COP5025 Spring 2001 – Final Exam (Chs. 8+9+Prolog)

Name: _____ (Please print)

You can put the answers on these sheets. Use additional sheets when necessary. If possible, show how you derived your answers (this is helpful for partial credit). You can collect 100 points in total for this exam. A bonus question is included for an additional 15 points. **This exam is open book and open notes.**

1. (10 points) Which of the following pairs of Prolog terms unify and what are the resulting value bindings of the Prolog variables? Recall that Prolog variables start with an upper-case letter, while Prolog atoms (named constants) start with a lower-case letter.

Term 1	*Term 2*	*Unify? Yes/No*	*Variable Bindings*
`john`	`john`		
`john`	`peter`		
`john`	`X`		`X =`
`city(new_york)`	`city(X)`		`X =`
`state(name(florida), capital(tallahassee))`	`state(name(X), capital(Y))`		`X =` `Y =`
`state(name(florida), capital(tallahassee))`	`state(name(X), capital(X))`		`X =`
`state(name(new_york), capital(new_york))`	`state(name(X), capital(X))`		`X =`
`state(name(X), capital(new_york))`	`state(name(Y), capital(Y))`		`X =` `Y =`

2. (10 points) Consider the language of *simple expressions* (lecture note 95 and Section 8.6) What functions do the following expressions denote?

   (a) **let** $x = 1$ **in** $x + y$ **end**

   (b) **let** $y = 1$ **in let** $x = y + 1$ **in** $x - y$ **end end**

3. (10 points) Consider the language of *simple expressions with error values* (notes 96–97 and Section 8.7) What functions do the following expressions denote?

   (a) x

   (b) **let** $y = 1$ **in** x/y **end**

4. (15 points) Consider the C conditional-and operation `&&` with the usual semantics:

   $(x \texttt{\&\&} y) = 0 \quad \text{if} \quad x = 0$
   $(x \texttt{\&\&} y) = 0 \quad \text{if} \quad x \neq 0 \text{ and } y = 0$
   $(x \texttt{\&\&} y) = 1 \quad \text{if} \quad x \neq 0 \text{ and } y \neq 0$

   Give a denotational semantics description of the C conditional-and operator with this behavior. The language of simple expressions is used (note 95 and Section 8.6), extended with a grammar production of the form

   $E \rightarrow E_1 \ \texttt{\&\&} \ E_2$

5. The order of operand evaluation in C/C++ expressions is undetermined. We can describe such a "random" reordering with denotational semantics:

$\mathcal{M}[\![E_1 + E_2]\!]\ \rho\ \sigma$

```
= if    random
  then let  ⟨n, σ'⟩ = M[[E1]] ρ σ
       in   let  ⟨m, σ''⟩ = M[[E2]] ρ σ'
            in   ⟨n + m, σ''⟩
            end
       end
  else let  ⟨m, σ'⟩ = M[[E2]] ρ σ
       in   let  ⟨n, σ''⟩ = M[[E1]] ρ σ'
            in   ⟨n + m, σ''⟩
            end
       end
```

where *random* evaluates to TRUE or FALSE depending on the "flip of a coin".

(a) (15 points) Prove that $\mathcal{M}[\![E_1 + E_2]\!]\ \rho\ \sigma$ always returns the same result if E_1 and E_2 do not change the state of the machine. That is, $\mathcal{M}[\![E_1]\!]\ \rho$ and $\mathcal{M}[\![E_2]\!]\ \rho$ applied to a state return a tuple with the state unchanged. For this proof, show that $\mathcal{M}[\![E_1 + E_2]\!]\ \rho\ \sigma$ results in a tuple $\langle k, \sigma \rangle$ with some value k and unchanged state σ.

(b) (10 points) Suppose that E_1 and/or E_2 do change the state of the machine, i.e. the expressions have side-effects. Then $\mathcal{M}[\![E_1 + E_2]\!]$ is possibly nondeterministic.

Definition 1
A program P is deterministic if its denotation $\mathcal{P}[\![P]\!]$ is a deterministic function.

Definition 2
A program P with input n is deterministic if $\mathcal{P}[\![P]\!]$ n is deterministic.

Consider the following program:

```
program (x); x:=x+(x:=0) end.
```

That is, the second operand of the addition assigns 0 to `x`. The semantics of this language are described in Section 8.10 and the new semantic function for addition is given above. Answer the following questions without deriving the result with denotational semantics:

- Is this program deterministic in general?
- Is this program with input 0 deterministic?

6. (10 points) Which of the following Hoare triples are valid?

(a) $\{y = a\} \quad x := x + y \quad \{x = a\}$

(b) $\{x = 0\} \quad$ **while** FALSE **do** $x := x + 1 \quad \{x \leq 0\}$

(c) $\{x > 0\} \quad$ `if` $x = 0$ `then` $x = y$ `else` $y = x \quad \{y > 0\}$

(d) $\{x = 1 \,\&\, y = 2\} \quad z := y \quad \{y = 2\}$

7. The following algorithm computes $s = \sum_{i=0}^{n} i$ by summing over $i = 0, \ldots, n$:

```
i:=0;
s:=0;
while i < n do
  i:=i+1;
  s:=s+i
end
```

(a) (20 points) Find the weakest precondition of this program and proof it's correctness by annotating the program with conditions. The postcondition is $s = \frac{1}{2} * n * (n+1)$ and the loop invariant is $I = (s = \frac{1}{2} * i * (i+1)\ \&\ i \leq n)$. (Note: the postcondition and loop invariant are based on the identity $\sum_{i=0}^{n} i = \frac{1}{2} * n * (n+1)$, $(n \geq 0)$.)

8. **bonus question** (15 points)

Consider the procedure:

procedure s**(in** c**, inout** x**, inout** y**);**
  if c **then**
    (x, y) **:=** (y, x)
  else
    skip

Derive the weakest precondition of the following program fragment:

call $s(a < b, a, b)$
$\{a \geq b\}$

The proof rule for this procedure is:

$$\frac{\{P\}\ \textbf{if}\ c\ \textbf{then}\ (x, y) := (y, x)\ \textbf{else skip}\ \{R[(a, b) := (x, y)]\}}{\{P[(c, x, y) := (a < b, a, b)]\}\ \textbf{call}\ s(a < b, a, b)\ \{R\}}$$

(Hint: First derive P from the if-then-else with postcondition $R[(a, b) := (x, y)]$ where $R = (a \geq b)$.)