CSCI 5150 **Fall 2014**
Homework 1 **Due: September 8**

This assignment involves creating a client and a server program. Your will write then using C++ and the g++ compiler. See the examples in `~barrettm/5150/examples`. One of the programs there shows how to process command-line parameters. Two others show how to set up a basic client and server, but each will need significant modifications. In addition, there is a makefile that compiles and links each program. Feel free to copy these examples to your directory space.

Your server program should take one command line parameter pair, `-p portnum,` to indicate which port number to use for its socket. Ports from 0 – 1023 are reserved, so you can try any port number higher than that. However, some of those ports will be in use – so write a port stepper that starts at `portnum`, checking for an error returned from `bind()`. If so, increment the `portnum` until you find one not in use. Display the actual port number used, so that the client program will know what to use. Next, accept connection requests from clients. When one arrives, create a thread that handles all read requests from that client until the client sends an "exiting" request. The integer returned from `accept()` must be passed to the thread function and used for any reads and writes there; do not use the value returned by `socket()`, but note that this is different in the client. The server main program should loop forever, waiting for `accept()` calls, so the server must be shut down manually – make sure that you do, in fact, shut it down after testing.

Your client program should take two command line parameter pairs: `-s servername` and `-p portnum`. The `servername` is the domain name of the server, like "einstein.etsu.edu". The `portnum` is the port number successfully bound in the server program; note that no port stepping is used on the client side. Reads and writes on the client side use the value returned from the `socket()` call, as noted above.

Each request will have the following message format; not all fields will be used in this assignment.
```
struct serviceRequest {
   char domainName[256];   // requester's domain name
   int  portNumber;        // requester's port number
   int  requestType;       // 0=new client, 1=client is exiting, 2=query, 3=response
   char requestString[32]; // Parameters of request
   int  requestID;         // Unique ID
   char payload[256];      // Other request information, to be determined
};
```

When a client sends the first request, the requestType will be 0, the domainName will be the client's domain name, and the requestID will be some integer value assigned by the client. The server must keep a list or table of the client connections (in later assignments, there will be more than one client at a time). For this assignment, the client will then send zero or more queries (type 2) until it sends a type 1 query to quit. The server should respond to each request from the client (type 3).

Design your client and server programs so that the communication infrastructure is in a separate module or class.

Email me your code at barrettm@etsu.edu. Zip up your source files only – no object or executable files, please – plus your makefile (but make sure your makefile does not contain references to your directory structure).