9주차 결과보고서

전공: 기계공학과 학년: 3학년 학번: 20191820 이름: 김형준

**1.**

자료구조 중 Sorted Linked List를 사용했다. 노드의 구조는 아래와 같이 이름을 저장하는 name, 점수를 저장하는 score, 순위를 저장하는 rank, 다음 노드의 주소를 저장하는 next로 구성되어있다.

typedef struct \_Node{

char name[NAMELEN+1];

int score;

int rank;

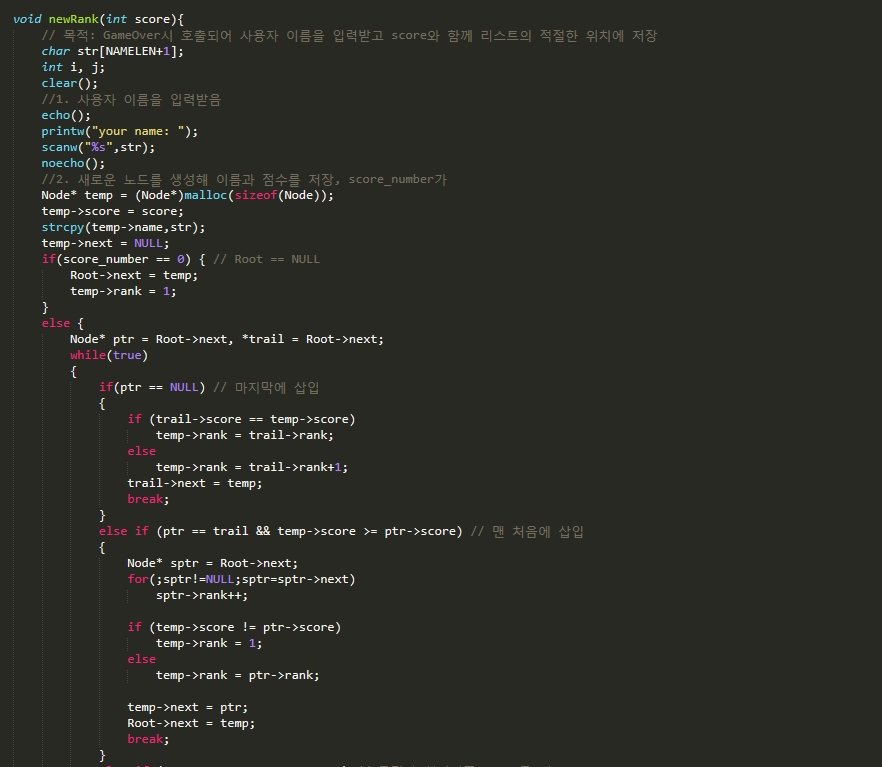
struct \_Node \*next;

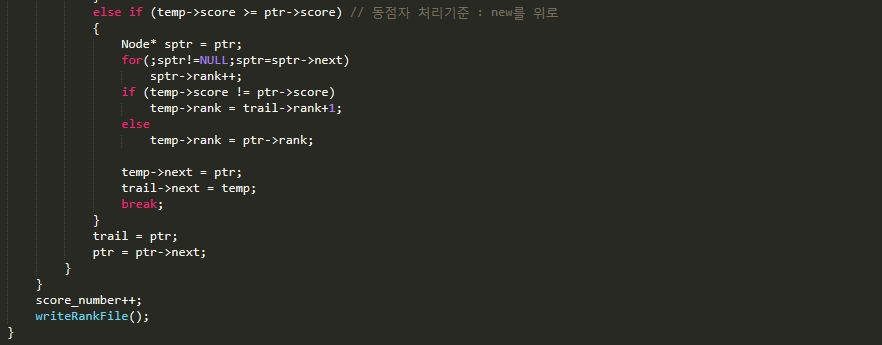
} Node;

연결리스트는 다음과 같은 두 개의 전역변수로 접근할 수 있는데, score\_number는 연결리스트에 저장된 노드의 갯수를(헤드 노드 미포함), Root는 연결리스트의 헤드 노드를 주소를 저장한다. (헤드 노드는 프로그램 시작시 할당되며 별도로 저장된 정보는 없음)

Node\* Root = NULL;

int score\_number = 0;





1st 평가 기준 (GameOver 이후 랭킹 등록) :

(공통사항) 새 노드를 할당받고, 그 노드에 입력받은 데이터를 모두 저장한다.

