Computer Networks and Internet 1 - 83455
Wet Assignment (Network Programming)

Submission Guidelines:

- Please zip and submit your files through Moodle.
- Submission is in singles or in pairs
- Late submissions will not be accepted unless approved **in advance** by one of the TAs.
- Questions regarding the exercise can be sent to one of the TAs.
  Please write the course number in the mail subject

An online "Casino War" card game (a variation of the popular "war" game)

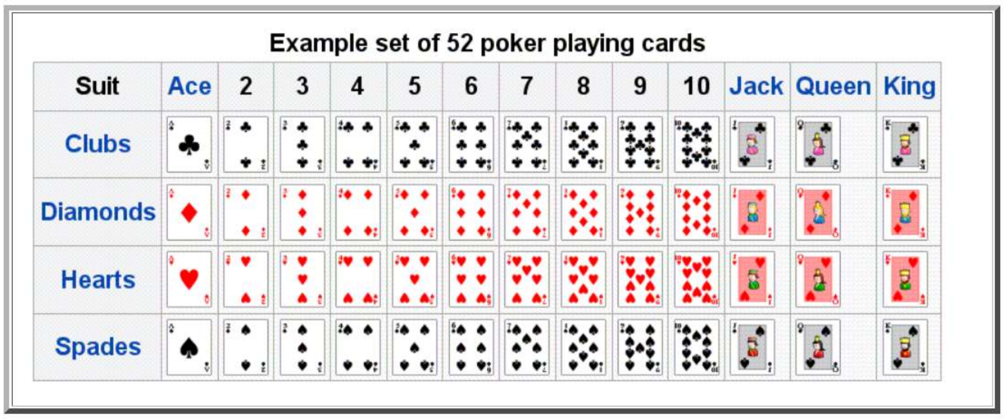
1. Introduction

Your task is to implement an online "Casino War" card game between a client (a "human" player) and a server (remote "gaming machine" – a "dealer").
The rules of the game:
The game is normally played with six standard 52 card decks, **BUT WE WILL USE ONLY 1 DECK**. The cards are ranked in the same way that cards in poker games are ranked, with aces being the highest cards.
One card each is dealt to a dealer (the server) and to a player (the client).

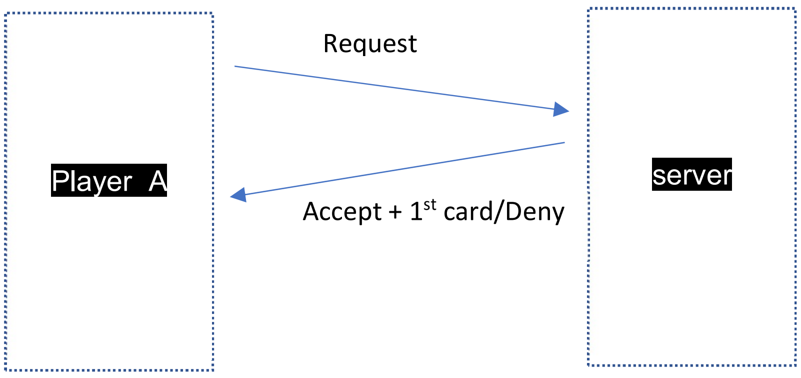
- If the player's card is higher, he or she wins the wager they bet. However, if the dealer's card is higher, the player loses their bet.
- A tie occurs when the dealer and the player each have cards of the same rank. In a tie situation, the player has two options:
  i. The player can surrender, in which case the player loses half of the bet.
  ii. The player can go to war, in which case the player must double his stake. If the player continues play in view of a tie, the dealer burns (discards) three cards before dealing each of them an additional card. If the player's card is ranked higher than the dealer's card, then the player wins the amount of his original bet. If the dealer's card is ranked higher than the player's card, the player loses his (doubled) bet. If the ranks are equal, then the player wins the amount of his doubled bet.

2. Technical details

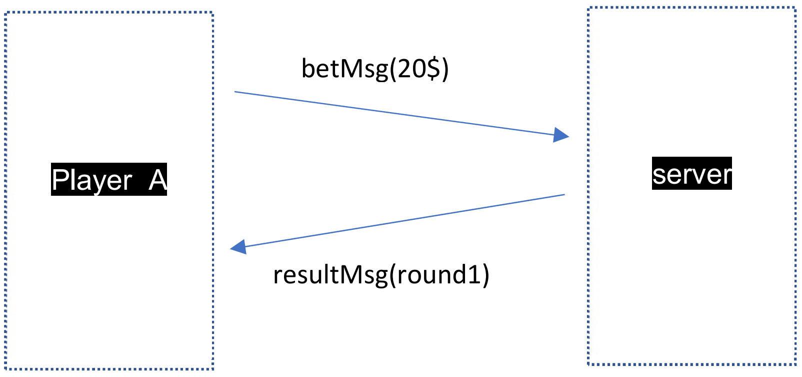
a. The server should support two games in parallel (two players can play at the same time).
   We recommend that you design the code so that the server holds the logic and game history, while the player is only responsible for sending and receiving messages
b. Naming the cards in the correspondence between the server and the client:
   Each card shall be represented by a string of 2 or 3 characters. 1st character is the rank and the 2nd is the suit, e.g. “Ad” means ace of diamonds, “10c” means 10 of clubs, etc.


