NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING

EXAMINATION FOR
Semester 2 AY2012/2013

CS3211 – PARALLEL AND CONCURRENT PROGRAMMING

May 2013 Time Allowed: 2 Hours

INSTRUCTIONS TO CANDIDATES

1. This examination paper contains **five (5)** questions and comprises **TEN (10)** printed pages, including this page.
2. Answer **ALL** questions within the space in this booklet
3. This is an Open Book examination.
4. Please write your Matriculation Number below.

MATRICULATION NO: ______________________________

This portion is for examiner's use only

Question	Marks	Remarks
1	/ 10	
2	/ 10	
3	/ 5	
4	/ 10	
5	/ 10	
Total	/ 45	

Question 1 [10 marks]

Consider an array X storing the names of people working in A*STAR, another array Y storing the names of alumni of NUS School of Computing, and another array Z of people living in the Clementi area of Singapore. All three arrays are sorted in alphabetical order. There exists at least one person whose name exists in all the three arrays (there may be more).

A. Write a Promela process to find out the alphabetically first such person, whose name appears in all the three arrays.
B. Develop a concurrent solution in Promela with several processes co-operating to solve the problem.

-BLANK PAGE-

Question 2 [10 marks]

Suppose there are n passenger threads and a car thread. The passengers repeatedly wait to take roller coaster rides in the car, which can hold C passengers, where C < n. The car can go round the tracks only when it is full. Moreover:

- Passengers should invoke `board` and `unboard`.
- The car should invoke `load`, `go_around` and `unload`.
- Passengers cannot `board` until the car has invoked `load`.
- The car cannot depart until C passengers have boarded (also stated in the text above).
- Passengers cannot `unboard` until the car has invoked `unload`.

Write a multi-threaded Java program for the passengers and car that enforces these constraints.

BLANK PAGE

Question 3 [5 marks]

Following is the code executed by process id, for 1 ≤ id ≤ N. Note that x is a shared variable where 0 ≤ x ≤ N. Any atomic operation is enclosed in angle brackets. The `skip` statement does nothing. Assume that each process is executing on a separate processor, so we have truly parallel execution of the processes. Atomic execution of an operation here means that its effect is visible to all the processes on all processors. The variable x is simply stored in a shared memory which is accessible by all the processors.

```
repeat
     while <x != 0> <skip>;
     < x = id>; < delay >
until < x == id>;
CRITICAL SECTION
< x = 0 >;
```

Is mutual exclusion preserved when there is no `delay` operation, that is, delay == 0? Does the preservation of mutual exclusion get altered for higher values of delay? Give detailed comments.

Question 4 [10 marks]

A propositional logic formula is made out of atomic propositions, binary operators $\wedge$, $\vee$ (for logical AND, OR) and the unary operator $\neg$ (logical NOT). The Sharp-SAT or #SAT problem for propositional logic is the problem of counting the number of satisfying assignments of a given propositional logic formula. Write an MPI program which efficiently solve the #SAT problem for a given propositional logic formula, by taking advantage of parallelism. Give detailed comments on how your program is exploiting parallelism.

BLANK PAGE

Question 5 [10 marks]

A. Two students are concurrently taking the CS3211 final exam. The pseudo-code executed by the two students are as follows. The number N is a non-negative integer. Note that s1 and s2 capture whether the two students are in/out of the exam hall. Initially, s1 == in, s2 == in, and N == 1. For each student process, each iteration of the `do-forever` loop is executed atomically.
Can you draw a global finite state machine describing the behavior of the concurrent system? Use the finite state machine to reason that two students are not out of the exam hall at the same time.

```
do forever{
   if s1 = in and N is odd
      {s1 := out; }
   else if s1 = out
      { s1 := in; N := 3*N+1}
   else {do nothing }
}
```

```
do forever{
      if s2 = in and N is even
                 {s2 := out; }
      else if s2 = out and N is even
                 { s2 := in ; N := N/2 }
      else { do nothing }
}
```

B. For the concurrent system in previous page, develop process equations for the individual processes. Use the process equation notation used throughout the semester in our class. What are the processes you need, and how would you show their interactions? Explain each process, and each action label you use.

END of PAPER