연결리스트에 헤드 노드만 존재하는 경우 헤드 노드 뒤에 노드를 삽입하고, 삽입할 노드의 rank를 1로 설정한다.

연결리스트에 헤드 노드 이외의 노드가 존재하는 경우, 연결리스트의 처음부터 끝까지 순회하면서 노드를 삽입할 위치에 도달할 때까지 반복하다가 도달하면 현재 위치에 노드를 삽입한다. (trail : 이전노드, ptr : 현재 노드, temp : 삽입할 노드, 삽입후 trail, temp, ptr 순서가 됨)

연결리스트의 마지막에 노드를 삽입하는 경우, 삽입할 노드의 점수가 이전 노드의 점수와 같은 경우 rank를 이전 노드의 rank값으로 설정하고, 다르면 rank+1로 설정한다.

연결리스트의 맨 처음에 노드를 삽입하는 경우, 연결리스트의 모든 노드의 rank값을 1 올리고 삽입할 노드의 점수와 현재 노드의 점수가 다르면 rank에 1을, 같으면 rank에 현재 노드의 rank를 대입한다. (동점자 처리)

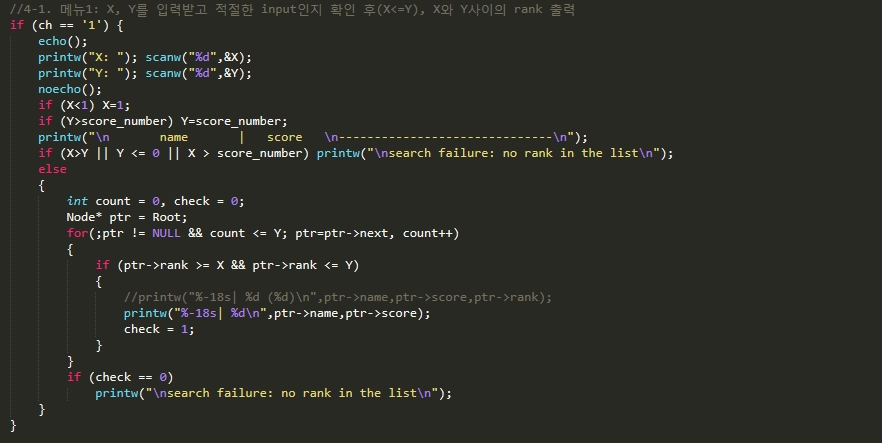
연결리스트의 중간에 노드를 삽입하는 경우, 현재 노드부터 마지막 노드의 rank를 1 증가시킨다. (새로 삽입되는 노드는 동점인 경우 동점자중 맨 앞에 기록되므로 삽입 노드와 동점인 노드들은 항상 삽입 노드보다 뒤에 기록되게 된다.) 삽입할 노드의 점수와 현재 노드의 점수가 다르면 이전 노드의 rank+1을, 같으면 현재 노드의 rank를 대입한다.

마지막으로 score\_number를 1 증가시켜 연결리스트의 노드 개수를 갱신한다.

현재 연결리스트에 존재하는 노드의 개수가 n개라고 하면 시간 복잡도는 O(n)이고, 공간복잡도는 O(1)이다. (호출당 최대 1개의 노드를 생성함)



(rank 함수에 대해서 X와 Y는 각각 1과 score\_number로 초기화 되어있음)



2nd 평가기준 (모드 1, X등부터 Y등까지 출력) :

X값과 Y값을 scanw로 입력받고, (미입력시 각각 초기화값인 1과 score\_number를 가짐)

X가 1보다 작으면 1로, Y가 score\_number보다 크면 score\_number로 값을 변경한다.

