BBB priority scheduling exercises

Consult the Linux on-line manual
http://man7.org/linux/man-pages/dir_section_3.html
during this exercise for further explanations of the pthread library functions to be used.

a)

Make a program with two threads. One which output prime numbers using this silly code:

```
int i, n = 2;
       while (n < 1000000000)
       {
              for (i = 2; i <= n; i++) if (n % i == 0) break;
              if (i == n) cout << " " << n << flush;
              n++;
       }
```

And one with higher priority: It's a "clock-thread". Its job is to output the elapsed number of seconds followed by a new line (<< endl <<flush;). Make a loop, where a "seconds" counter is increased and written to the screen and then puts the thread to sleep for a second.(Remember to include unistd.h)

Both threads are to be made in the real-time class. First declare thread attr objects of the type pthread_attr_t . Then initialize them using the function `pthread_attr_init.` Then use the function `pthread_attr_setinheritsched` to tell that the attributes shall not be inherited from the creator thread by feeding it the value `PTHREAD_EXPLICIT_SCHED` . Next step is to use the function `pthread_attr_setschedpolicy` to choose SCHED_FIFO as scheduling policy. Then declare struct variables of the struct type `sched_param` and set the field `sched_priority` to a value between 0 and 99 (highest priority). Use these struct variables as a parameter in the calls of the function `pthread_attr_setschedparam` in order to set the priorities. Finally use these attribute variables when you create the threads.

b)

Expand your clock-thread to output minutes as well as seconds (use a double-loop).

c)

Expand your clock-thread to output tenth of seconds as well as minutes and seconds (use a triple-loop).

d)

If you have time: Use the function `pthread_getschedparam` in the clock-thread to check its scheduling policy and priority (output them to the screen).

e)

Now raise the priority of the primes thread above the level of the clock-thread. What happens?

f)

Will the scheduling policy `SCHED_RR (the round robin version)` instead of SCHED_FIFO do any difference?

g)

Will it help to introduce a little sleep in the primes thread?

h)

Now give the two threads the same priority. Will the scheduling policy `SCHED_RR (the round robin version)` instead of SCHED_FIFO do any difference