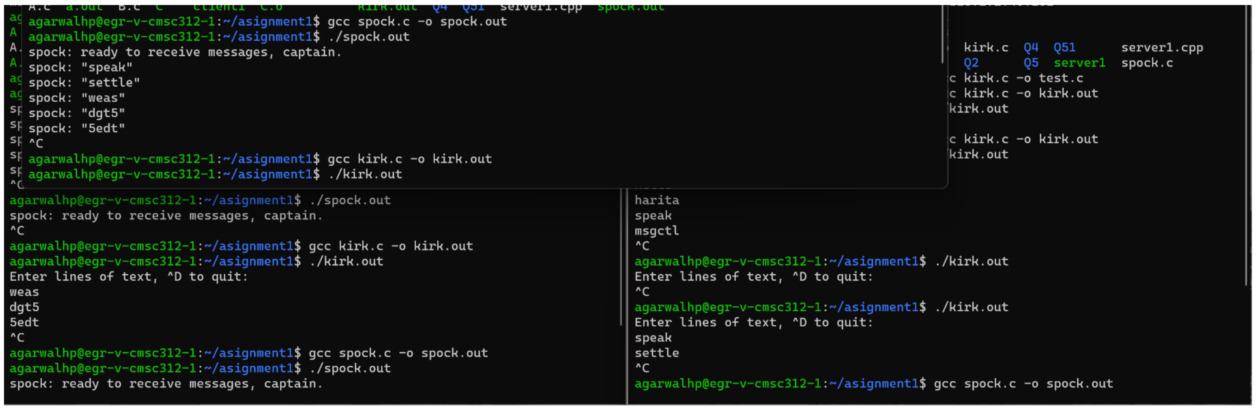
a) Discuss and evaluate what happens when you're running both in separate windows and you kill one or the other.
When both programs are executed in different windows, the following happens:
If you stop the "kirk" program, the message queue will stay intact, and the "spock" program will continue to wait for messages since it is responsible for deleting the message queue through the msgctl function call; the message queue will be cleaned only after the "kirk" program dies. The "spock" program will no longer be waiting for messages if you exit it, but the message queue will remain intact. Messages will continue to be delivered to the queue by the "kirk" software, but no program will be accessible to read them.

b) **Discuss what happens (and why) when you run two copies of kirk.**
When two copies of the "kirk" software run concurrently, they connect to the same message queue and can send messages to it. Nevertheless, the messages will be read by the "spock" program in the order in which they were received, with no way of knowing which "kirk" program supplied which message. This is because the "mtype" field of the message buffer is not used to identify the sender. The "mtype" parameter is always set to 1, indicating that it is not used to distinguish message kinds in this circumstance.



```
agarwalhp@egr-v-cmsc312-1:~/asignment1$ gcc spock.c -o spock.out
agarwalhp@egr-v-cmsc312-1:~/asignment1$ ./spock.out
spock: ready to receive messages, captain.
spock: "speak"
spock: "settle"
spock: "weas"
spock: "dgt5"
spock: "5edt"
^C
agarwalhp@egr-v-cmsc312-1:~/asignment1$ gcc kirk.c -o kirk.out
agarwalhp@egr-v-cmsc312-1:~/asignment1$ ./kirk.out
```