(X~Y 범위내에 출력 가능한 범위가 존재할 경우 출력 범위에 맞게 재설정함)

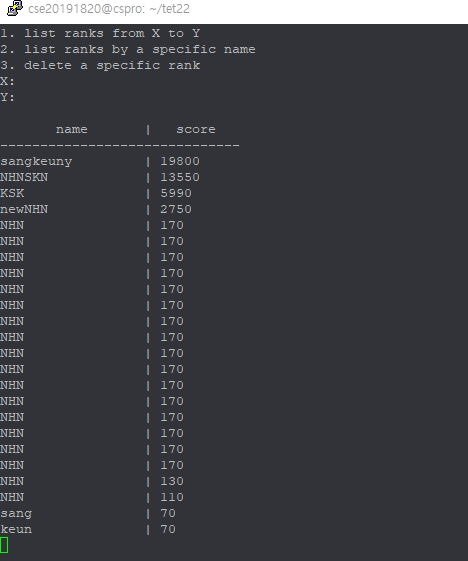
X~Y 범위내에 출력 가능한 항목이 존재하지 않는 경우 (X>Y, Y<=0, X>score\_number)면 에러메세지를 출력한다. 출력 가능한 항목 존재시 연결리스트의 처음부터 끝까지

연결리스트를 탐색하면서 현재 노드의 rank가 X~Y 이내의 값이면 해당 항목을 출력한다. 출력이 한번도 실행되지 않았으면 에러메세지를 출력한다.

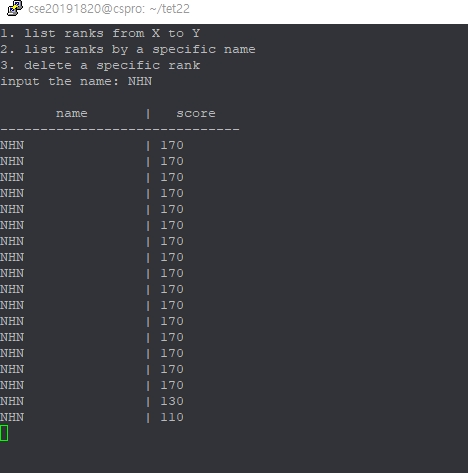
현재 연결리스트에 존재하는 노드의 개수가 n개라고 하면 시간 복잡도는 O(n)이고, 공간복잡도는 O(1)이다.

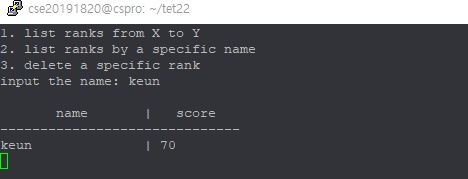
따라서 Sorted Linked List는 총 노드의 개수가 n개라고 하면 필요한 공간복잡도는 O(n)이고, 두 평가기준에서의 함수(삽입, 출력)의 시간 복잡도는 O(n)이고 공간복잡도는 O(1)이다. 정렬이 되어있지 않은 연결리스트를 사용할 경우에는 X등~Y등을 차례대로 출력할 때 rank의 값으로 정렬을 하거나 (O(nlogn)), 2중 루프를 돌면서 차례대로 순서에 맞는 rank에 해당하는 노드의 값을 출력해야 하므로 (O(n^2)) Sorted Linked List에 비해 비효율적이다. 또한 배열을 사용하는 경우, 값을 삽입할 때 최악의 경우는 배열의 맨 앞쪽에 삽입하는 경우인데, 이 경우에는 배열 전체 원소를 모두 다음 원소의 자리로 옮겨야 하므로 (O(n)), 시간복잡도는 Sorted Linked List와 같으나, 모든 원소를 옮기는 연산이 추가되어 실행시간의 비효율성이 증가하게 된다. 따라서 Sorted Linked List는 두 평가기준을 고려할 때, 효율적인 자료구조이다.

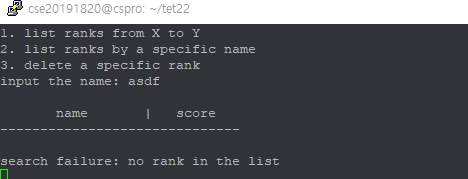
**2.**

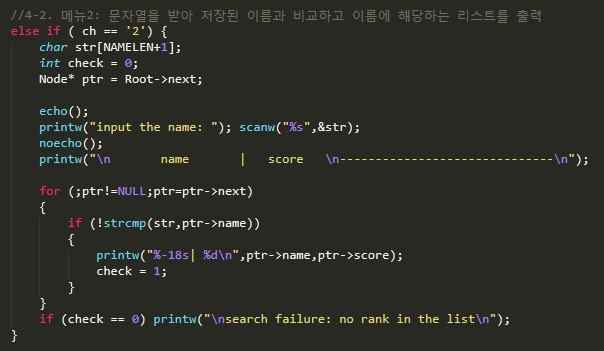


(rank.txt에 존재하는 모든 랭킹 리스트)









