3) a.

Killing a Spock while a Kirk is running only kills the Spock. The Kirk still functions normally. This is because Kirk utilizes a buffer system to store strings that can be sent once a Spock reconnects to the queue. Since the Kirk program creates and maintains the queue, simply disconnecting Spock has no bearing on the functionality of Kirk. Killing Kirk, however, is another episode. Doing this causes the message queue to terminate. The Spock program is terminated as well. This is because deleting the server causes the connection formed by the client Spock to separate.

b. Running 2 copies of Kirk connects both copies of Kirk to the message queue. They each have their own buffer system to store messages to send. Since both copies are connected, Strings can be sent to the same Spock by both Kirks get output to the console. Killing one (or both) of the copies of Kirk also disconnects the Spock program from the message queue, and it also causes the active copy of Kirk to stop functioning properly. It cannot send a message to the buffer since deleting a copy of Kirk using ^d also deletes the actual queue.

c. Running 2 copies of Spock connects both copies of Spock to the message queue. With a copy of Kirk running, sending a message through Kirk causes the 2 Spocks to receive the message in an alternating pattern. This is most likely due to the message queue using an iterator to pass over all connected Spocks and sending each a message. Terminating one of the Spocks follows the pattern of the 1Spock to 1 Kirk case, with none of the other connected components within the queue. Kirk can still send messages and the other copy of Spock can still receive them.