HW#3 과제 안내

1. 일정

   게시 : 3/23(화) 17:00
   제출 마감 : 3/30(화) 15:00
   채점 결과 확인 : 4/5(월)
   이의신청 마감 : 4/12(월) (이의신청 이메일 : greenlife124@yonsei.ac.kr, [데이타구조론]HW#3 이의신청)

2. 설명

   모든 코드의 **핵심 부분**에는 **comment를 달아 설명** 할 것 (**not option!!**)
   compiler는 visual studio 2019 이상을 사용 하여, **HW#2_학번_이름 하나의 파일로 압축**하여 제출 할 것

   1. folder name : **HW#3_1**, included file name : **SLinkedList.h**, **SLinkedList.c**, **SLinkedListMain.c**
      SLinkedList.h 제공, SLinkedList(**getLength**, **insert**, **insertFirst**, **insertLast**, **deleteNode**, **deleteData**, **search**), SLinkedListMain (**main**) **작성**
   2. folder name : **HW#3_2**, included file name : **SLinkedList2.h, SLinkedList2.c, SLinkedListMain2.c**
      SLinkedList2.h 일부 **수정**, SLinkedList(**getLength**, **insert**, **insertFirst**, **insertLast**, **delete**, **search**, **reverse**), SLinkedListMain2 (**main**) **작성**
   3. folder name : **HW#3_3**, included file name : **SLinkedList3.h, SLinkedList3.c, SLinkedListMain3.c**
      SLinkedList3(**getLength**, **sortInsert**, **delete**, **search**), SLinkedListMain3 (**main**) **작성**

HW#3.1 Linked List 구현 (HW#3_1)

- 다음과 같이 출력 되도록 SLinkedList.c, SlinkedListMain.c 작성
  - SLinkedList.c 작성해야 할 함수
    - getLength(linkedList* L)
    - insert(linkedList* L, listNode* pre, element x)
    - insertFirst(linkedList* L, element x)
    - insertLast(linkedList* L, element x)
    - deleteNode(linkedList* L, listNode* p)
    - deleteData(linkedList* L, element x)
    - search(linkedList* L, element x)
  - SLinkedListMain.c 작성해야 할 함수
    - main( )

- 주의사항
  - insert, deleteNode, deleteData 함수 작성 시, 주석에서 요구한 내용 모두 작성 할 것 (공백 리스트인 경우, 삭제할 노드가 없는 경우 등)
  - HW#3_1 폴더 안에, 주어진 3개의 코드 모두 존재 해야 함

출력

```
(1)공백리스트 생성하기
L=()
리스트에 저장된 데이터 개수 : 0

(2)리스트에 10, 50 노드를 순서대로 삽입하기
L=(10, 50)
리스트에 저장된 데이터 개수 : 2

(3)리스트에 마지막에 80 노드를 추가하기
L=(10, 50, 80)
리스트에 저장된 데이터 개수 : 3

(4)리스트에서 50 노드 탐색하기
50 노드를 찾았습니다

(5)50 노드 뒤에 60 노드 삽입하기
L=(10, 50, 60, 80)
리스트에 저장된 데이터 개수 : 4

(6)리스트에서 80 노드 삭제하기( node 사용 )
노드 삭제 성공!
L=(10, 50, 60)
리스트에 저장된 데이터 개수 : 3

(7)리스트에서 50 노드 삭제하기( element 사용 )
노드 삭제 성공!
L=(10, 60)
리스트에 저장된 데이터 개수 : 2
```

HW#3.2 Linked List 추가 구현 (HW#3_2)

- 다음과 같이 출력 되도록 SLinkedList2.c, SlinkedListMain2.c 작성 및 SLinkedList2.h 부분 수정 (element의 data type 변경)
- 리스트의 노드 순서를 역순으로 바꾸는 reverse() 연산 추가
  - SLinkedList2.c 작성 해야 할 함수
    - getLength(linkedList* L)
    - insert(linkedList* L, listNode* pre, element x)
    - insertFirst(linkedList* L, element x)
    - insertLast(linkedList* L, element x)
    - delete(linkedList* L, listNode* p)
    - search(linkedList* L, element x)
    - reverse(linkedList* L)
  - SLinkedListMain2.c 작성해야 할 함수
    - main( )
  - SLinkedList2.h 수정해야 할 부분
    - typedef int element