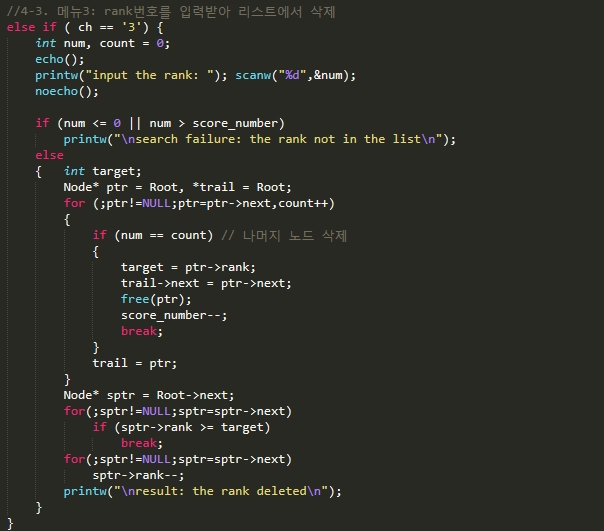
모드 2 (사용자가 입력한 이름에 해당하는 랭킹정보의 출력) :

scanw로 검색할 이름을 입력받고, 연결리스트의 시작부터 끝까지 순회한다. (ptr : 현재 노드)

순회하면서 현재 노드에 저장된 이름(ptr->name)이 입력된 문자열과 같으면 해당 노드의 값을 출력한다. 단 한 번도 출력되지 않았다면 에러메세지를 마지막에 출력한다.

연결리스트에 있는 노드의 개수가 n이라고 하면, 시간복잡도는 O(n)이고 공간복잡도는 O(1)이다.

**3.**



모드 3 (사용자가 입력한 순위에 해당하는 랭킹정보의 삭제) :

scanw로 삭제할 순위를 입력받고, 연결리스트의 시작부터 끝까지 순회한다.

(입력받은 순위가 연결리스트의 범위를 벗어난 경우 에러메세지를 출력하고 종료함)

(trail : 이전 노드, ptr : 현재 노드, target : 삭제할 노드의 rank값)

순회하면서 삭제할 노드에 도달하면 (num == count), target에 현재 노드의 rank을 저장하고, 현재 노드를 연결리스트에서 삭제한다.

(score\_number의 값을 1 감소시켜 연결리스트에 있는 노드의 개수를 갱신함)

