1. 2주차 실습에 구현하는 랭킹 시스템에 대한 자료를 읽어보고, 이를 구현하기 위한 다양한 자료구조를 2가지 이상 생각한다.

(1) 배열 (array)

이름과 점수를 점수에 따라 차례대로 배열에 저장해 줄 수 있다. 이 경우 새 rank를 삽입할 때 그 점수가 어느 위치에 들어가야 하는지 찾아야 하고 찾은 뒤에는 삽입 후 그 뒤의 rank 기록들을 하나씩 뒤로 옮겨줘야 한다. 삭제의 경우에도 마찬가지로 삭제할 기록의 위치를 찾은 뒤 삭제 기록 뒤의 기록들을 하나씩 앞으로 옮겨줘야 한다. 즉 노드의 개수가 n개라면 최악의 경우 삽입과 삭제 시 시간복잡도가 O(n)이다.

(2) 링크드 리스트 (linked list)

이름과 점수를 점수에 따라 차례대로 연결하여 저장해 줄 수 있다. 이 경우 새 rank를 삽입 및 삭제 시 배열과 마찬가지로 그 점수가 어느 위치에 들어가야 하는지 찾아야 한다. 따라서 최악의 경우 노드의 개수가 n개일 때 시간복잡도가 O(n)이다. 한편 링크드 리스트의 경우 삽입과 삭제 시 해당 기록 뒤의 모든 기록들을 옮겨줄 필요는 없다. 삽입 및 삭제하려는 노드 앞의 노드만 수정해주면 된다.

2. 생각한 각 자료구조에 대해서 새로운 랭킹을 삽입 및 삭제하기 위해 필요한 pseudo code를 작성하고, 시간 및 공간 복잡도를 계산한다. (노드의 전체 개수: n개)

(1) 배열 (array)

```
insert_rank(name, score)
    i = 0
    while (rank_array[i] ≥ score)
        i = i + 1
    for (i = len downto i)
        rank_array[i+1].name = rank_array[i].name
        rank_array[i+1].score = rank_array[i].score
    rank_array[i].name = name
    rank_array[i].score = score
    increase length of rank_array
```

시간 복잡도: O(n), 공간 복잡도: O(n)

```
delete_rank(rank)
    for (i = rank to length of rank_array)
        rank_array[i].name = rank_array[i+1].name
        rank_array[i].score = rank_array[i+1].score
    decrease length of rank_array
```

시간 복잡도: O(n), 공간 복잡도: O(n)

(2) 링크드 리스트 (linked list)

```
insert_rank(name, score)
       make new node with name, score, null link
       if (linked list is empty)
          new node becomes head node
       else if (linked list has only one node)
          compare the score
          insert the new node into appropriate position and adjust links
       else if (linked list has more than two nodes)
          compare the score with the nodes that already exists
          insert the new node into appropriate position and adjust links
```

시간 복잡도: O(n), 공간 복잡도: O(n)

```
delete_rank(rank)
    if (rank is out of range)
        print error message
    else if (you want to delete head node)
        delete head node
        set the second node as the head node
    else
        find the correct position according to `rank`
        delete the node and adjust links
```

시간 복잡도: O(n), 공간 복잡도: O(n)

3. 생각한 각 자료구조에 대해서 어떻게 정렬된 랭킹(x~y위)(x≤y, x, y는 정수)을 얻을 수 있을지에 대해서 생각해보고, 그 방법에 대해서 pseudo code를 작성하고 시간 및 공간 복잡도를 계산한다.

(1) 배열 (array)

```
rank()
    get x and y from the user
    if (x and y are out of range)
        print error message
    else
        if (user didn't write x) x = 1
        if (user didn't write y) y = length of the array
        for (each node from x-th node to y-th node)
            print name and score
```

시간 복잡도: O(y - x)이므로 O(1), 공간 복잡도: O(1)

(2) 링크드 리스트 (linked list)

```
rank()
    get x and y from the user
    if (x and y are out of range)
        print error message
    else
        if (user didn't write x) x = 1
        if (user didn't write y) y = length of the linked list
        for (each node from x-th node to y-th node via link)
            print name and score
```

시간 복잡도: O(y - x)이므로 O(1), 공간 복잡도: O(1)