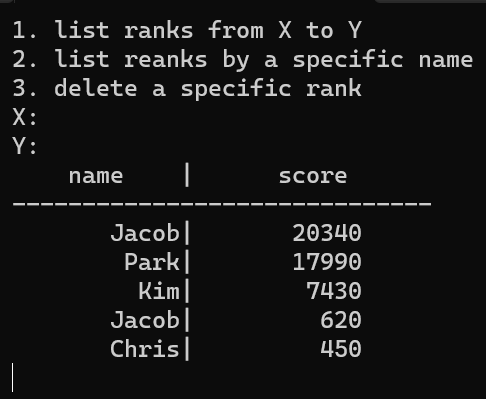
9주차 결과보고서

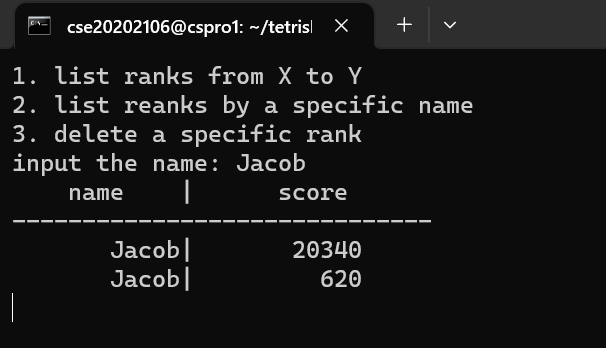
전공: 컴퓨터공학과 학년: 4학년 학번: 20202106 이름: OSHIMA ASUKA

1. 랭킹의 요소수가 작을 때는 배열과 Linked List의 효율성은 데이터 사이즈가 작기 때문에 시간에 큰 영향을 주지 않고 큰 차이가 없다. Linked List의 경우 삽입이나 삭제할 요소까지 회하면 되지만 목적으로 하는 요소를 찾기 위해 모드 Linked List를 회하는 필요가 있어 시간 복잡도는 O(n)이다. 배열은 삽입이나 삭제가 실행 된 위치 이후의 요소를 모드 시프트할 필요가 있기 때문에 시간복잡도는 O(n)이다. 두개의 자료구조는 모두 최악의 경우 시간 복잡도는 같기 때문에 상황에 따라 결정하는 것이 좋다고 생각한다. 랭킹시스템의 요소개수가 작고 삽입과 삭제 빈도가 그다지 높지 않는 경우 배열을 이용하면 깔끔하고 효율적으로 구현할 수 있다. 하지만 랭킹 요소수가 많아지고 새로운 랭킹 정보의 삽입이나 삭제의 빈도가 많다고 과정한다면 LinkedList를 이용하는 것이 좋다. 이번 실습에서 배부된 rank.txt는 요소수가 10보다 많고 게임 사용자가 늘수록 데이터도 많아지는 것을 고려하고 LinkedList를 이용했다.

아래에 과제내용 코드를 작성하여 실행한 결과 화면을 첨부하도록 한다. 먼저 그림1은 기존 랭킹파일이다. 과제에서 작성한 랭킹 시스템 모드2를 입력하여 표시하고자 하는 사용자 이름(Jacob)를 입력한 결과는 그림2와 같다. 사용자 이름을 검색할 때 그 이름이 두개 이상 있는 경우가 있기 때문에 모두의 LinkedList를 회하는 필요가 있다. 따라서 시간 복잡도는 랭킹수 n라 하면O(n)이다.

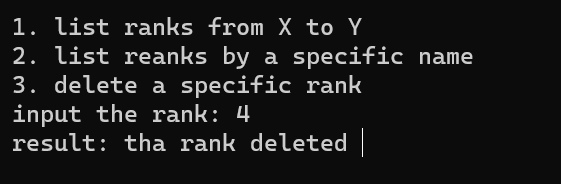
****

그림



그림

랭킹 시스템모드 3의 결과는 그림3 그림4부터 확인할 수 있다. 그림3은 삭제하고자 하는 랭킹 번호를 입력하는 화면이고 그림4는 삭제후의 랭킹리스트의 결과다.

