

EECS 338 Assignment #4: Concurrent Programming Algorithms #2
Due: Tuesday, February 27th, 2007

Turn in your typed hard-copy in the class, and deposit your solution doc (word, pdf, text) to Digital Dropbox.

(10 pts) **(1)** Consider the conditional critical region statement “**region v when B do S**” and its implementation with semaphores below.

```
wait(x-mutex);
if not B then begin
    x-count:=x-count + 1;
    signal(x-mutex);
    wait(x-wait);
    while not B do begin
        x-temp:=x-temp + 1;
        if x-temp < x-count then signal(x-wait) else signal(x-mutex);
        wait(x-wait);
    endwhile;
    x-count:=x-count - 1;
endif;
S;
if x-count > 0 then begin x-temp:=0; signal(x-wait) end else signal(x-mutex);
```

Initially, x-mutex semaphore is 1; x-wait semaphore is 0, and integers x-count and x-temp are zeroes. Give a semaphore-based implementation of the other conditional critical region construct:

```
region v do begin S1; await(B); S2; end;
```

(10 pts) **(2)** Consider the following monitor implementation with a single condition variable x:

Initialization:

```
semaphore mutex := 1; semaphore next := 0; int next-count := 0;
semaphore x-sem := 0; int x-count := 0;
```

For mutual exclusion, each procedure P is implemented as:

```
wait (mutex);
    Body of P;
if next-count > 0 then signal(next) else signal (mutex);
```

Each x.wait in a monitor procedure is implemented as:

```
x-count := x-count +1;
if next-count > 0 then signal (next) else signal (mutex);
wait (x-sem);
x-count := x-count - 1;
```

Each x.signal in a monitor procedure is implemented as:

```
if x-count > 0 then begin
    next-count :=next-count + 1;
    signal (x-sem);
    wait (next);
    next-count := next-count -1;
end;
```

- (a) Assume that process P_i is waiting on x inside a monitor procedure, and process P_j issues $x.signal$. Then what happens: Does P_i wait until P_j leaves the monitor, or vice versa? Explain your answer.
- (b) Assume that our monitor has two condition variables, namely x and y . Revise the above monitor implementation with semaphores (i.e., give implementations for the mutual exclusion, $x.wait$, $x.signal$, $y.wait$, and $y.signal$).

(80 pts) **(3) Railroad Crossing Problem.** Consider a single-track railroad with multiple trains using the track (always from east to west) and a one-way (always from north to south) car crossing where cars can cross the railroad in both directions at the same time.

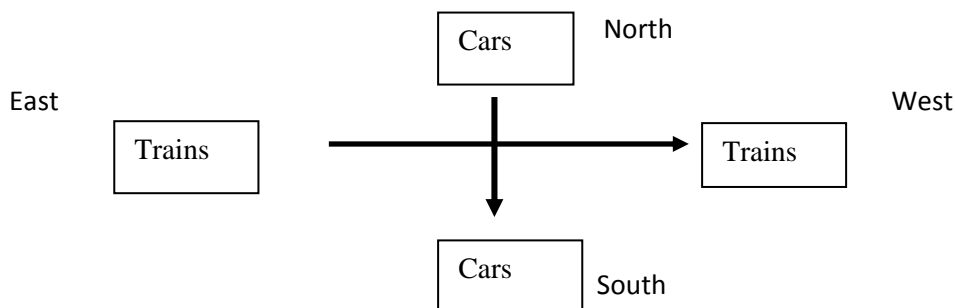
The crossing is controlled by a “crossing light”. The light always rotates from green to yellow to red, and then to green. When the light is green, at most four cars (from north to south) can be crossing the railroad at the same time.

When the light is yellow, no new cars are allowed to cross the railroad, and the cars that are in the process of crossing the railroad track complete their crossing. When the light is red, no cars are allowed to cross the railroad.

We assume that there is enough time between the event that the light changes from green to yellow and the event that the light changes from yellow to red so that all cars that are in the process of crossing the railroad complete the crossing process safely and, when the light turns red, there are no more cars that are crossing the railroad.

Multiple trains can use the railroad at the same time, and arrive at the crossing one after another—but, always in the same direction (east-to-west); we assume that the sequence of trains using the railroad is controlled by an independent authority that makes sure that there are no trains colliding with each other.

At most four cars can cross the railroad crossing at the same time. Cars can starve while crossing the railroad tracks; trains never do.



Your task is to give a concurrent process management algorithm to the Railroad Crossing Problem where trains and cars are processes. The train approaching the crossing activates the light that changes from green to yellow. The change of the light from yellow to red is activated either by the last car crossing the railroad (at most four at a time) or, if no car is passing the crossing at the time, by the train approaching the crossing. The change of the light from the red to green is activated by the train that is finishing its crossing. Your algorithm does not know the total number of trains and cars in the system.

- (a) Give a conditional critical region-based algorithm (solution) to the Railroad Crossing Problem. Explain your algorithm, and explicitly specify any assumptions you make about the model.
- (b) Give a monitor-based algorithm (solution) to the Railroad Crossing Problem. Specify the use of the monitor procedures by cars and trains. Explain your algorithm, and explicitly specify any assumptions you make about the model.