As we begin yet another project, I am for once at a huge loss for where to start. While previous programs have used data structures kept completely in memory, the very basics of this program will require that I am far more up to date on C++ style file processing than I am currently. My plan to proceed with this project is as follows.

First, I will need to review my previous code from CSIS 111 and 112 on the basics of C++ ins and outs of files when reading into memory. When I am done with that I will need to find how to extend this knowledge to binary mode, as this is a mode that I have not worked with before and may prove to be more difficult. To do that, I will likely need to look into cppreference.com as this website keeps good up to date information and guides maintaining its status as the main place to refer to on C++ projects.

After I have familiarized myself with the how to bring data in from a file, I will need to get started on implementing the buffer block ADT. This is the first component of the final project and will be required to work before I can move on with the buffer pool, as the pool is simply a collection of blocks and the accompanying management software. To do this, all of the virtual functions as well as the constructor and destructor need to be implemented properly. Since these blocks do not know how they will be ordered, they will need IDs so that they can be identified on what their contents are. These contents should come directly from the disk, 4096 bytes at a time, however that is something that need to happen at the pool level. When designing the blocks, I will need to only keep in mind the char\* organization of the data. I believe that I will be able to adapt a similar algorithm that was I already designed in CSIS 112 to insert the data into a char\*. That program also made use of external files, though not one nearly as large as the provided mydatafile.txt. The remaining getter and setters should be trivial after setting the char\*.

The next thing I will need to do is implement the functions from the buffer pool ADT. The constructor should be easy with an array of blocks. I want to use smart pointers here so that I do not have to worry about the memory as much as well as to get me more practice than the first assignment gave us. I still feel that I have room to improve in my understanding and use of this vital tool. Next is the getByes() function. This will likely be the hardest part of the program as it is where the management of exactly enough data from disk needs to be taken in binary form. While setting a limit itself should not be hard, thinking in terms of binary length instead of character length will make modeling the pool more demanding. That said, I will have next to no work after getBytes() is finished as the remaining virtual functions are simply a print and a get function. Keeping that in mind, I am going to dig extra hard into binary reads so that I do not struggle as hard when writing my algorithm.