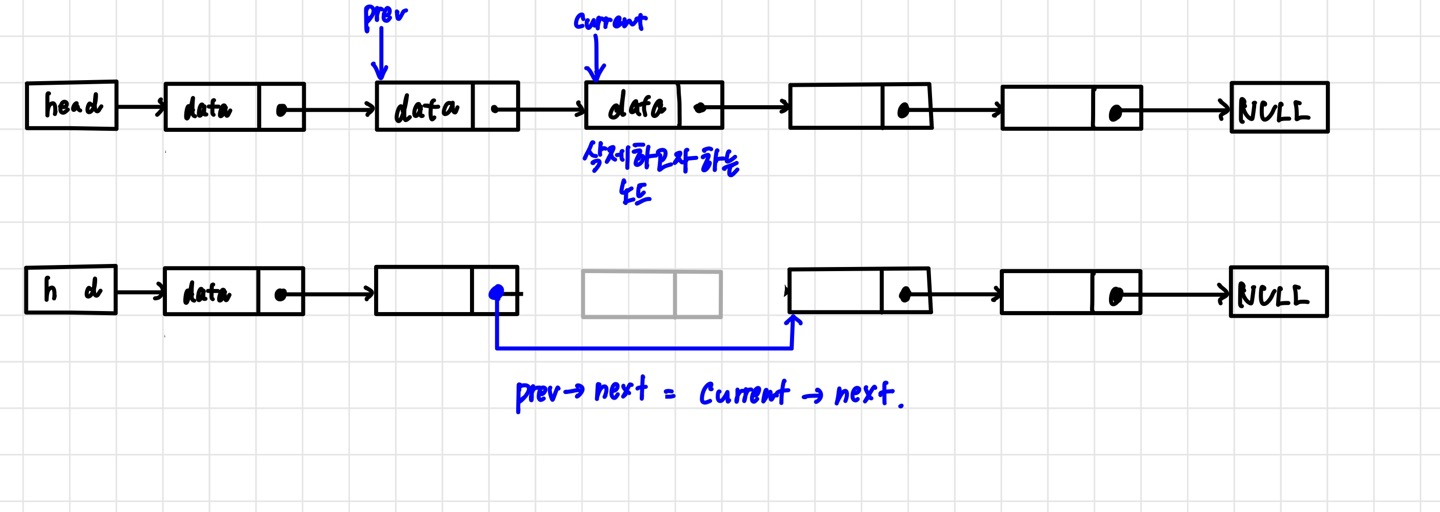


그림



그림

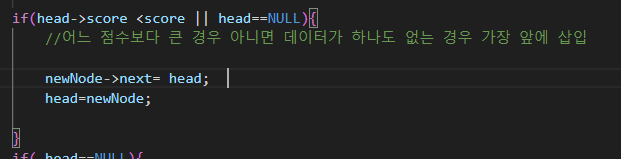
그림5는 삭제하는 알고리즘을 그림으로 표현하는 그림이다. 삭제하고자 하는 노드의 하나 앞에 노드의 위치의 포인터를 prev라하고 prev->next에다가 삭제한 노드의 뒤에 이어지는 linkedlist 구조를 붙인다.



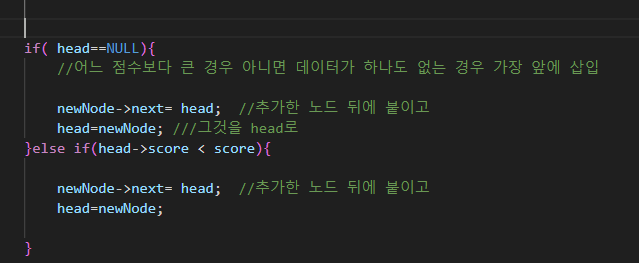
그림

1. 본 실험 및 과제를 통해서 습득한 내용

구현하고자 하는 프로그램의 접근 방법이 여러 개 있는 경우 어떤 자료구조가 효율적인지를 모둔 상황을 고려하여 가장 효율적인 방법으로 구현해야 한다. 실습에서는 rank.txt파일이 없는 상태에서 게임을 플레이 하고 랭킹 정보를 저장하는 과정에서 segmentation fault 가 나왔었다. segmentation fault는 존재하지 않은 메모리 영역에 접근을 시도했을 때 나오는 error이다. 수정전은 그림6 수정후는 그림7이고 조건문을 바꿈으로 해결되었다. 그림6에서는 조건문에서 head값이 NULL일 때도 멤버에 접근하는 가능성이 있다. 이것을 조건문을 나눠 head값이 NULL일 때 메모리에 접근하지 않도록 코드를 수정함으로 segmentation fault은 해결되었다.



그림



그림