Problem 1. [50 pts]
In view of the economic crisis, some hypothetical building designers decide to create one restroom per floor and have it shared among women and men, but under the following policy: a sign on the door indicates its current state — (a) empty, (b) in use by at least one woman, or (c) in use by at least one man. If it is empty, either a man or a woman may enter; if it is occupied, a person of the same sex may enter, but a person of the opposite sex must wait until it is empty.

You are asked to emulate this behavior using either POSIX or Java threads by creating FCOUNT of women (female) threads, MCOUNT of men (male) threads for a restroom of RR_CAP capacity. There will be four functions in this approach: woman_enter, man_enter, woman_leave, and man_leave, to be used by the following pseudocode:

```c
const int FCOUNT
const int MCOUNT
const int RR_CAP

/* woman thread */
while (TRUE) {
    woman_enter();
    use_restroom();
    woman_leave();
    do_other_stuff();
}

/* man thread */
while (TRUE) {
    man_enter();
    use_restroom();
    man_leave();
    do_other_stuff();
}
```

where use_restroom() and do_other_stuff() call nanosleep() (this is a thread-safe sleep function) for a small yet random number of seconds. See “man nanosleep” for use of it.

Your *_enter() and *_leave() functions should use your design choice of semaphores or mutexes and feature a verbose mode, where they print out messages of the type “Man id entering, there are number of men now in the restroom” and “Woman id exiting, there are number women left in the restroom”.

Deliverables for this problem: a narration of your design choices and how you organize your logic, followed by your implementation (source code) along with sample runs. Point break down: pseudocode/basic logic [15], implementation [20], quality of code [5] quality of narration [10].
Problem 2 [25 pts]
There is this idea of refitting this usage policy to buildings with two restrooms, that is to make both mens and womens restroom into common-use restrooms under Problem 1's usage policy. Is it a bad idea or a good idea? Will the average waiting time increase or decrease? Will the variance of the waiting time increase or decrease? Will the chance of starvation increase or decrease? Perform two experiments, slightly modifying Problem 1's code, once using two restrooms with the new policy, and once using it to emulate a conventional way of using restrooms (mens only and womens only). Record the waiting times of the threads and plot a chart of their distribution. Do you notice any differences? Is the average waiting time or its variance changed? Are these changes as expected?

Problem 3 [25 pts]
Argue whether Petterson's solution will break if the entry section is not executed atomically. If you claim that it breaks, show an example sequence of instruction execution that breaks it. Otherwise, provide a sketch proof of correctness.

Problem 4 [20 pts]
Design a solution on how to implement wait() and signal() using the TestAndSet instruction with minimal busy waiting.