Cassidy Brown – cmb195
OS – HW 6
10/26/16

Each process/thread counted up to 100 million.

Problem 1: Peterson's solution with multithreading

```
cassidy@nyota:~/eecs338/hw6$ gcc -o p1 hw6_p1.c -lpthread
cassidy@nyota:~/eecs338/hw6$ ./p1
Child thread count: -100000000
Parent thread count: 100000000
Global counter: -45
```

Not perfect, but magnitudes better than when the threads are left to their own devices. Another test resulted in a global counter = -2

Problem 2: Peterson's solution with fork and shared memory

```
cassidy@nyota:~/eecs338/hw6$ gcc -o p2 hw6_p2.c -lrt
cassidy@nyota:~/eecs338/hw6$ ./p2
Child process count: -100000000
Parent process count: 100000000
Global counter: 0
```

Beautiful. A later test resulted in global counter = -1, but that's still very good.
I had to allocate shared memory for the turn and flag variables as well as the global counter. Trying to make them global variables outside of the main function was ineffective.

Problem 3: Semaphores with multithreading

```
cassidy@nyota:~/eecs338/hw6$ gcc -o p3 hw6_p3.c -lpthread
cassidy@nyota:~/eecs338/hw6$ ./p3
Parent thread count: 100000000
Child thread count: -100000000
Global counter: 0
```

So the semaphores worked better than Peterson's solution, but they were significantly slower.

Problem 4: Semaphores with fork and shared memory

```
cassidy@nyota:~/eecs338/hw6$ gcc -o p4 hw6_p4.c -lpthread -lrt
cassidy@nyota:~/eecs338/hw6$ ./p4
Child process count: -100000000
Parent process count: 100000000
Global counter: 0
```

Again, semaphores are successful, but much slower.