CPS 536 Homework 4
Due in class, at class time, Tuesday, 4 April, 2017

You are to write a program that involves multiple processes, shared memory, and semaphores for ensuring proper implementation of critical sections. The task of the processes is to compute the entries *f[2]* through *f[TABLE_SIZE]* where *TABLE_SIZE* is 999999; ignore *f[1]*. Each *f[x]* is *the number of distinct prime numbers in the range [2, 1000] that are factors of x.* For example, f[2] = 1 and f[150] = 3 as the distinct prime factors of 150 are 2, 3, and 5. Similarly f[510509] = 1 as the prime factors of 510509 are 61 and 8369 (and 8369 is not in the specified range). Your program is also to output a list of primes in a specified range. While the task itself is very simple, the idea is to gain understanding of the concepts of sharing, synchronization (to some extent), and controlling critical section using semaphore(s) (in UNIX).
(As an aside, for an x in [2, 999999], how much different can *f[x]* be from the total number of distinct prime factors of x?)

Invocation
Your program will be invoked from the command line as follows:
```
$ program-name  n-procs  low  high
```
where n-procs is the number of *worker processes* (see below) involved in the computation and it is an integer in the range [1, 10]. Each of low and high is an integer in the range [1001, 999999]. Their purpose is explained later and you may assume low ≤ high holds.

Description of the problem
The *main process* (the one that runs your program) is to set up appropriate shared memory area for *f[]* and anything else needed, perform any needed initializations, get semaphore(s), and initialize them properly. It should then fork n-procs children (worker processes) and wait for all of them to finish. Each child does the following **repeatedly** until *all f[]* entries have been computed (how can a process detect this condition?): logically speaking, "take responsibility" for *one entry* of *f[]* that has not been computed yet, and then compute and store that *f[]* value. I will leave it to you to decide which operations have to be protected and how they should be implemented. When all the *children* are done, the *main process* is to output the information (to standard output) as described below.

Required output
Consult the sample output. Your program should output the specific section of *f[]* as shown in sample output. In addition, your program should list (**by consulting the *f[]* values your program computed**) the list of primes found in the range [low, high].

NOTE

- Your program should make the critical section as small as possible (thus making the parallelism as large as possible). Therefore, you should only place those statements in the critical section that really belong there.

- Mark the beginning and end of your critical section(s) with /** begin csection **/ and /** end csection **/ respectively.
- Turn in a brief schematic of your high level algorithm for the worker process clearly indicating the various semaphores used, their initial values, and the wait() and signal() operations used on them.
- Implement any required wait() and signal() on semaphore as separate functions (as opposed to the sample code I showed in class) with appropriate parameters.
- Make sure that your program releases all shared memory segments and semaphores properly. As discussed in class, the system does not remove them automatically when your program finishes. Remember to clean up after yourself.
- You have to think about an appropriate algorithm to use for the computation of an *f[]* value. If you use "brute force everywhere", your program is likely to run "forever". Feel free to precompute prime numbers in the range [2, 1000] (at the start of the program) and use them in your algorithms. Instead, you can find such a list of primes on the web and use that also.
- Note that the *f[]* values computed have enough information to figure out if an integer `x` larger than 1000 is prime. Given that `x` is less than a million, if `x` is not prime, surely some prime number divides `x`. How large a prime number should we consider for being a divisor of `x` before we can be sure `x` is a prime?

Sample output

```
$ h4 12 300 500
n-procs must be in [1, 10].
$ h4 5 300 500
low must be in [1001, 999999].
$ h4 5 300000 300100
-----------------------------
f[710006] =   4
f[710007] =   2
f[710008] =   2
f[710009] =   0
f[710010] =   5
-----------------------------

Primes in the range [300000, 300100]:
300007 300017 300023 300043 300073 300089
```

What to leave on the system ?

Create a directory with the name `hwk4` in your home directory. Give the group `r-x` permission for directory `hwk4` and for all the files there. Leave your files there.