## Mock Test for Software Verification (L19)

Matthew Parkinson

March 12, 2010

## Instructions

Answer all questions in Part A. Answer two questions from Part B.

## Part A

Question 1. Prove the following, or say why they are not true:

- 1.  $\{x = 5\}$  x := 3  $\{x = 3\}$
- 2.  $\{even(\mathbf{x})\}$   $\mathbf{x} := \mathbf{x} + \mathbf{1}\{odd(\mathbf{x})\}$
- 3.  $\{x = 5\}$  skip  $\{odd(x)\}$

Give complete details of every rule used. Justify each logical implication you required in the rule of consequence. [6 marks]

Question 2. Prove the following:

- 1.  $\{true\}$  while true do skip  $\{false\}$
- 2.  ${x = 5}$  while x=5 do (continue; x:=3)  ${false}$

[4 marks]

Question 3. Prove the following:

1.  $\{a[x] = 3\}x := a[x] \{x = 3\}$ 2.  $\{true\}y := a[x]; a[y] := y\{a[x] = a[y]\}$ 

[4 marks]

Question 4. Prove the following program:

```
{true}
let f(x) =
    if x = 0 then 1
    else
        local y in y := f(x-1);
        return x*y; in
    z := f(2)
{z = 2}
```

Explain which function definition and call rules you use, and why. You may assume the following logical facts:

$$\begin{aligned} &fact(0) = 1 \\ &\forall n. \; fact(n+1) = (n+1) * fact(n) \end{aligned}$$

[8 marks]

Question 5. Using separation logic prove the following

 $\begin{aligned} 1. & \{ \mathbf{x} \mapsto 5 \} \quad [\mathbf{x}] := 6 \quad \{ \mathbf{x} \mapsto 6 \} \\ 2. & \{ \mathbf{x} = \mathbf{y} \land \mathbf{x} \mapsto \mathbf{z} \} \quad \mathbf{x} := [\mathbf{x}] \quad \{ \mathbf{y} \mapsto \mathbf{z} \land \mathbf{x} = \mathbf{z} \} \end{aligned}$ 

Give full outlines and make it clear each rule that you use. [6 marks]

Question 6. Verify using separation logic

[6 marks]

**Question 7.** Using separation logic, verify the following, or state why it is not possible:

1.  $\{x \mapsto ...\}$   $[x] := 5 || [x] := 5 \{x \mapsto 5\}$ 2.  $\{x \mapsto ... \land f \neq g\}$ if f = 1 then [x] := 5|| if g = 1 then [x] := 5 $\{x \mapsto ...\}$ 

[8 marks]

Question 8. Using the Owicki/Gries method verify

$$\{ \mathbf{x} = 0 \} \\ \mathbf{x} := \mathbf{x} + \mathbf{3} \mid | \mathbf{x} := \mathbf{5} \\ \{ \mathbf{x} = 8 \lor \mathbf{x} = 5 \}$$

[10 marks]

## Part B

Question 9. (a) The conjunction rule is:

$$\frac{\{P_1\} C \{Q_1\}}{\{P_2\} C \{Q_2\}} \\ \overline{\{P_1 \land P_2\} C \{Q_1 \land Q_2\}}$$

Prove the conjunction rule is sound for sequential Hoare logic. That is, show:

$$\models \{P_1\} C \{Q_1\} \land \models \{P_2\} C \{Q_2\}$$
$$\implies \models \{P_1 \land P_2\} C \{Q_1 \land Q_2\}$$

(b) The disjunction rule is:

$$\begin{array}{c} \{P_1\}\, {\tt C}\, \{Q_1\} \\ \{P_2\}\, {\tt C}\, \{Q_2\} \\ \hline \{P_1\vee P_2\}\, {\tt C}\, \{Q_1\vee Q_2\} \end{array}$$

Prove the disjunction rule is sound for sequential Hoare logic.

**Question 10.** Verify the following programs meet the given specifications: (a)

```
{true}

r := x;

d := 0;

while r >= y do

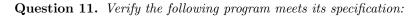
r := r - y;

d := d + 1

{x = (d \times y) + r \wedge r < y}
```

*(b)* 

```
{list(x)}
h := new;
t := h;
while true do
p := t
v := [x+1];
[p+1] := v;
x := [x];
if x==0 then break;
t := new;
[p] := t;
[p] := 0;
{list(x) * list(h)}
```



```
\begin{array}{l} \{ \mathbf{x} = 0 \} \\ \text{if } \mathbf{x} = 0 \text{ then } \mathbf{x} := \mathbf{x} + 1 \\ | | \\ \text{if } \mathbf{x} = 0 \text{ then } \mathbf{x} := \mathbf{x} + 2 \\ | | \\ \text{if } \mathbf{x} = 0 \text{ then } \mathbf{x} := \mathbf{x} + 1 \\ \{ \mathbf{x} \in \{1, 2, 3, 4\} \} \end{array}
```

You will have to add auxiliary state to the program. You may use either Owicki/Gries method or rely-guarantee. **Question 12.** Consider the following programming language with loops and rapid exits from the loops using break and continue:

 $C::= \ x:=E \mid C; C \mid \textit{if } B \textit{ then } C \textit{ else } C \mid \textit{while } B \textit{ do } C \textit{ break } n \mid \textit{continue } n$ 

The command break 1 breaks out of the directly enclosing loop. The command break n break out of n enclosing loops. The command continue 1 restarts the directly enclosing loop. The command continue n restarts the nth enclosing loop.

(a) Extend the basic Hoare logic with breaks and continues to deal with additional parameter for the level of loop it applies to. Give rules for both while-do and break n and continue n. [Hint: Consider extending the context to carry multiple break and continue contexts for each level of loop.]

(b) Verify the following program with your extended logic