Chapter 8, pages 12-18 and Chapter 9, pages 1-2

You will need to open 5 terminal windows to do this homework (to run 2 copies of heavyweight and 3 copies of lightweight). Remotely,this means 5 ssh sessions or using a terminal multiplexer (tmux or an alternative). I suggest setting up ssh keys if you haven't already to avoid typing your password 5 times (ssh-keygen).

Program 1:

A simple exercise in using counting semaphores (see chap 8 pages 14-16). You will build "LightWeight" and "HeavyWeight" programs. There will be a total of 5 resources. The LightWeight program will lock two of them, the HeavyWeight program will lock three of them.

Both programs will do the following (use a for loop) 5 times:

1) Lock the required number of resources.
2) Print a message (either "HeavyWeight Starting" or "LightWeight Starting")
3) Sleep 4 seconds.
4) Print a message (either "HeavyWeight Ending" or "LightWeight Ending")
5) Unlock the resources
6) Sleep 8 seconds.

Testing: Start 2 copies of the HeavyWeight program, then 3 copies of the LightWeight program. Check your output to make sure you never have 2 HeavyWeights running at the same time, or any other combination that would require more than 5 resources. This means you can only have 2 lightweights starting and running, OR 1 lightweight and 1 heavyweight. If you have more in "Starting" state than the allowed 5, you are doing something wrong.

Hint: You may need to cleanup leftover semaphors. See chap 8 page 13 for removing them in your program, and chap 8 page 18 for removing them from the command line.

Demo: Your 5 light and heavy program runs and show the instructor the code.

Program 2:

An exercise using multiple semaphores. See chap 9 pages 1-2.

You will build a solution to the readers and writers problem. You will build both a Readers and a Writers program.

The writer program will do the following 5 times:

1) perform all necessary locks
2) print "Writing"
3) Sleep 4 seconds
4) print "Done writing"
5) perform all necessary unlocks
6) Sleep 8 seconds

The reader program will do the following times:

1) perform all necessary locks and associated counting
2) print "Reading"
3) Sleep 2 seconds
4) print "Done reading"
5) perform all necessary unlocks and associated counting
6) Sleep 4 seconds

Use a counting semaphore (as shown in lecture) to keep track of the number of readers. Use an array of 4 semaphores; three for the semaphores, one to count the number of readers.

Testing: Start 2 copies of the writer program, then 3 copies of the reader program. Check your output to make sure that when a writer is active, no one else is.

The correct P and V code for readers and writers is given on the back of this page.

Demo: Your 5 reader and writer program runs and show the instructor the code.

```
      Writer
P(BlockReaders)
P(WriterLock)
write
V(WriterLock)
V(BlockReaders)
```

```
      Reader
P(BlockReaders)
P(CounterLock)
  if (Counter == 0) P(WriterLock)
  Counter++
V(CounterLock)
V(BlockReaders)
read
P(CounterLock)
  Counter--
  if (Counter == 0) V(WriterLock)
V(CounterLock)
```

Multiple program discussion:

In both cases the solution requires you to run more than one program. This is the first time you have done that. You will discover that, if you build both programs in the same directory, you can't call both executables `a.out` (because you can't have two files in a directory that have the same name. An easy solution is to give the executables different name using a compiler option.

For example: `gcc writer.c -o w.out`
will cause the executable that the compiler outputs to be called `w.out`. (`a.out` is a default name when you don't tell it what to call it)

WARNING: `gcc writer.c -o writer.c`
will DESTROY your source code by dropping the executable on top of it; so be careful. I usually end my executable names in `.out` so this doesn't happen.