Question 3: Message Queues (9 pts)
Review the programs (spock.c and kirk.c).
Answer (or discuss) questions listed below:

a) Discuss and evaluate what happens when you're running both in separate windows and you kill one or the other. (3)
When you close spock while kirk is still running, the messages will no longer be printed to spock even though you are still able to write messages into spock. This is because kirk sends the msgctl response to close out with the ipc_rmid argument but not spock sending anything to tell kirk to close. When you close kirk first while spock is running kirk sends the msgctl response to spocks instance which triggers the msgrcv call to close and use perror("msgrcv"). Hence both close in this case.

b) Discuss what happens (and why) when you run two copies of kirk. (3)
Both kirks attach to the same message queue since ftok is using the file to determine a key, which inturn gives them both the same key to the message queue. Likewise since spock uses the same key as kirk any message sent to the message queue through the two kirks will be in the same space so it gets printed no matter what. When you close either kirk tho its clear that the message queue will still send the msgrcv terminate which causes spock to close.

c) Discuss what happens (and why) when your run two copies of spock.(3)

When any message is written to kirk the two spocks take turns printing out the message in a round robin like fashion. When a message is written to kirk, the message queue struct is incremented by 1, signaling to the other two spocks that a message has been written to the buffer. Then the spock that was ran first has priority so it will print it out and decrement by 1. On the next message from kirk the other spock will run and print the message. Only 1 can print at a time because once 1 prints, the message will be removed from the buffer so the other instance of spock has no message to print.