Kyle Pellerin, Mati Gibbons
Sean Barker
CSCI 3325
Final Project Check-in #1

Server:

We started out by taking our code from project 2 that we used to create the frontend server and basically just copying it into a new file "main server" we then deleted all our methods besides main as well as any reference to the other servers we used in that project. We left mainly intact and set the port to be 8089 to keep in track with what we had done in our other projects. In thinking about how we wanted to handle files. We then created a hashmap that is going to store our list of what clients have what files. The way we did this was by assigning keys to be strings that will be filenames and having FileList objects (see data structures section below) to store the corresponding files (and eventually what parts of those files) a given peer had possession of. We then decided to add two simple methods for now. The first method we added was "register_files" which basically a peer will call when they connect to tell the server what files they have and this will update the file list on the main server (which will eventually be sent to the backup server). The second method we added was "search file" which returns if a file is contained by the peers and if so what peers contain it. We did some basic testing and were able to connect the peer to the server and input the list of files a given peer had in csv format.

Client: (peer)

To start working on our client we decided that we felt most comfortable working with the XML and RPC systems that we used in project 2. So taking our code from project two we used this as a skeleton model to begin our Peer.py file that would allow for each of our peers to connect to the server. We started by having the peer first print it up for usage later along with connecting to the main server. Right now we still have the IP address from the previous machines used in project 2. But we will switch this over to the new IP address' once we get those. Then, once connected to the main server, we query the peer to announce or register the files it is willing to share with the server and other clients. Then once this list has been given it will communicate this with the server and begin a while(true) loop that will wait for a command/request from the client. It can either decide to search for a file on the P2P system, or it can choose to act simply as a listener.

Data Structure (FileList):

We created an in-house, adaptive, statically typed, data structure called file list that stores a list of filenames as strings. This will eventually be expanded to contain what parts of each file a peer contains but for simplicity's sake now we don't do this. The data structure contains methods for adding, removing, and checking what files exist in the list.