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 problem related to having unique ID's for each account currently in service. It was difficult to figure out how to implement this as a request for service would need to check both that an account is not in service, and if it is in service, the request should recognize if it is the same instance serving the account and proceed if so. We found that the only way to implement this functionality was with unique ID's. An issue that came up with that 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 the value and it would simply pass on an increment every time a service ID was requested.

An important thing to note about this project was that a lot of the code was standard and not too versatile. Setting up a client server connection in C is very straightforward and is mostly defined by a ordered set of steps. Using a timer functionality for something like the 15 second signal interrupt was also standard code with pre-defined structures and steps. We simply researched the code for these types of functions and adapted it to suit our program.

**Implementation Basic Overview:**

Our client program has 3 functions. The main method contains the connection code 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 sends the respective information to the server. It formats the result and instructs the server to call whatever data manipulation functions necessary. The output thread handles any and all possible server responses and prints the relevant information to standard out.

Our server program has a few different functions and structures. 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. 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 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. An option, service ID, and any other necessary data is taken in and processed through conditional logic. Printbankaccnlist prints the accounts when it is called every 15 seconds and clientcommandwrapper routes client input to calling the proper functions and arguments. Timer is the method that keeps the 15 second repeat going and quit is called to terminate the server 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 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 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.