Exercise one, inf-3701 2013

Old Maid

You will implement a client for the game “Old maid”. It will be played with 53 cards, a full deck + a joker. It will be played by 2 to 8 players.

The rules:

The players will discard equal pairs of cards (spades and clubs of same value or diamonds and harts of same value). The player left with the joker (the “old maid”), looses.

When the game starts, each player will pull a single card from the deck. When all cards have been issued, pairs can be discarded and the first player will offer his hand to the next player, who will pick a card. The player can now discard a pair of cards if they match the criteria, and then offers his hand to the next, and so on.

This will be provided:

Table:

A table service, with the following API. The return values and parameters are coded as JSON. It will be running on the machine “njaal.net” on port 9898.

Join game:
{“cmd”:”join”, “nick”:your_nick} → {“result”:”ok”, “players”:[“player1”:[(ip, port), nick], “player2”:[(ip, port), nick], …]}
The join game function blocks until all players of a game has joined. Only one client can join from one IP/Port pair, and the nick must be unique for the game. The table decides when the game starts (by returning a player list).

Draw a card:
{“cmd”:”draw”} → {“result”:”ok”/”last_card”/”error”, “card”: [“3”, “spades”]}
Returns a single card

Discard a pair of cards
{“cmd”:”discard”, “cards”: [[“3”, “spades”], [“3”, “clubs”]], “last_cards”:”true”/”false”} → Returns {“result”: “ok”/”error”, “message”:”ok”/error_message}

Out of cards
{“cmd”:”out_of_cards”} → {“result”:”ok”}
Send this when your client is out of cards.

Get status
{“cmd”:”status”} → {“in”:[“player1”, “player2”, …], “out”:{“player3”}} “out” players are ordered by how early they were out – winner first. This call is optional. “player1”... are the given nick names.

Client:

You will implement a client in any programming language you want.

The API of the client is as follows:
The client **must** accept a connection from it's right player. This player will offer the hand.
All clients **must** connect to their left player (the next player). Players should thus have 3 connections open, one to the table, one to their right player (the last player, initiated by the remote player) and one to their left player (initiated by the client itself).

Your turn
{“cmd”:”your_turn”} → Returns {“result”:”ok”}

Offer hand:
{“cmd”:”offer”, “num_cards”:number of cards left} → {“result”:”ok”/”out”}. If “ok”, the player is still in the game and will send a “pick card” request. If “out”, you must connect to the next player in the list until you are either out of other players (you lost) or you get an “ok”.

Pick card:
{“cmd”:”pick”, “card_num”: the number of the card chosen (must be between 0 and number_of_cards_left offered} → {“result”:”ok”/”error”, “card”:[“3”, “spades”]}

Your client must first join the game (use a reachable IP address) by connecting to the table and sending the Join request. When the join game returns “ok”, a connection must be made to the next player in the list. If you are “player2”, you must contact “player3”. If you are the last player, you must connect to the first player. Player1 starts drawing a card, then sends “your turn” to the next player. This is repeated until a player receives the “last_card” reply. Remember that you must either add each new card you receive at a random place in your hand, or shuffle the hand. This is always true when you receive a card. You should also always discard pairs as soon as possible. The last client to draw will start offering his hand to the next player.

The game ends when all pairs have been discarded. The client left with the Old Maid card lost.