Monitors and Priority Waits

This is an example of a monitor using priority waits. It implements a simple alarm clock; that is, a process calls `alarmclock.wakeme(n)`, and suspends for $n$ seconds. Note that we are assuming the hardware invokes the procedure `tick` to update the clock every second.

```pseudocode
1 alarmclock: monitor;
2 var now: integer;
3     wakeup: condition;
4 procedure entry wakeme(n: integer);
5     begin
6         alarmsetting := now + n;
7         while now < alarmsetting do
8             wakeup.wait(alarmsetting);
9         wakeup.signal;
10     end;
11 procedure entry tick;
12     begin
13         now := now + 1;
14         wakeup.signal;
15     end.
```

*lines 2-3* Here, `now` is the current time (in seconds) and is updated once a second by the procedure `tick`. When a process suspends, it will do a wait on the condition `wakeup`.

*line 6* This computes the time at which the process is to be awakened.

*lines 7-8* The process now checks that it is to be awakened later, and then suspends itself.

*line 9* Once a process has been woken up, it *signals* the process that is to resume next. That process checks to see if it is time to wake up; if not, it suspends again (hence the `while` loop above, rather than an `if` statement). If it is to wake up, it *signals* the next process...

*line 14* This is done once a second (hence the addition of 1 to `now`). The processes to be woken up are queued in order of remaining time to wait with the next one to wake up first. So, when `tick` signals, the next one to wake up determines if it is in fact time to wake up. If not, it suspends itself; if so, it proceeds.