Black Jack Project Report

By. Daniel Yakovlev and Marci Ma

## Compiling and Running Instructions:

To compile the Server:

g++ -pthread Server2.cpp dealer.cpp Deck.cpp -o server

Run the Server:

./server 4198

To compile the Client:

g++ Client2.cpp Player.cpp -o client

Run the Client:

./client [server ip]

## Game Instructions:

While connected to the server, the application will ask for a username to store the points on the scoreboard, once the username input the username, the game will start. The deck will be shuffled and display the deck at the server side, so the client cannot see what is coming on the next card. Dealer and user will both have two cards in hand, the user is able to see the dealer's first card, the user can make two choices while gaming, to HIT or to STAND. To hit and get the next card, type ‘H’, to stand and stop getting the card, type ‘S’. Dealers must get the card if their sum is less than 17, the system will calculate the sum of both sides if they both stop getting the card.

If the dealer sum is less than 21 and greater than the user sum, or the user busted, the dealer won. If the dealer and user busted or their sum of the card are equal, then in this situation it is a push. Otherwise, the user won. The game will display the result and ask the user if they want to play again, if yes, type ‘Y’, or type ‘N’ to exit the game. If the user chooses to exit, then the game is ended, and then the server and client will also close.

## Implementations:

**Register**.

The Client will be given the prompt to enter a username, once they enter the username they want to use. The client will send that information to the server where it will output the user that joined.

**Create Game**.

A game can be created when the server is run. Users will be able to join the server once it is available.

**Join Game**.

Users are able to join games when they are given the ability to start. They will type in the run command along with the ip address of the server.

**Exit Game**.

Both the server and client have the option of exiting the game gracefully. The user can exit a game after they won or lost the black jack game. They will be given the option to enter Y or N. Y to continue the game or N to exit. Once they type N, the program will close.

The server has the option to exit the game whenever possible by typing E. This will be input that a server input thread will take in. From there the server will close gracefully and take care of any memory. However, it is recommended to close the server when there are no games running as it will interrupt the existing games.

**Unregister**.

The user unregisters by not continuing to play the game. At that point the game will close the file descriptor for that user.

**Application Specific protocol**.

The application has a specific protocol that it uses to send messages. The client will send messages with characters that will let the server know what it wants to do. If the server receives H it will know to send a card. If it receives a S it will know to stand. The server protocol for sending information is to send the card information. This will be the first letter in the buffer that is sent.

**Game List.**

The game list has not been implemented as the client directly connects to the server.

# Daniel Self Grading

I believe that I did an equal amount of work. I worked on the main game logic of the game in the client side. I spent most of the time trying to get the game to work and not hang up on different occasions. I also implemented the protocol on how the client and server would communicate. I would grade myself a 9/10. I think that I put a good amount of work in for the project along with my partner.