David Tian (dt474) & Rahul Boppana (rsb172)

Design:

As this was a completely different project than the last few, there were many things that we needed to figure out. We had a few design issues such as launching threads, making sure socket connections were all functional and some of the logic involved. One of the more complex logic problems that came up was how there might be multiple clients and they would not be able to access a shared value. We decided to make the server responsible for using mutexes to access the value and read or write to it as necessary.

Other than the necessary logic, this project involved learning about many new features of the C languages, including sockets, semaphores, and signals. We simply researched the code for these types of functions and adapted it to suit our program. For example a timer functionality for the 15 second signal interrupt was standard code with pre-defined structures and steps.

Implementation Basic Overview:

Our client program has 3 functions. The main method contains the code for setting up the socket connection and launches both the input and output thread. The input thread is launched with the userPrompt function and the output thread is launched with the serverResponse function. The input function contains a loop that takes in user commands and sanitizes it before sending the proper information to the server. It keeps track of an inSession flag for input checking and closes the output thread if quit is called. The output thread receives all server responses and prints the relevant information to standard out and updates inSession if needed. If the socket is shut down from the server end, it closes the input thread as well.

Our server program has a few different functions and structures. The account structure holds information for an account and the node struct is part of a link list. Each node points to an account. The main method sets up the server to listen and accept connections and also sets up the signals that can be caught for both account printing and server termination.
When a client connects, the main method creates a new thread for that connection with the clientCommandWrapper function. This function continuously reads data from the socket and calls the following methods accordingly. The createAccount method creates an account by adding a new node and populating proper fields of the account with default values and the account name. The serve function processes all functions that can be completed during a service session. Only the serve command needs to check if the account of interest is already being serviced; the server thread stores the account name for future use and simply keeps operating on the same account until end is called. PrintBankAccnList prints the accounts when it is called by the timer every 15 seconds. Timer is the method that keeps the 15 second repeat going and quit is called to terminate the server, deallocate memory, close connections, terminate threads, and close gracefully.

Headerfile:

We did not find a header file to be necessary for this assignment and therefore included all of our methods and variables in our bankingClient.c and bankingServer.c files.

Testing Process:

We tested this program as we developed the code. Every time we wrote a method or code that was supposed to support a certain functionality, we would export that code to another document and test it individually to make sure it performed as expected. This allowed for us tackle small errors as we went along.

For the final testing process, we tried different cases and combinations of commands. We connected multiple clients and observed the behavior of the program. We made sure to check that all of our clients' commands were taken in properly and that all of the threads were being launched and run properly. We also confirmed our data's integrity using the proper locking mechanisms.

Instructions for Running:

Please compile using the make file and run using the format specified in the assignment document. If the proper arguments are not inputted, instructional error messages will be displayed.