList* addPoly(polyList* A, polyList* B) : 다항식 A와 B의 합을 구함
  - polyList* mulPoly(polyList* A, polyList* B) : 다항식 A와 B의 곱을 구함
  - void displayPoly(polyList* PL) : 다항식 리스트 출력

Doubly Linked Lists
Polynomial: 항 삽입

- 다항식 리스트의 마지막 요소 뒤에 항 삽입
  - 리스트의 마지막 노드로 삽입하는 알고리즘과 유사

Algorithm	void appendTerm(polyList* PL, float coef, int exp)

```
appendTerm(PL, coef, exp)
    newNode.coef ← coef
    newNode.exp ← exp
    newNode.link ← NULL
    if (PL = NULL) then
        PL.head ← newNode
    else
        PL.last.link ← newNode
    endif
    PL.last ← newNode
end appendTerm()
```

Doubly Linked Lists
Polynomial: 다항식 덧셈

Algorithm | **polyList* addPoly(polyList* A, polyList* B)**

```
addPoly(A, B)
   p ← A, q ← B
   C ← NULL
   while (p ≠ NULL and q ≠ NULL) do
      if (p.exp = q.exp) then
         sum ← p.coef + q.coef
         if (sum ≠ 0) then appendTerm(C, sum, p.exp)
         p ← p.link, q ← q.link
      else if (p.exp > q.exp) then
         appendTerm(C, p.coef, p.exp)
         p ← p.link
      else
         appendTerm(C, q.coef, q.exp)
         q ← q.link
      endif
   endwhile
   while (p ≠ NULL) do
      appendTerm(C, p.coef, p.exp)
      p ← p.link
   endwhile

   while (q ≠ NULL) do
      appendTerm(C, q.coef, q.exp)
      q ← q.link
   endwhile

   return C
end addPoly()
```

HW#4.2. 다항식의 덧셈

- 다항식 A, B가 아래와 같이 주어졌을 때, addPoly() 연산 구현

$$A(x) = 4x^3 + 3x^2 + 5x$$

$$B(x) = 3x^4 + x^3 + 2x + 1$$

- PolyList.h 및 PolyListMain.c제공, PolyList.c 완성