T-106.5600 Concurrent Programming Examination December 2011/ hsa

Be concise and clear. The shorter you can be the better. The simplicity of the solutions and answers is an important grading criterion!

1. Show with a proper invariant that the following algorithm solves the safety criterion of two-process critical section problem. Does it satisfy the necessary liveness requirements? Prove or disprove.

boolean wantp ← false, wantq ← false			
	p		**q**
	loop forever		loop forever
p1:	non-critical section	q1:	non-critical section
p2:	wantp ← truc	q2:	wantq ← truc
p3:	while wantq	q3:	while wantp
p4:	wantp ← false	q4:	wantq ← false
p5:	wantp ← true	q5:	wantq ← true
p6:	critical section	q6:	critical section
p7:	wantp ← false	q7:	wantq ← false

2. A binary semaphore may take only the integer values 0 and 1, so a signal on it when it's at 1 has no effect. Prove or disprove the correctness of the following algorithm for implementing a general semaphore using binary semaphores:

binary semaphore S ← 1, gate ← 0 integer count ← 0	
	wait
p1:	wait(S)
p2:	count ← count − 1
p3:	if count < 0
p4:	signal(S)
p5:	wait(gate)
p6:	else signal(S)
	signal
p7:	wait(S)
p8:	count ← count + 1
p9:	if count ≤ 0
p10:	signal(gate)
p11:	signal(S)

3. Solve the five dining philosophers problem using tuple-space, so that the resource utilization rate is maximized. Define clearly the meaning of the different tuple types used and attach appropriate tags to them. No global variables except tuples should be used. The simpler your solution is the better. Discuss the pros and cons of your solution.

Use tuple-space primitives are: `postnote ('tag', ...)`, `readnote ('tag', ...)`, `removenote ('tag',...)`. Indicate clearly in a `readnote(...)`, or `removenote(...)` operation when a matching tuple containing element equal to the value of a program variable v value is sought for with syntax " v=", from the case where an element value of a otherwise matching tuple is just assigned to a program variable (syntax " v").

4. You are asked to write in Java a simple barrier synchronization monitor, which has only one method, `Wait_for_all_n()` with the following semantics. The calling thread is put on wait "behind the barrier" if there are less than n-1 other threads waiting behind it. Otherwise "the barrier is opened", i.e. the thread will cause all the n-1 waiting threads to wake up and to pass the "barrier" together with it, whereas all potential newcomers will stop to wait for the next n threads to arrive behind the barrier before proceeding in due time.

```
Class Simple_barrier {
      …
      synchronized void Wait_for_all_n() {
      …
      }
}
```

Specify the criterions of "Safety" and "No unnecessary waiting" for your solution in terms of appropriate invariants and argue why they are maintained.

5. A process receives data on two input channels where each input channel may receive data at any rate from "never" to "all the time". The process interleaves the data on one output channel. Develop an algorithm to achieve a *fair merge* that is free from starvation of both input channels. Use the pseudo-code of the textbook.