Client: Create socket  
Client: Attempt connection to server  
Client: loop between Send and Recv functions with appropriate I/O based on questions the server is giving as it will always be going back and forth between sending and receiving (in my eyes it will never receive twice or send twice in a row)  
Client: Exit loop when connection is killed by server  
  
Server: Create socket  
Server: Wait for connection  
Server: Connection is made and fork is used to allow multiple processes **(would fork be before new-s [receptionist] initialization or after?)**  
Server: Infinite loop starts and sends message "Do you want to play (Y/N)" and waits for client to respond. Basic loop to keep message going until Y or N is inserted. If Y, continue into game, if N, send goodbye statement and close the connection as well as exits the child  
Server: At game start (after Y is inputted) , Call rand function to get a number 0-9 and put it into a char array 5 times (gets a 5 digit code). Also set a tries variable to a number and a boolean win variable to false  
Server: Asks Client to guess and displays number of tries left (loop repeats if tries>0 && !win)  
Server: Checks if Client response is 5 characters, if not display message   
Server: If Client Response is 5 characters, create a new Char buffer of size 5 and go through each character array (both the answer and the client response) with a simple nested for loop. If a match is found and index's match, put a "C" in the corresponding index of the new buffer then break the for loop. If a match is found and index's don't match, put a "O" in the corresponding index and break the for loop IF the answer index > client response index) if there is not a match found, put an "X" in the corresponding index.  
Server: If this index is all C, set win to true  
Server: If (wins), use a specific sprintf that takes the new char buffer and has a pre-written win message  (that has "press enter to continue in order to facilitate the client alternating send/recv pattern)  
Server: Else If (tries > 0), use a specific sprintf that takes the new char buffer and tries left  
Server: Else, use a specific sprintf that takes the new char buffer and has a pre-written lose message (that has "press enter to continue in order to facilitate the client alternating send/recv pattern)  
Server: Sends message to client