**Programming Least Recently Used Bufferpool Assignment Instructions**

**Overview**

Implement a disk-based buffer pool class based on the Least Recently Used (LRU) buffer pool replacement strategy.

**Instructions**

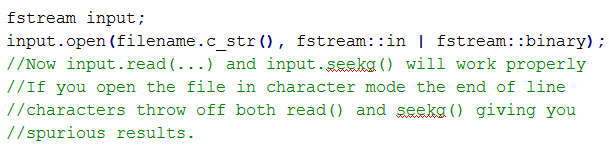
Implement and demonstrate a disk-based buffer pool class based on the LRU buffer pool replacement strategy. Disk blocks are numbered consecutively from the beginning of the file with the first block numbered as 0. Assume that blocks are 4096 bytes in size. Use the supplied C++ files to implement your LRU Buffer Pool based on the instructions below.

* Implement a BufferBlock class using the supplied BufferBlockADT.h
  + Your Buffer Block must inherit BufferBlockADT or you will not get credit for your work.
  + All BufferBlockADT virtual functions must be implemented in BufferBlock
  + Block Size: 4096 bytes
  + Add an instance variable to your buffer block implementation to store the block id; i.e., “int blockID.”
* Implement a Buffer Pool by inheriting BufferPoolADT (BufferPoolADT.h) – implement all of BufferPoolADT’s functions (if you do not inherit BufferPoolADT you will not get credit for your work).
* Your buffer pool should consist of 5 buffer blocks
* Your buffer pool should manage the buffers using the LRU strategy
* Your buffer pool should be named LRUBufferPool and the file containing the LRUBufferPool class should be named LRUBufferPool.h
* Use the provided main.cpp and the included test file mydatafile.txt to test your program.

## File Provided:

* BufferBlockADT.h
* BufferPoolADT.h
* constants.h (contains both constants and test function definitions)
* main.cpp (test program driver)
* mydatafile.txt (input file for the test run)
* Buffer Pool Output.txt – if your program is correct your output should look like this

## Note: You must open the input file in binary mode for the read() and seekg() methods to work properly. See example code below.



## Testing

Use main.cpp as the test driver and mydatafile.txt as the input file.

## Assignment Submission:

Put the following files into a zip file and submit your assignment to the assignment link in Canvas:

* All your source code files (.cpp and .h files), including the files provided for the assignment. Note that none of the files that have been provided can be changed (depending on how you created your project in Visual Studios you may need to add #include “stdafx.h” to main.cpp to get it to compile, which is a permitted change).
* If you used Visual Studios to build your project, please give me your entire project folder.
* Word document describing your approach to solving this problem.
* Word document with screen shots showing results of your test run (I need to see the entire run, so give me as many screen shots as necessary) and integrity statements.
* Executable file (.exe file).

Put all your source files, test files, executable, and Word document(s) into a zip file and submit it to Canvas.