Jonathan Chu
CSCI-402
February 22, 2013

Part B: Coarse

Design & Implementation

For this part of the project, the way that I allowed access into the database was to use a global mutex for the database. Threads that wanted to access the database would lock the mutex, blocking any other threads from accessing, and then they would do their operations. When designing the places that I would lock, I had to make the decision between locking within the db_coarse interpret_command, or within the separate functions that are called by interpret_command (i.e. query, add, xremove). I decided to use the latter. In order to do this, I had to identify which of the functions should the locking take place in. I concluded that since search, node_create, and node_destroy are used by the other functions, I would not lock within that function, only locking in query, add, and xremove.

The implementation is pretty simple, I declare and initialize a static mutex at the top, and that will be the mechanism to control access into the database. Inside the functions, I started each function by locking the mutex. If another thread is already operating, the current thread calling the lock would block until the other finishes, thus serializing access. After locking the mutex, the thread would carry out its operations, and unlock the mutex when it is done.

Known Defects

I tested this pretty extensively, and I have not found any defects.