Example set of 52 poker playing cards
Suit
Ace
2
3
4
5
6
7
8
9
10
Jack
Queen
King
Clubs
Diamonds
Hearts
Spades

c. Starting a game session:
   The communication sequence starts with the player A (client) sending a request to the gaming machine (server) wishing the start a game. The gaming machine replies according to its status, meaning if it is occupied already with two games , it replies that it is occupied and denies the request by sending a “deny message”, otherwise it will receive the invitation, and send back an “accept message”. **The “accept message” should already include a card to be sent to the player.**
   Each message should be displayed on his stdout by the recipient.
   Here is a graphic representation of the request sequence:



d. Once the game starts and the player receives his 1st card, it is time to bet. At the start of each round, the player sends a bet message to the server stating the sum of the bet and nothing more (the server should know what card he gave to the player so no need to resend it. Once the message is received, the server sends a reply message with the dealer's card, the result of the current round, and the number of the current round. For example, for the 1st round the dealer sends a message that should display on the stdout of the player as follows:
The result of round 1:
Dealer won: 20$
Dealer's card: AS
Player's card: 10D



In Case of a tie, the server sends a message declaring a tie and asks the player to make a move according to the rules of the game. The following is an example of a tie declaration message as displayed by the player:

The result of round 1 is a tie!
Dealer's card: AS
Player's card: AD
The bet: 100$
Do you wish to surrender or go to war?

The player's reply will be as follows:
If the player chooses to surrender, he sends a message to the server and tells him he wishes to surrender.
In that case, the server will send back a message that will be displayed as follows:
Round 1 tie breaker:
Player surrendered!
The bet: 100$
Dealer won: 50$
Player won: 50$

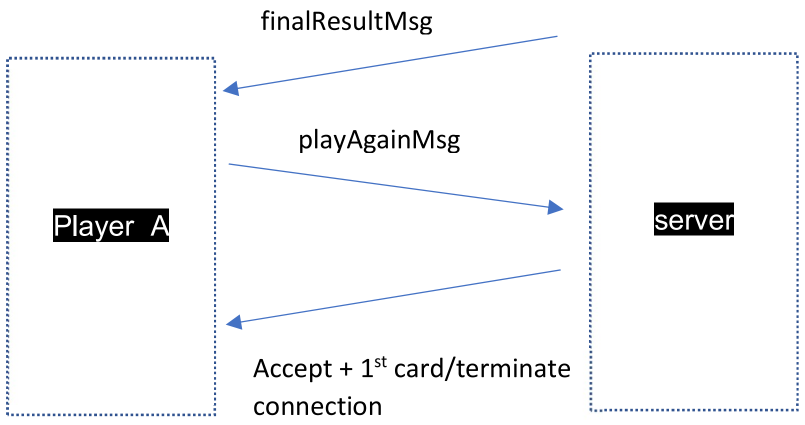
In case the player goes to war, he sends a “war declaration” to the dealer. The dealer then goes along the game description and replies with a message according to the scenario:
Example 1 – the player received a higher card
Round 1 tie breaker:
Going to war!
3 cards were discarded.
Original bet: 100$
New bet: 200$
Dealer’s card: 5C
Player’s card: QD
Player won: 100$

Example 2 – the dealer received a higher card
Round 1 tie breaker:
Going to war!
3 cards were discarded.
Original bet: 100$
New bet: 200$
Dealer’s card: KC
Player’s card: QD
Dealer won: 200$

Example 3 – another tie.
Round 1 tie breaker:
Going to war!
3 cards were discarded.
Original bet: 100$
New bet: 200$
Dealer’s card: KC
Player’s card: KD
player won: 200$

e. The game ends once there are no cards left in the deck. In this case the dealer (server) sends a message declaring that the game has ended and naming the winner, and the net amount of money the player won/lost. Also it asks the player if he wishes to play again. For example:
The game has ended!
Player won: 1200$
Player is the winner!
Would you like to play again?

If the player decides not to play, he sends a “no” and the connection is terminated by the server. If he sends a “yes” than a new game is launched.



f. A special feature of the server is to allow the player to view his status in the middle of the game and to be able to terminate the game at any time.
If the player sends the status message, the server replies with a message that looks the following (on the player’s stdout):
Current round: 11
Player won: 15$

If the player quits midgame, the server replies and the player views a message similar (but not identical) to the end of game message written before:
The game has ended on round 13!
The player quit.
Player lost: 1200$
Thanks for playing.

3. Implementation

You may implement under either Windows or Unix/Linux with an editor and language of your choice.
For C language:
You **may not** use any pre-made libraries **except for** stdio.h, stdlib.h, winsock*, socket.h, arpa/inet.h, sys/types.h, string.h, conio.h, curses.h, unistd.h, errno.h, fcntl.h and time.h (make sure you **need** the ones you include). You need to implement everything else; if you have doubt, please email a TA.

For any other language:
You may use standard libraries, NOTICE: if you have doubt, please email a TA.
Important: you may choose to make certain relaxations of the design (e.g., include the IP addresses in the input strings), yet notice these **may result in a point reduction**, depending on their being substantial or minor. In case you knowingly do choose to make these, you must provide their exact description in the external documentation file; otherwise, these will be regarded as a mistake and result in a greater point reduction. If you have questions on this issue, please contact a TA.

4. Documentation

Your code must be documented, **both internally** (using comments to briefly describe functions wrote and non-trivial logic) **and externally** (a text document providing description of the functions and the program flow containing screen-shots of running game).

5. Submission

You are to submit two code files (dealer_*os, player_*os) - os represents the operating system you use, two appropriate executable files (.exe, or another executable files depends on the language you used) and the document containing your names and ID numbers, external documentation and any remarks you find necessary.
The code files must begin with the names and ID numbers of the submitters as a comment.
Please zip the files and submit to Moodle (only one of the member of each pair has to submit).

Please try to enjoy your time, by doing this practice you will establish your network programming skills and improve your coding.
😊