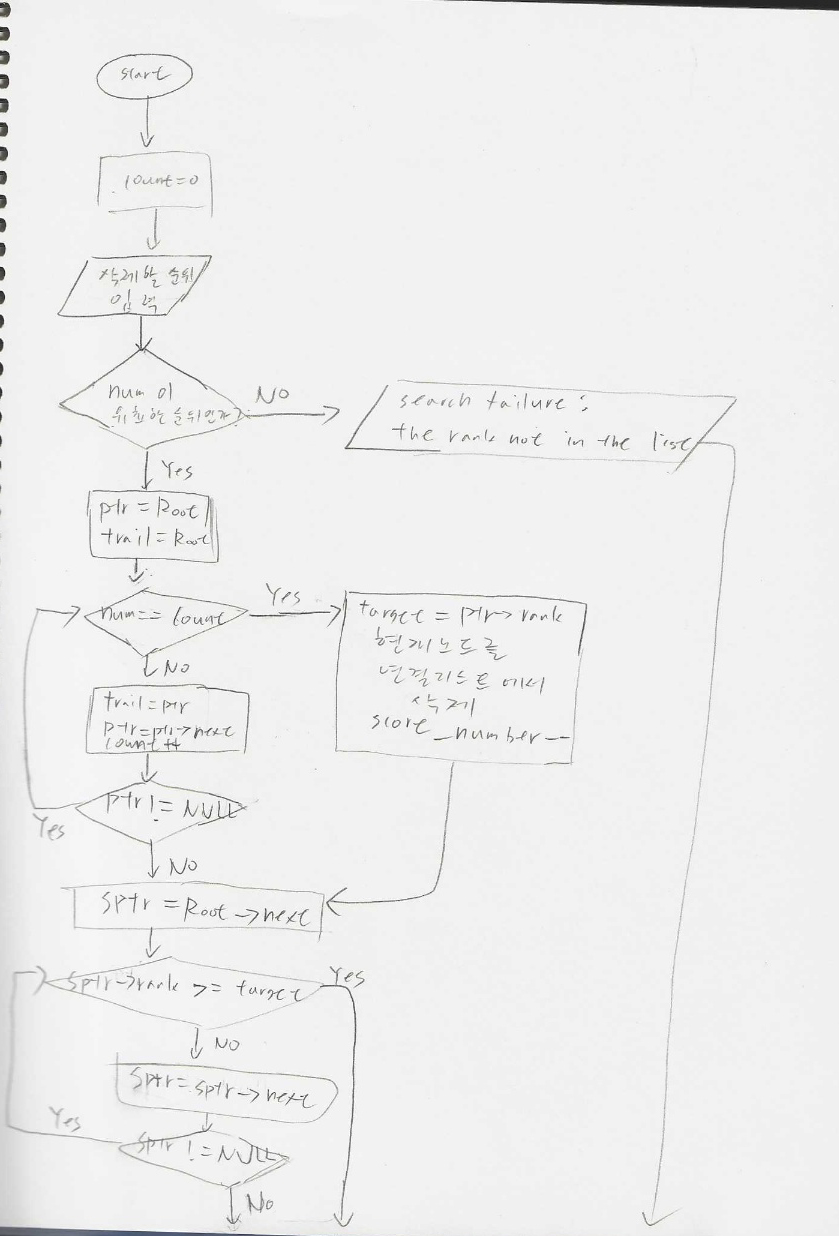
그 다음, 연결리스트를 다시 한번 처음부터 끝까지 순회하여 노드의 rank값이 target이상인 지점을 찾는다.

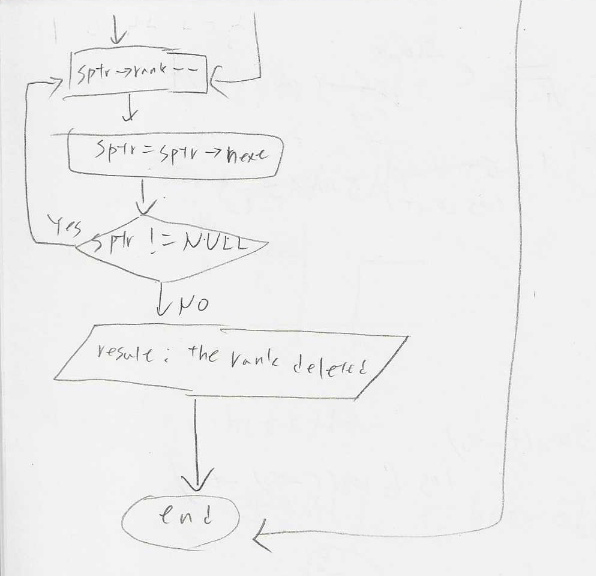
(순회 이후 sptr에는 sptr->rank>=target인 노드의 주소가 담기게됨, 이 경우 sptr~연결리스트의 마지막 노드까지의 rank가 변경되어야 한다.)

마지막으로 sptr부터 시작하여 연결리스트의 끝까지의 노드의 rank값을 1 감소시킨다.

(이에 해당하는 노드들은 삭제된 노드와 동순위거나, 밑 순위이기 때문에 순위가 1씩 올라가야함)

연결리스트에 있는 노드의 개수를 n이라고 하면 시간복잡도는 O(n)이고, 공간복잡도는 O(1)이다.



****

**4.**

본 실험 및 숙제를 통해 Linked List와 Array에 자료를 삽입, 삭제, 출력 하는 것을 스스로 구현해보면서 해당 자료구조를 사용하기 위해 어떻게 코드를 작성해야 하는지에 대해서 자세하게 알게되었다. 또한, 동점자 처리 기능을 구현하기 위해 노드의 멤버변수에 rank항목을 추가하고 랭킹 시스템의 각 함수에 rank값을 갱신하는 기능을 추가하면서 연결리스트를 좀 더 익숙하게 다룰 수 있게 된 것 같다.