**운영체제 과제2  
Concurrency Hash Table**

**과제 설명**

본 과제에서는 OS 부교재 OSTEP **“29.Lock-based Concurrent Data Structures**”의 **Concurrent Hash Table**을 구현한 뒤 데이터 입력, 검색, 삭제 등의 기능을 구현합니다. Hash Table의 각 Bucket은 Concurrent Linked List이며, 각각의 List마다 mutex를 갖는 구조입니다. 해시 **테이블의 크기는** **10007**이며, 해시 테이블에 넣으려는 key와 item은 모두 정수입니다. 쓰레드는 **4개**를 사용합니다. **Key**를 bucket의 **index**로 변환하는 **hash function**은 주어진 key를 10007로 나눈 나머지를 구하는 함수로 구현합니다.

사용자에게 4개의 파일이 주어집니다. 각 파일의 이름은 input0.txt, input1.txt, input2.txt, input3.txt이며, 사용자는 4개의 쓰레드를 이용하여 각 쓰레드마다 파일 하나를 읽어야 합니다. 파일에는 key와 item이 공백으로 구분되어 적혀 있으며, 쓰레드는 key와 item을 읽어와서 Concurrent Hash Table에 데이터를 입력합니다. 쓰레드 4개의 입력이 모두 끝나면 파일 4개로 주어진 key, item을 입력하는데 걸린 시간을 측정하여 출력합니다. **측정 코드 예시는 아래에 있으니 참고하여 과제를 수행해주시길 바랍니다.**

데이터 입력 후의 입/출력은 다음과 같습니다.

**INPUT:** **1** 3 5 // 1은 key와 item의 삽입을 나타내고, 3과 5는 각각 key, item.  
**OUTPUT:** 0 // 출력으로는 새로운 데이터를 입력 시 0  
 // 데이터가 이미 존재하여 업데이트 했을 시 1

데이터 입력 시, 계산된 bucket 의 List 에 삽입하게 됩니다.  
새로운 (key, item) 인 경우, 해당 List 에서 item 크기로 오름차순 했을 시 알맞은 위치에 삽입  
기존의 (key, item) 을 업데이트하는 경우, 업데이트한 item 크기에 맞는 위치에 삽입

즉, 새로운 아이템을 삽입하는 경우에도 무조건 head에 삽입하는 것이 아니라, List의 목록을 한번 보면서 item 크기에 맞는 위치를 찾아서 삽입하게 됩니다.

***한 List 에 동일한 크기의 item은 주어지지 않습니다.***

**ex. 기존의 0번 bucket에 (0, 3) 이 들어있다고 가정했을 때,**

**1 10007 5 일 경우, (10007, 5) 는 (0, 3) 뒤에 들어갑니다. (0, 3) – (10007, 5)  
1 10007 1 일 경우, (10007, 1) 은 (0, 3) 앞에 들어갑니다. (10007, 1) – (0, 3)**

**INPUT: 2** 3 5 // 2은 key와 item의 위치 검색을 나타냄. 3은 key, 5는 item  
**OUTPUT:** 3 0 // (3, 5)아이템의 hash table의 bucket 번호는 3,  
 // 3번 bucket의 List를 순회하며 발견한 위치는 0.  
 // 데이터가 없을 경우 -1만 출력

**INPUT:** **3** 3 // 3은 주어진 key의 삭제를 나타냄.  
**OUTPUT:** 0 // 성공적으로 데이터 삭제 시 0 출력  
 // 삭제할 데이터가 없을 경우 -1을 출력

**INPUT:** 0 // 0 입력 시 프로그램 종료

**예제 및 출력 형식 (2018/06/01 수정)**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **input0.txt** | | **input1.txt** | **input2.txt** | **input3.txt** |
| **0 2**  **10007 3** | | **4 1**  **10011 2** | **8 1**  **10015 2** | **12 3**  **10019 4** |
| **입력** | 1 2 2  1 10007 1  2 0 2  2 10007 1  3 10007  3 10008  0 | | | |
| **출력** | 0.001030 // input 0, 1, 2, 3 txt를 모두 hash table에 입력하는데 걸리는 시간  0  1  0 1 // 삽입된 데이터의 위치는 위의 설명 참조  0 0  0  -1 | | | |

