**CIS 3207**

**Professor Eugene Kwatney**

**CIS 3207**

**Lab 3**

To execute this program, just do make and then use ./spellChecker [optional port number] [optional dictionary] to execute the program and then use telnet localhost [portnumber] or nc localhost [portnumber] to start the connection and use the program. By default the number of workers has been set to 2. To change the number of workers, the Macro is on line 21 of the code called NUM\_WORKER.

**Main()**

The main function does all the house keeping in this software. It looks for arguments passed in. If the user did not pass in a port number or a dictionary as arguments, then the program chooses 3207 as the default port number and chooses the dictionary in the OS as the default dictionary. Later, the main function opens a dictionary and puts it into a vector. Then it opens the port number and created all of the necessary threads such as worker threads, log thread and connectionGetter thread.

**getConnection()**

This is the getConnection function, it runs on its own thread and keeps listening for incoming connection requests and assign a clientSocket port to it. The client sockets are then pushed into a sockets queue for workers to use. It keeps listening for new connenctions on a ontinious while loop

**Worker()**

This is the worker thread that takes a clientSocket and process their requests. At first this function looks for a client socket to connect to if the queue is empty, the client socket goes to sleep, otherwise it connects. It starts communication and send and recieve data. It takes a word from client and looks up the word in the dictionary. If it is there the comparing function returns the word concatanated with OK or Misspelled. When it is done (the client presses ESC character) the client closes the connection and goes to accept another client socket. If the client queue is empty, it goes to sleep.

**writeLog()**

This is the writeLog function that gets passed in to be executed in a thread. This function opens a file named log.txt and then read string from logQ and print it in the log.txt file.

**Compare()**

This function takes a string from the recvBuffer and take the vector reference of the dictionary. At first the function strip off any extra character such as \n, \0 \t from the string and then str.compare() to compare the string with every single word in dictionary. Based on the result the dictionary retruns the string concatenated with OK or Misspelled.