DB Project 2 Report (SimpleDB):

Task 1: G Clock replacement policy:

Startup.java -

1. Added new argument to read the reference count value from user.
2. Added code to default to "studentdb" directory and "5" if user starts without any parameters.

simpleDB.java -

1. Changed prototype of init function to read the reference count value for the clock replacement policy.

Buffer.java-

1. Added parameterized constructor to Buffer to initialize G Clock reference count value.
2. Added privae variables to keep track of maxRefCountValue and refCount.
3. When a buffer is pinned for the first time, the refCount is reset to -1.
4. When a buffer is unpinned by everyone, the refCount is set to maxRefCountValue.

BasicBufferMgr.java -

1. Constructor now calls SimpleDB method to get the maxRefCountValue, and initializes all buffers with that value.
2. The class also keeps track of the current clock hand position.
3. ChooseUnpinnedBuffer() is modified to start from the current clock hand position, and rotate at most (maxRefCountValue + 1) times. If a unpinned buffer has refCount > 0, it is decremented, and if it already 0, it is sent for replacement.
4. Method added to print buffer parameters as asked by TA.

Task 2: Using a map data structure to keep track of the map

BasicBufferMgr.java -

1. A map reference "bufferPoolMap" of type hashmap has been added
2. The function flushAll(int txnum) is modified to implement an iterator to loop over the map and delete entries corresponding to the given transaction number
3. The pin(Block blk) and pinNew(String filename, PageFormatter fmtr) funtions have a put method to add the given <Block,Buffer> pair to the map
4. The findExistingBuffer(Block blk) function gets any existing buffer of the Block in the map by using the get method of the map
5. The chooseUnpinnedBuffer() is modified, To find a Buffer which has not yet been allocated to the map and To remove an entry form the map based on the block value.
6. containsMapping(Block blk) - function added to check the working of the map
7. getMapping(Block blk) - function added to check the working of the map

BufferMgr.java -

1. containsMapping(Block blk) - function added to check the working of the map
2. getMapping(Block blk) - function added to check the working of the map

Task 3: Using blocks as elements of recovery:

Buffer.java -

1. Variable added to keep the buffer aware of it's index in the BasicBufferMgr array. (Value initialized during creation)
2. SaveBlock method added. If setInt or setString is called on an unmodified buffer, the saveBlock method saves a copy of the page by calling the saveBlock method. This saves the buffer on file by using the "page" classes' write method with a custom Block object. The page contents are saved at an offset of "index * BLOCK_SIZE" in the backup file.
3. RestoreBlock method added. This is called during transaction undo. The offset into the file is received as follows:

"The UPDATE RECORD creator buffer's index * BLOCK_SIZE". The other params of buffer are initialized as well.

The read from file is done using page's filemgr read method with a custom Block.

LogRecord.java -

1. static final variable was added for update record with value = 6

LogRecordIterator.java -

1. next() method updated to handle a record of type Update

RecoveryMgr.java -

1. The setInt and setString methods were modified to write a record of type UPDATE to the log

UpdateRecord.java -

1. New Class added to for the record type of update having the following variables
   txnum - the transaction number
   offset_blk - the offset of the block in the original file
   offset_bckpFile - the offset of the block in the backup file
   filename - the filename where the block is to be replaced
2. There are two constructors
   UpdateRecord(int txnum, Buffer buff) to create the record from the transaction number and buffer
3. UpdateRecord(BasicLogRecord rec) to create a new record by reading an existing record from the log
4. writeToLog() write a record to the log in the format : "UPDATE, txnum, filename, offset_blk, offset_bckpFile"
5. op() returns the integer value of the commit record type
6. txNumber() returns the transaction number of the given record
7. toString() returns a string representation of the UpdateRecord Class in the format
   "<UPDATE txnum filename offset_blk offset_bckpFile>"
8. undo(int txnum) Restores the value of the saved block from the backup file for the given transaction

Notes:

1.Buffer contents are printed as requested.

a.For a buffer (index starting at 0):

Pin count tells whether the buffer is currently pinned or unpinned.
Ref Count >= 0 indicates block contents are valid (even if buffer is unpinned)
Ref Count = -1 indicates that block is currently empty.

b.Num available displays the number of unpinned buffers (not necessarily empty buffers)