T-106.5600 CONCURRENT PROGRAMMNG EXAM 16.12.2013

PLEASE, BE CONCISE AND WRITE CLEARLY! The shorter you can be without loosing clarity the better scores you will get.

1. Concepts

Select three topics from the following list and write a short definition and description of it. Use not more than one hundred words (articles and prepositions excluded) per topic for your answers:

a) Differences between processes and threads
b) Amdahl's law and scalability
c) Explicit and implicit concurrency
d) Task parallelism and data parallelism
e) Deadlock, livelock and starvation

2. Mutual exclusion

Examine the following mutual exclusion algorithm for N threads and answer all three questions. Prove your answer or provide an execution trace that shows that the property is false.

```
integer turn <- 0
boolean busy <- false

loop forever - for each thread Pi
1     do {
2          do {
3               turn <- i
4          } while (busy)
5          busy <- true
6     } while (turn != i)
7     critical section
8     busy <- false
```

1) Does the algorithm satisfy mutual exclusion?
2) Is the algorithm free from starvation?
3) Is the protocol free from deadlocks?

3. Verification of Safety and Liveness

Give an axiomatic proof of the a) safety and b) liveness of the following algorithm for critical section:

Algorithm 3.15: Doran-Thomas algorithm	
boolean wantp←false, wantq ← false integer turn←1	
p	**q**
loop forever	loop forever
p1: non-critical section	q1: non-critical section
p2: wantp←true	q2: wantq←true
p3: if wantq	q3: if wantp
p4: if turn = 2	q4: if turn = 1
p5: wantp←false	q5: wantq←false
p6: await turn=1	q6: await turn=2
p7: wantp←true	q7: wantq←true
p8: await wantq=false	q8: await wantp=false
p9: critical section	q9: critical section
p10: wantp ← false	q10: wantq ← false
p11: turn←2	q11: turn←1

4. Java Monitor for Readers and Writers

```
public interface ReadWriteLock {
      void writerLock();
      void writerRelease();
      void readerLock();
      void readerRelease();
}
```

You have been given the following Java interface:

a) Implement the interface using the Java synchronization primitives `wait()`, `notify()` and `notifyAll()`. Multiple readers may run concurrently, but a writer must exclude all other actions, including other writers. You may ignore exception handling. You can assume that readers and writers adhere to the following access protocol:

```
rwlock.writerLock();
//do writing
rwlock.writerRelease();
```

```
rwlock.readerLock();
//do reading
rwlock.readerRelease();
```

b) Modify your solution so that writers are given priority over readers.

Your solutions should take less than 30 lines of code and it should fit with explanations in a sheet of paper.

5. A General Semaphore in a Tuple Space

Implement a general semaphore in Tuple Space using just one tuple.