자료구조응용
6. Sorting: insertion sort, quick sort

1. 다음 입력 리스트에 대해 insertionSort(Program 7.5)의 for문에서 insert() 함수 실행 이후의 배열상태를 순서대로 기술하라.

   입력 리스트 (12, 2, 16, 30, 8, 28, 4, 10, 20, 6, 18)

```
void insertionSort(element a[], int n)
{/* sort a[1:n] into nondecreasing order */
    int j;
    for (j = 2; j <= n; j++) {
        element temp = a[j];
        insert(temp, a, j-1);
    }
}
```

Program 7.5: Insertion sort

※ 연습장에 적은 후 사진을 찍어도 되며 그 결과를 보고서에 넣을 것

2. 다음과 같이 사용자로부터 데이터를 입력받아서 삽입정렬(insertion sort)을 수행한 결과를 출력하라. 각 레코드는 (key, name, grade)의 필드로 구성된다.

(1) 입력파일(input.txt)

```
7
10 송중기 95
35 조인성 89
25 김수미 59
50 홍길동 33
15 아이유 65
11 박용우 78
33 장윤정 67
```

(2) 실행순서

① 입력파일로부터 데이터를 읽어 들여 구조체 배열에 저장한다.
② key에 대해 삽입정렬을 실행한다.
③ 정렬된 순서대로 (key, name, grade)를 화면에 출력한다.
④ 정렬결과를 파일(output.txt)에 저장한다.

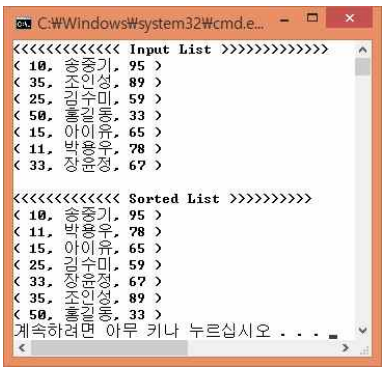
```
void insert(element e, element a[], int i)
{/* insert e into the ordered list a[1:i] such that the
    resulting list a[1:i+1] is also ordered, the array a
    must have space allocated for at least i+2 elements */
   a[0] = e;
   while (e.key < a[i].key)
   {
      a[i+1] = a[i];
      i--;
   }
   a[i+1] = e;
}
```

Program 7.4: Insertion into a sorted list

```
void insertionSort(element a[], int n)
{/* sort a[1:n] into nondecreasing order */
   int j;
   for (j = 2; j <= n; j++) {
      element temp = a[j];
      insert(temp, a, j-1);
   }
}
```

Program 7.5: Insertion sort

(3) 실행 예



3. 다음 입력 리스트에 대해 퀵정렬(quick sort)를 수행하고자 한다. Figure 7.1과 같은 퀵정렬 과정을 보이고 이를 통해 quickSort 함수가 몇 번 호출되는지 계산해 보라.

입력 리스트 (12, 2, 16, 30, 8, 28, 4, 10, 20, 6, 18)

R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}	*left*	*right*
[26	5	37	1	61	11	59	15	48	19]	1	10
[11	5	19	1	15]	26	[59	61	48	37]	1	5
[1	5]	11	[19	15]	26	[59	61	48	37	1	2
1	5	11	[19	15]	26	[59	61	48	37]	4	5
1	5	11	15	19	26	[59	61	48	37]	7	10
1	5	11	15	19	26	[48	37]	59	[61]	7	8
1	5	11	15	19	26	37	48	59	[61]	10	10
1	5	11	15	19	26	37	48	59	61		

Figure 7.1: Quick sort example

※ 연습장에 적은 후 사진을 찍어도 되며 그 결과를 보고서에 넣을 것

4. 위 3번 입력리스트의 데이터를 파일로 입력받아 퀵정렬 수행결과 및 quickSort 함수호출 회수를 구하여 출력하라. 단, 각 레코드는 하나의 int형 key 필드로 구성되어 있다.

(1) 입력파일(input.txt)

```
11
12 2 16 30 8 28 4 10 20 6 18
```

※ 첫 줄은 레코드의 정렬할 키의 개수

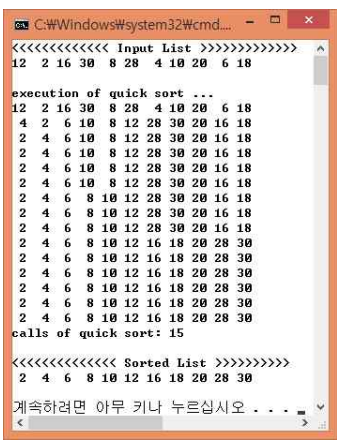
(2) 실행순서

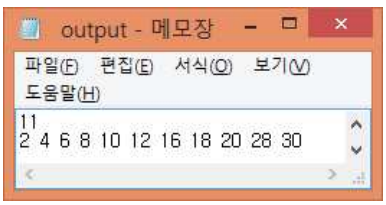
① 입력파일(input.txt)로부터 데이터를 읽어 들여 구조체 배열 a에 저장한다.
② 각 레코드의 key에 대해 퀵정렬을 실행한다.
③ **정렬된 key값** 및 **quickSort 함수호출 회수**를 화면에 출력하라.
④ 정렬결과를 파일(output.txt)에 저장한다.

```
void quickSort(element a[], int left, int right)
{/* sort a[left:right] into nondecreasing order
    on the key field; a[left].key is arbitrarily
    chosen as the pivot key;  it is assumed that
    a[left].key <= a[right+1].key */
  int pivot,i,j;
  element temp;
  if (left < right) {
     i = left; j = right + 1;
     pivot = a[left].key;
     do {/* search for keys from the left and right
            sublists, swapping out-of-order elements until
            the left and right boundaries cross or meet */
        do i++; while (a[i].key < pivot);
        do j--; while (a[j].key > pivot);
        if (i < j) SWAP(a[i],a[j],temp);
     } while (i < j);
     SWAP(a[left],a[j],temp);
     quickSort(a,left,j-1);
     quickSort(a,j+1,right);
  }
}
```

Program 7.6: Quick sort

(3) 실행 예





■ 제출 형식

- 제출 : 12월 6일자 채점서버에 소스 업로드(12월 19일까지)
- 실행화면을 캡쳐하여 한글파일 보고서(**DS 6_학번.hwp**)에 추가 후 과제에 제출 (12월 19일까지)
- 각 소스파일에 주석처리 추가

  “학번 이름”

  “본인은 이 소스파일을 다른 사람의 소스를 복사하지 않고 직접 작성하였습니다.”

■ **주의**

- 소스복사로는 실력향상을 기대할 수 없습니다!!!

- 먼저 제출한 학생은 남은 시간 동안 자료구조 관련 개인학습을 하거나 동료를 도와 줄 것
- 수강생 끼리 서로 물어보고 논의를 해도 됨
- 채점서버에 제출하지 않는 경우 최종 점수의 0점으로 처리함

(사용법을 잘 모르겠다면 개인적으로 튜터나 TA를 찾아와서 물을 것)