CSCI 3357: Database System Implementation

Homework Assignment 3

Due Friday, September 20

The SimpleDB buffer manager is grossly inefficient in two ways:

- When looking for a buffer to replace, it uses the first unpinned buffer it finds, instead of doing something intelligent like LRU.
- When checking to see if a block is already in a buffer, it does a sequential scan of the buffers, instead of keeping a data structure (such as a map) to more quickly locate the buffer.

I would like you to fix these problems by modifying the class `BufferMgr`. Please use the following strategy:

- Keep a list of unpinned buffers. When a replacement buffer needs to be chosen, remove the buffer at the head of the list and use it. When a buffer's pin count becomes 0, add it to the end of the list. This implements LRU replacement.

- Keep a map of allocated buffers, keyed on the block they contain. (A buffer is allocated when its contents is not null, and may be pinned or unpinned. A buffer starts out unallocated; it becomes allocated when it is first assigned to a block, and stays allocated forever after.) Use this map to determine if a block is currently in a buffer. When a buffer is first allocated, you must add it to the map. When a buffer is replaced, you must change the map—the mapping for the old block must be removed, and the mapping for the new block must be added.

- Get rid of the `bufferpool` array. You no longer need it.

In addition, you should modify the class `Buffer` so that each `Buffer` object knows which buffer it is. In particular, its constructor should have a third argument denoting the buffer's id, and a method `getId()` that returns that id.

You should also modify `BufferMgr` to have a method `printStatus` that displays its current status. The status consists of the id, block, and pinned status of each buffer in the allocated map, plus the ids of each buffer in unpinned list. For example, here is what the output of the method should look like for a database having 4 buffers, in which blocks 0 to 3 of file "test" were pinned, and then blocks 2 and 0 were unpinned.

```
Allocated Buffers:
   Buffer 1: [file test, block 1] pinned
   Buffer 0: [file test, block 0] unpinned
   Buffer 3: [file test, block 3] pinned
   Buffer 2: [file test, block 2] unpinned
Unpinned Buffers in LRU order: 2 0
```

Note that the buffers in the above output are in seemingly random order because they were retrieved from a hash map. The information within brackets comes from calling the `toString` method of `BlockId`.

I have written a test program called `HW3Test`, which pins and unpins buffers, occasionally calling the buffer manager's `printStatus` method. You should download it. It will help you debug your code.