Lab 3 Approach Document

# LRU Buffer Pool

# Assignment Objective

# Implement a disk - based buffer pool class based on the LRU buffer pool replacement strategy.

# Assignment Requirements

Your system should have the ability to

* You must open the input file in binary mode for the read() and seekg() methods to work properly
* Implement a BufferBlock class using the supplied BufferBlockADT.h
* Implement a Buffer Pool by inheriting BufferPoolADT

# Approach

* Review all files (documents, code, helps, etc.) provided as part of the assignment
* Select a data structure and implementation that could satisfy this assignment:
  + Implement a BufferBlock class using the supplied BufferBlockADT.h
    - “LRUBufferBlock.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
    - 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).
    - “LRUBufferPool.h”
    - Your buffer pool should consist of 5 buffer blocks (your buffer pool constructor should load your buffer pool with the first 5 blocks of your file and the initial order of your buffer pool blocks should be 0, 1, 2, 3, 4).
    - Your buffer pool should manage the buffers using the LRU strategy
    - Your buffer poo l should be named LRUBuffer Pool and the file containing the LRUBuffer Pool class should be named LRUBuffer Pool.h Build Log
    - Create a dynamic array of Buffer Blocks (set to size 5 in constructor)
    - Buffer Pool private variables
      * The buffer pool consists of X number of buffer blocks (using dynamic array) but uses POOL\_SIZE global variable to prevent it from going out of bounds
    - Constructor
      * Constructor gets the filename of the file to be buffered, opens the file, and instantiates poolSize buffer blocks by reading the file and filling the blocks in order. When the constructor is done the buffer pool blocks should be full with the beginning contents of the input file.
    - getBytes()
      * Get the starting index by rounding down to the nearest block
      * Get the data from the start block and check if the ID for the block is in the pool
        + If the ID is found, mark the index in a variable and use a Boolean variable to execute functions in separate statement
      * If the ID is found, then execute the following functionality (**LRU USED HERE**)
        + Create a temporary variable of the char array being moved to the front of array
        + Move down blocks from beginning to make space at the front of the list for LRU
        + Set the first block to the temp block
      * If the ID is **not** found
        + Move down blocks from beginning to make space at the front of the list for LRU
        + Reopen the file to get the new data not in the pool (redo of what’s done in the constructor)
        + Creating temp char array to hold the data
        + Instantiate the buffer block and Set the ID of the block to the start of index block
      * Get the data from the end block

# Build Log

10/19/23 – Today I finished going over all the assignment documentation and other files and have created my approach document up to this point. I created the project folder and imported the required files from Dropbox. Working through this with Cameron Kauffman

10/20/23 – Met with Cameron and finished BufferPool and BufferBlock first drafts.

10/24/23 – Met with tutor to fix issues with LRU pool and getBytes function. Got program fully working and need to meet with Cameron to go over changes made during tutoring.

10/26/23 – Met with Cameron to go over changes made and make final submission.