* **주의 사항**
* **List 와 Hash table 구현에 STL 사용을 금지합니다. (List, map 등등)**  
  OS는 **Windows, Ubuntu (OS 실습 서버)**로 제한합니다.
* 언어는 **C, C++**만 허용하며 소스 코드에 간단하게 **주석**을 달아주세요.
* **표준 입출력**을 사용합니다. (printf, scanf / cout, cin)
* **보고서**는 **구현 환경** (OS, IDE, 언어), **함수 설명, 실행 화면 사진**을 포함하여 작성해주세요.
* 과제 제출 기한은 **6월 17일 오후 11:59분**까지 입니다.
* 다른 사람의 과제를 베끼거나 인터넷의 코드를 그대로 사용할 경우 과제 점수 0점 및 학점에 영향이 있을 수 있습니다.
* **프로젝트 전체 폴더**(윈도우) 또는 **소스코드 및 실행파일**(우분투)과 **보고서**를 압축하여 업로드해 주세요. 파일 이름 형식은 다음과 같습니다.

**(윈도우)프로젝트: OS2\_학번  
(우분투)소스코드 및 실행파일: OS2\_학번.확장자, OS2\_학번**

**보고서 파일: OS2\_report.확장자**

**압축 파일: OS2\_학번.zip (해당 압축파일을 제출)**

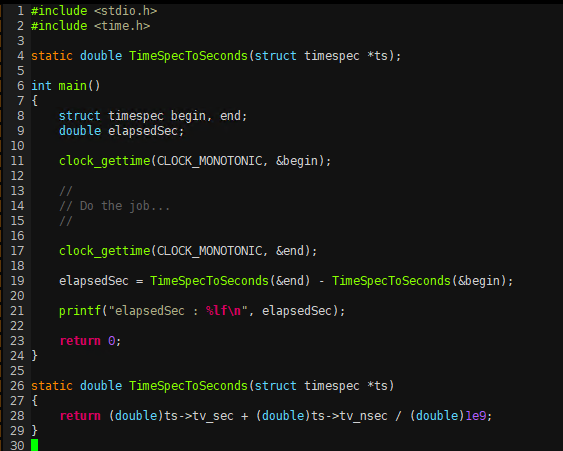
부교재의 코드는 참고용으로만 사용하시고  
과제의 요구사항을 중점으로 코드를 작성해 주세요.

**주의 사항 미 준수 시 감점이 있을 수 있습니다.**

(빼먹은 주의사항, 혹은 과제의 오류 등에 대해서 추가적인 수정이 있을 시, 과제란에 수정 시각과 수정 내용을 올리겠습니다. 이상한 점이 있을 경우 확인해보시고, 수정본이 없다면 조교 메일 또는 질문 게시판에 문의해주시기 바랍니다.)

**추가 사항**

1. 시간은 다음과 같은 코드로 측정합니다.

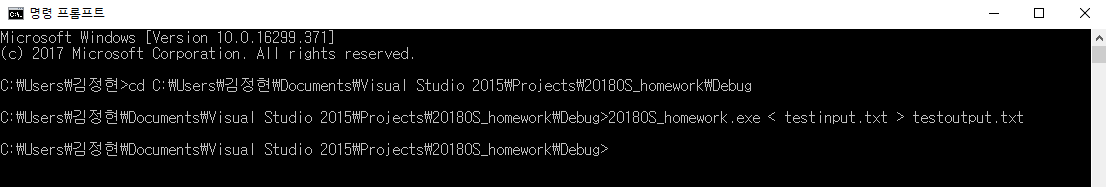


2. 리눅스의 경우 컴파일 시 옵션에 **-lpthread** 를 포함해야 합니다.



3. 테스트 케이스 파일은 다음과 같은 방법으로 표준 입출력을 통해 넣을 수 있습니다.

**(윈도우) 실행파일.exe < testinput.txt > testoutput.txt**



**(리눅스) ./실행파일 < testinput.txt > testoutput.txt**

