Andrew Lu
Joshua Olazo

bankingServer.c
The basic structure of the server is that it takes input and creates a global LL of accounts and modifies balance and the service flag. A linked list of all sockfd and all thread ids are made for closing the program.

nameAlreadyExists(char * input) - takes input char* goes through the global LL looking for the same name. Returns true if found, false if not found.

createAccount(char * input) - takes input char* creates a new account gives it the name of input, sets balance to 0, flag to false and then if the global list is empty this new account becomes the global node otherwise attaches it to the end of the global LL.

trimcommand(char * input, int a) - trims off first a letters of input, removes leading and trailing whitespace, return the resulting string.

isNumeric(char* data) - Checks if a string is indeed a number, this is so we can convert it to a number later.

metadata() - prints out properly formatted list of all the accounts along with balance as well as IN SERVICE if the account is in service.

Void * print(void * args)- Used to set 15 second alarm. Calls metadata(). A semaphore is used so that nothing can access the database until the metadata printing is done.

Void banking(void * args)- Selects proper account to work with this round as example based first on flag and then if not the last node in the global LL. Calls of: create,serve,deposit, withdraw, query,end,quit here. Intercepts first n letters from buff and uses strcmp on these letters to determine which command to execute if at all. Uses trimcommand to clean buff then proceeds to execute whatever function which has already been documented in the Assignment 3 pdf. Only input argument is sockfd.

Main function- Inputs handled here. Code to create socket, bind socket, and connect server. The main banking method is called here.

bankingClient.c
No synchronization required
The Client creates a socket and connects to the server. Two threads are made to talk with the server, one for reading and one for writing. UNDER THE ASSUMPTION THAT CLIENT WILL NOT EXIT WITH CONTROL C ONLY QUIT

void writr(void * args) - reads commands from uses and writes it to socket using write() the thread then sleeps for 2 seconds. Keeps asking for new command if newline is entered. Input args are the tids of both threads and the sockfd

void readr(void * args) - reads messages from server using read() and prints.
Input args are the tids of both threads and the sockfd If message is a follow up from client quit or server control c, client gracefully closes program If message read is 0 bytes, server has been killed through other means and client gracefully exits.

banking.h
Sets max buffer size to 300 and SA as a short for struct sockaddr.
Global port is defined
Structs:

1. Bool - true and false
2. Account- holds name, balance, inservice flag, and next account used for server.c
3. Client- holds tid, socket, and next Client used for server.c
4. thread_args, holds socket, read and write tids used for client.c

Commands:
Before every input the Account of interest is read from global

1. Create- There is a lock before every account create to database
2. Serve- There is a lock before every account create to database
3. Deposit - Since only one client can serve an account a lock is not necessary
4. Query - Since only one client can serve an account a lock is not necessary
5. Exit - stops service session.
6. Quit - exits the account if in session and sends a message to client that will be read and make the client exit

Metadata print - see functions print() and metadata().

Control C (SIGINT): When server recieves sigint, it will take the linked list of all sockfds and thread ids and close/exit all of them. It then closes the sockfd for listening for new clients. UNDER THE ASSUMPTION THAT CLIENT WILL NOT EXIT WITH CONTROL C