1.Briefly describe the design of the program. How does your program control when the client runs and when the server runs?

When the server runs, the server sets up a handshake and a listening socket and waits in the accept() call for a connection. When a client connects, the sever begins to read from the socket. It then writes the message it received back to the socket in all caps. When socket read is empty, the server exits.
When the client runs, the client sets up a session socket and attempts to connect to the server. If the server doesn't respond the client exits with an Error #2 for the socket.

2.What is the purpose of the handshake socket? Why not have the server create and bind session sockets that clients may connect to directly?

Handshake sockets are a single point of contact with a server by which clients can acquire session setup data from the servers handshake response for setting up the session sockets.

3.For the simple / client server program, we chose to use sockets instead of pipes to send messages between the client and server. Why are sockets preferred over pipes for this program? Give at least two reasons.

A single socket can perform bidirectional communications and sockets can be used to communicate with many processes like in the traditional many client one server scenario.