NATIONAL UNIVERSITY OF SINGAPORE
SCHOOL OF COMPUTING

EXAMINATION FOR
Semester 2, 2006/2007
CS5219 - AUTOMATED SOFTWARE VALIDATION

April 2007 Time Allowed: 2 Hours

INSTRUCTIONS TO CANDIDATES

1. This examination paper contains **four**(**4**) long questions and comprises **ten** (**10**) pages.
2. Answer **ALL** questions in the space provided in this booklet.
3. **ALL** answers must come with the correct explanations. There is no credit for blind guesses.
4. This is an **OPEN BOOK** examination.
5. **Please write your Matriculation Number below.**

MATRICULATION NO.:

(This portion is reserved for the examiner's use only)

Question		Marks	Remark
Question A	15		
Question B	13		
Question C	12		
Question D	10		
Total	50		

A. Theorem Proving
(10 + 5) = 15 marks

1. Formally prove that each of the following programs computes in z the product of x and y. That is, at the end of the program $z = x0 * y0$ where $x0$ and $y0$ are the initial values of variables x and y. You may assume that $odd(x)$ is a function which returns true if x is odd and false otherwise; the code for this function is not shown. Also, x, y, z are non-negative integers; the operation / denotes integer division, that is, it returns an integer. You may use the rules of Hoare logic, or you can present a handcrafted formal proof.

```
z = 0;               z = 0;
while (x != 0){      while (x != 0){
   z = z + y;           if odd(x){ z = z + y;}
   x = x - 1;           y = y * 2; x = x/2;
}                    }
```

2. The PVS theorem prover employs a sequent calculus for constructing proofs. The following is a simple proof fragment taken from the PVS manual. Show the soundness of these steps by converting each of the sequents to a first order logic formula and establishing their equivalence.

```
|----------------
{1} (FORALL x: P(x) AND Q(x)) IMPLIES (FORALL x: P(x)) AND (FORALL x: Q(x))
Rule? (flatten)

{-1} (FORALL x: P(x) AND Q(x))
    |--------------------------------
{1}  (FORALL x: P(x)) AND (FORALL x: Q(x))
Rule? (split)
this yields two subgoals

[-1] (FORALL x: P(x) AND Q(x))
    |--------------------------------
{1}  (FORALL x: P(x))

[-1] (FORALL x: P(x) AND Q(x))
    |--------------------------------
{1}  (FORALL x: Q(x))
```

B. **Model Extraction and Refinement**
(4 + 6 + 3) = 13 marks

1. Consider the program fragment `x = 5; x = x +1 ; x = x - 1; y = x`

   Suppose we want to prove that $y = 5$ at the end of the program. Construct a predicate abstraction that is insufficient to prove this property. Also construct a predicate abstraction that is sufficient to prove the property. You are only allowed to abstract the data store of the program via predicates, but the control flow should not be abstracted.

2. Construct the finite state transition systems resulting from the two predicate abstractions in part 1 of this question.

-BLANK PAGE-

3. Consider predicate abstraction of a C program (without pointers) with a *single path*, i.e. there are no branches in the program. You can assume the predicate abstraction scheme discussed in class.

   - Show that the abstracted program can have multiple paths. Give concrete examples and explain the reason for proliferation of program paths.
   - Proliferation of paths (due to predicate abstraction) raises the time overheads for verification. So, what is the benefit in performing predicate abstraction prior to software model checking?

C. **Property Specification**

(8 + 4) = 12 marks

1. Assume that p is an atomic proposition. What can you say about the equivalence of the following pairs of temporal formulae? If they are equivalent, then provide a formal proof. If not construct an example Kripke Structure to show that they are not equivalent.

   (a) the LTL formula **GF**p and the CTL^* formula **AGF**p

   (b) the CTL formulae **AGAF**p and the CTL formula **AGEF**p

   (c) the LTL formula **GF**p and the CTL formula **AGAF**p

2. Suppose we want to verify in SPIN the LTL formula $\mathbf{G}(p \Rightarrow \mathbf{F}q)$ for a Promela program P, where p, q are atomic propositions. What is the property automata (internally constructed by SPIN) that is synchronously composed with the state transition system of P ?

D. **Miscellaneous**
6 + 4 = 10 marks

1. Suppose you want to use a model checker (such as the SPIN tool discussed in class) to generate test cases of a terminating sequential program written in a C-style imperative language. How can you do so to meet the *statement coverage*, *edge coverage* and *condition coverage* criteria for test generation? Discuss in details.

2. Consider a multi-threaded Java program where n threads running on a single processor are trying to access a shared object using a round-robin scheme. We want to prove mutual exclusion of access of the shared object for any n. Can we employ the abstraction refinement based software verification discussed in class ? Justify your answer. If your answer is yes, explain how. If your answer is no, can you suggest any alternative verification methods ?

-END OF PAPER-