CS615 Autumn 2006 End-semester Exam

Time: 180 mins   Total marks: 40

- ***You must write your answers only in the spaces provided.***
- ***The exam is open book and notes.***
- ***Results/proofs covered in class/problem sessions/assignments may simply be cited, unless specifically asked for.***
- ***Unnecessarily lengthy solutions will be penalized.***
- ***If you need to make any assumptions, state them clearly.***
- ***Do not copy solutions from others or indulge in unfair means.***

1. Consider the following program P with location labels Li, in which all variables are of `integer` type:

```
L0:  i := 0;
L1:  y := n;
L2:  while (y <= 10) {
L3:    y := x * y;
L4:    i := i + 1;
L5:  }
L6:  i := i*i;
```

A student has conjectured (from a large number of trial and errors and informal reasoning!) that whenever the program is started with $(x \leq -2) \wedge (0 < n < 10)$, it terminates and the resulting value of i is at most 16. We wish to prove the partial correctness of this statement formally using our knowledge of Hoare logic. In other words, we are interested in proving that if P terminates when started with $(x \leq -2) \wedge (0 < n < 10)$, then $i \leq 16$ on termination.

(a) *[(2 marks)]* Give a loop invariant λ possibly (but not necessarily) involving all variable(s) `x, y, i, n` that will allow you to prove the above.

(b) *[(4+4 marks)]* Show that the following Hoare triples are valid.

- $\{\lambda\}$ `fragment_of_P with L2,L3,L4,L5` $\{\lambda\}$.
- $\{\lambda \wedge (y > 10)\}$ `i:= i*i` $\{i \leq 16\}$.

2. *[(10 × 1 marks)]* Consider the following program P with location labels `Li`.

```
L0:  x := y;
L1:  while (y >= 10) {
L2:    x := x + 1;
L3:    y := y - 1;
L4:    x := x + y;
L5:  }
```

We wish to construct a Boolean program using the following predicates such that the traces of P' are captured as accurately as possible by the traces of the Boolean program.

Boolean variable	Predicate
b_1	$y \geq 10$
b_2	$x \geq 10$

The Boolean program is obtained by giving expressions (`Ei`) in the following skeleton.

```
L0':    b1, b2 := E1, E2;
L1':    while (*) {
L1'':     assert(b1);
L1''':    b1, b2 := E3, E4;
L2':      b1, b2 := E5, E6;
L3':      b1, b2 := E7, E8;
L4':      b1, b2 := E9, E10;
L5':    }
```

Give each `Ei` to complete the Boolean program.

3. Consider the following program with location labels `Li`. All variables are of the `integer` type.

```
L0:  x := y;
L1:  while (y <= 10000) {
L2:    x := x + 1;
L3:    y := y + 1;
L4:  }
```

We wish to analyze this program using the abstract domain of intervals (or rectangles). Thus, our abstract domain is the set of tuples (a, b, c, d) where a, b, c, d are integers or $+\infty$ or $-\infty$. For every set S of integer pairs (x, y) in the concrete doamin, $\alpha(S) = (\min(x), \max(x), \min(y), \max(y))$ where the minimization and maximization are over all integer pairs in S, and for every four-tuple (a, b, c, d) in the abstract domain, $\gamma((a, b, c, d)) = \{(x, y) \mid (a \leq x \leq b) \wedge (c \leq y \leq d), x, y \text{ integers}\}$. The *lub* and *glb* operators to be used in the abstract domain are those that have been discussed in class. The widening operator is defined as follows:
$(a_1, b_1, c_1, d_1) \nabla (a_2, b_2, c_2, d_2) = (a_3, b_3, c_3, d_3)$ where

- if $a_1 \leq a_2$ then $a_3 = a_1$, else if $(-5000 \leq a_2 < a_1)$ then $a_3 = -5000$, else $a_3 = -\infty$.
- if $b_1 \geq b_2$ then $b_3 = b_1$, else if $(b_1 < b_2 \leq 5000)$ then $b_3 = 5000$, else $b_3 = +\infty$.
- if $c_1 \leq c_2$ then $c_3 = c_1$, else if $(-10000 \leq c_2 < c_1)$ then $c_3 = -10000$, else $c_3 = -\infty$.
- if $d_1 \geq d_2$ then $d_3 = d_1$, else if $(d_1 < d_2 \leq 10000)$ then $d_3 = 10000$, else $d_3 = +\infty$.

Starting with the precondition $(0 \leq x^2 + y^2 \leq 100)$, we wish to compute the abstract loop invariant at `L1` *by executing the program in the abstract domain.*

(a) *[(2 marks)]* Give the abstract precondition of the program.

(b) *[(2 marks)]* Compute the abstract postcondition at the end of the first iteration of the loop.

(c) *[(3 marks)]* Compute the loop invariant obtained by using the widening operator defined above. You must use the widening operator everytime you wish to compute an upper bound of two abstract elements at location `L1`

(d) *[(3 marks)]* Compute the concrete loop invariant at `L1` that would be obtained if we concretize the abstract loop invariant computed in the previous subquestion and push it through the body of the loop exactly once.

4. Consider the following two programs that read and write the same shared Boolean variables `x` and `y`. The programs are run *concurrently*, i.e. a sequence of instructions of one program can be interleaved in any arbitrary way with a sequence of instructions of the other program. However, at any time, the instruction of only one program can be executed. Assume that each instruction of each program executes atomically.

```
Program P1: do forever                      Program P2: do forever
            L1:  if (x) y = !x;                         L3: if (y) x = !y;
            L2:  if (x) x = y;                          L4: if (y) y = x;
```

   (a) *[(5 marks)]* Construct a Kripke structure representing the behaviour of the two programs running concurrently. You may assume that the shared variables are both set to `true` to begin with. In describing the Kripke structure, you must simply specify what are the values of the different state variables in each state, and which other states it has an edge to. Program P1 starts in location `L1` and program P2 in location `L3`.

   (b) *[(5 marks)]* Use the CTL model checking algorithm discussed in class to check if **AG EF** $(x \wedge y)$ is true in the starting state of the above Kripke structure.