- 주의사항
  - HW#3_2 폴더 안에, 주어진 3개의 코드 모두 존재 해야 함

출력

```
(1)공백리스트 생성하기
L=()
리스트에 저장된 데이터 개수: 0

(2)리스트에 월, 화, 목 노드를 순서대로 삽입하기
L=(월, 화, 목)
리스트에 저장된 데이터 개수: 3

(3)리스트의 가장 마지막에 일 노드 삽입 하기
L=(월, 화, 목, 일)
리스트에 저장된 데이터 개수: 4

(4)화 노드 뒤에 수 노드 삽입하기
화 노드를 찾았습니다.
L=(월, 화, 수, 목, 일)
리스트에 저장된 데이터 개수: 5

(5)리스트에서 토 노드 탐색하기
찾는 데이터가 없습니다.

(6)리스트 순서를 역순으로 바구기
L=(일, 목, 수, 화, 월)
리스트에 저장된 데이터 개수: 5

(7)리스트에서 화 노드 삭제하기
노드 삭제 성공!
L=(일, 목, 수, 월)
리스트에 저장된 데이터 개수: 4
```

HW#3.3 Linked List 추가 구현 (HW#3_3)

- 다음과 같이 출력 되도록 SLinkedList3.c, SlinkedListMain3.c 작성
  - SLinkedList3.c 작성 해야 할 함수
    - getLength(linkedList* L)
    - sortInsert(linkedList* L, int(*comp)(element d1, element d2))
    - delete(linkedList* L, listNode* p)
    - search(linkedList* L, element x)
  - SLinkedListMain3.c 작성해야 할 함수
    - main( )

- 주의사항
  - HW#3_3 폴더 안에, 주어진 3개의 코드 모두 존재 해야 함
- tip
  - return d1 > d2 ? 0 : 1; 은 if(d1 > d2) return 0; else return 1;
  - 뒷장 오름차순 정렬에 대한 참고 자료 참조

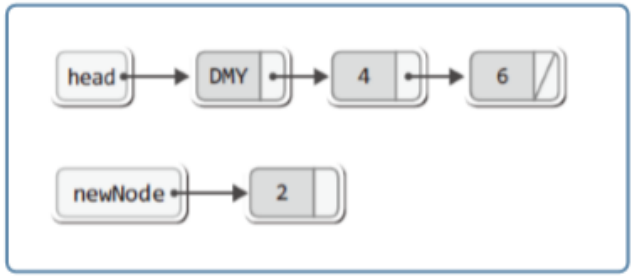
출력

```
(1)공백리스트 생성하기
L=()
리스트에 저장된 데이터 개수: 0

(2)리스트에 10, 80, 50 노드를 삽입하되 내림차순으로 들어가도록 하기
L=(10)
L=(80, 10)
L=(80, 50, 10)
리스트에 저장된 데이터 개수: 3

(3)르스트에서 50 노드 탐색하기
50 노드를 찾았습니다

(4)40 노드 삽입하기(내림차순)
L=(80, 50, 40, 10)
리스트에 저장된 데이터 개수: 4

(5)리스트에서 50 노드 삭제하기
노드 삭제 성공!
L=(80, 40, 10)
리스트에 저장된 데이터 개수: 3
```

HW#3.3 Linked List 추가 구현 (오름차순 정렬)

- 오름차순 정렬 참고 자료

```
void FInsert(List* pList, LData data)
{
	Node* newNode = new Node();
	newNode->data = data;

	newNode->next = pList->head->next;
	pList->head->next = newNode;

	(pList->numOfData)++;
}

void SInsert(List* pList, LData data)
{

}

void ListInsert(List* pList, LData data)
{
	if (pList->comp == NULL)    //정렬기준
		FInsert(pList, data);
	else
		SInsert(pList, data);
}
```



FInsert