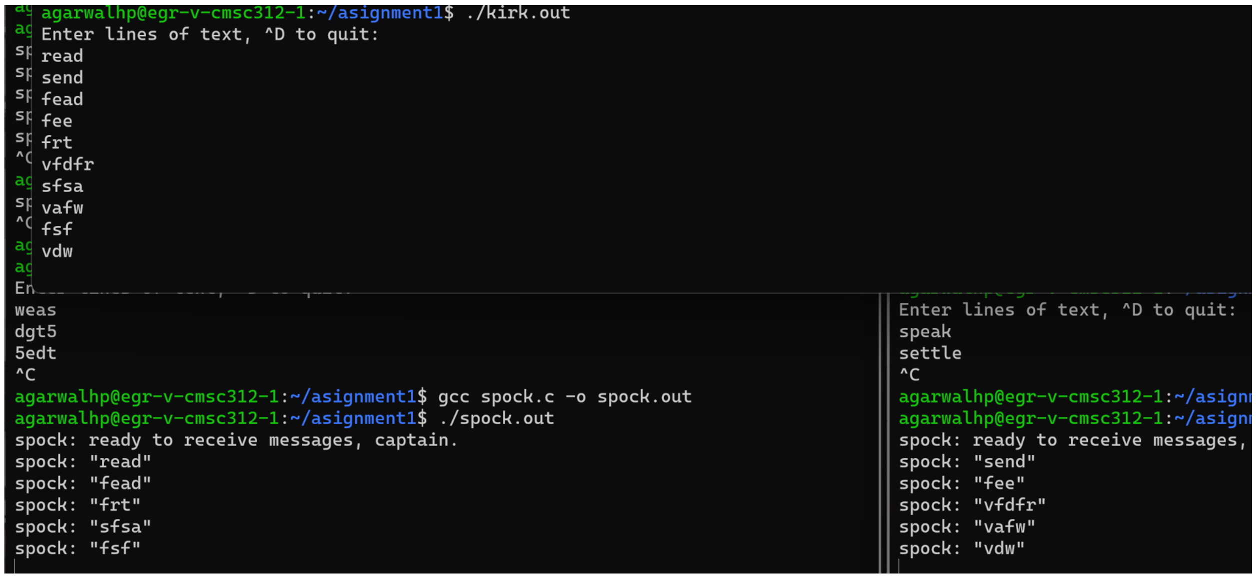
```
agarwalhp@egr-v-cmsc312-1:~/asignment1$ ./spock.out
spock: ready to receive messages, captain.
^C
agarwalhp@egr-v-cmsc312-1:~/asignment1$ gcc kirk.c -o kirk.out
agarwalhp@egr-v-cmsc312-1:~/asignment1$ ./kirk.out
Enter lines of text, ^D to quit:
weas
dgt5
5edt
^C
agarwalhp@egr-v-cmsc312-1:~/asignment1$ gcc spock.c -o spock.out
agarwalhp@egr-v-cmsc312-1:~/asignment1$ ./spock.out
spock: ready to receive messages, captain.
```

```
 kirk.c  Q4  Q51      server1.cpp
 Q2      Q5  server1  spock.c
c kirk.c -o test.c
c kirk.c -o kirk.out
kirk.out

c kirk.c -o kirk.out
kirk.out

harita
speak
msgctl
^C
agarwalhp@egr-v-cmsc312-1:~/asignment1$ ./kirk.out
Enter lines of text, ^D to quit:
^C
agarwalhp@egr-v-cmsc312-1:~/asignment1$ ./kirk.out
Enter lines of text, ^D to quit:
speak
settle
^C
agarwalhp@egr-v-cmsc312-1:~/asignment1$ gcc spock.c -o spock.out
```

c) **Discuss what happens (and why) when your run two copies of spock.**
When two instances of the "spock" program are run, they will both attempt to read from the same message queue. Yet, only one of them will be permitted to read each communication. This is because messages are withdrawn from the queue as they are read, thus each message can only be taken once. The two "spock" programs will battle for the messages, but only one will be allowed to receive each one. The messages will still be read in the order in which they were received, so the program that reads each message will determine how it is handled.



```
agarwalhp@egr-v-cmsc312-1:~/asignment1$ ./kirk.out
Enter lines of text, ^D to quit:
read
send
fead
fee
frt
vfdfr
sfsa
vafw
fsf
vdw
```

```
weas
dgt5
5edt
^C
agarwalhp@egr-v-cmsc312-1:~/asignment1$ gcc spock.c -o spock.out
agarwalhp@egr-v-cmsc312-1:~/asignment1$ ./spock.out
spock: ready to receive messages, captain.
spock: "read"
spock: "fead"
spock: "frt"
spock: "sfsa"
spock: "fsf"
```

```
Enter lines of text, ^D to quit:
speak
settle
^C
agarwalhp@egr-v-cmsc312-1:~/asignm
agarwalhp@egr-v-cmsc312-1:~/asignm
spock: ready to receive messages,
spock: "send"
spock: "fee"
spock: "vfdfr"
spock: "vafw"
spock: "vdw"
```