COP5025 Spring 2002 – 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. (15 points) Consider the following Prolog program:

```
mother(john, jane).
mother(mike, jane).
mother(jane, rose).
mother(rick, rose).
father(john, rick).
father(mike, alex).
father(rick, will).
parent(X, Y) :- mother(X, Y).
parent(X, Y) :- father(X, Y).
ancestor(X, Y) :- parent(X, Z), ancestor(Z, Y).
ancestor(X, Y) :- parent(X, Y).
```

What do the following queries return? Indicate whether the query is successful and if so the (first) binding found for the variable `X`.

Query	*Success? Yes/No*	*Variable Binding*
`father(rick, X)`		`X =`
`parent(john, X)`		`X =`
`parent(X, rose)`		`X =`
`parent(X, X)`		`X =`
`ancestor(john, X)`		`X =`

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

   (a) $x - x$

   (b) $(\mathbf{let}\ x = 0\ \mathbf{in}\ x\ \mathbf{end}) - 1$

3. (15 points) Consider the language of *simple expressions with error values* (notes 96–97 and Section 8.7) What functions do the following program and expression denote?

   (a) $\mathbf{program}(x);\ x + 1\ \mathbf{end}.$

   (b) $\mathbf{let}\ y = 0\ \mathbf{in}\ 1/y\ \mathbf{end}$

4. We define denotational semantics for code C of a stack machine with operations M:

$$
\begin{array}{lcl}
C & \rightarrow & \textbf{begin } M \textbf{ end} \\
M & \rightarrow & \textbf{load } \alpha \\
M & \rightarrow & \textbf{store } \alpha \\
M & \rightarrow & \textbf{push } n \\
M & \rightarrow & \textbf{pop} \\
M & \rightarrow & \textbf{add} \\
M & \rightarrow & M\ M
\end{array}
$$

Explanation:

- **load** loads the value located at address α and pushes it on the stack.
- **store** stores the top of the stack at address α and pops the value.
- **push** pushes a constant integer value on the stack.
- **pop** pops the top value from the stack.
- **add** pops the top two elements and pushes the sum.

The value domains are:

$$
\begin{array}{rl}
s, n \in \textbf{Int} = \{\ldots, -2, -1, 0, 1, 2, \ldots\} & \text{integers} \\
\alpha \in \textbf{Loc} & \text{locations} \\
\kappa \in \textbf{Stacks} = \textbf{Int} \rightarrow (\textbf{Int} + \textit{unused}) & \text{stacks} \\
\sigma \in \textbf{States} = \textbf{Loc} \rightarrow (\textbf{Int} + \textit{unused}) & \text{states}
\end{array}
$$

(a) (15 points) Complete the remaining definitions below:

$$
\begin{array}{rcl}
\mathcal{C}[\![\textbf{ begin } M \textbf{ end }]\!] & = & \texttt{let}\ \ \kappa = \lambda\, s\, .\, \textit{unused};\ \sigma = \lambda\, \alpha\, .\, \textit{unused} \\
 & & \texttt{in}\ \ \ \mathcal{M}[\![M]\!]\ 0\ \kappa\ \sigma \\
 & & \texttt{end} \\
\mathcal{M}[\![\textbf{ load } \alpha\ ]\!]\ s\ \kappa\ \sigma & = & \\
\mathcal{M}[\![\textbf{ store } \alpha\ ]\!]\ s\ \kappa\ \sigma & = & \langle s - 1, \kappa, \sigma[\alpha \mapsto \kappa(s-1)] \rangle \\
\mathcal{M}[\![\textbf{ push } n\ ]\!]\ s\ \kappa\ \sigma & = & \langle s + 1, \kappa[s \mapsto n], \sigma \rangle \\
\mathcal{M}[\![\textbf{ pop }]\!]\ s\ \kappa\ \sigma & = & \\
\mathcal{M}[\![\textbf{ add }]\!]\ s\ \kappa\ \sigma & = & \\
\mathcal{M}[\![\ M_1\ M_2\ ]\!]\ s\ \kappa\ \sigma & = & \texttt{let}\ \ \langle s', \kappa', \sigma' \rangle = \mathcal{M}[\![M_1]\!]\ s\ \kappa\ \sigma \\
 & & \texttt{in}\ \ \ \mathcal{M}[\![M_2]\!]\ s'\ \kappa'\ \sigma' \\
 & & \texttt{end}
\end{array}
$$

Note that s denotes the stack pointer. Each definition returns a triple with an updated stack pointer, stack, and state.

(b) (15 points) What does the following code denote?

begin push 7 **store** a **end**

5. (10 points) Which of the following Hoare triples are valid? Show the full derivation of the answer.

   (a) $\{x = 1 \vee y = 2\} \quad z := y \quad \{y = 2\}$

   (b) $\{x = -5\} \quad x := x + y;\ y := y - x \quad \{y = 5\}$

   (c) $\{x = 0\} \quad \textbf{while } x \leq 0 \textbf{ do } x := x + 1 \quad \{x \geq 0\}$ (use invariant "$x \geq 0$")

6. (10 points) We can use Hoare triples to verify signal logic. For example, the Hoare triple $\{P\} \quad x := x \vee y \quad \{x = \text{TRUE}\}$ is valid when P is equal or stronger than the weakest precondition "$(x \vee y) = \text{TRUE}$" of this statement. Which triples below are valid? Show the full derivation of the answer.

   (a) $\{x = \text{TRUE} \,\&\, y = \text{TRUE}\} \quad x := x \,\&\, y \quad \{x = \text{TRUE}\}$

   (b) $\{x = \text{TRUE}\} \quad x := x \vee y \quad \{y = \text{TRUE}\}$

   (c) $\{x = \text{TRUE}\}$ `if` $x \,\&\, y$ `then skip else` $y := x \vee y \quad \{y = \text{TRUE}\}$

7. The rules for axiomatic semantics in the textbook assume that integer representations are unlimited. Consider the modified rule for assignments that can deal with limited *positive* integer representations:

$$\frac{}{\{Q[V := T \bmod N]\} \quad V := T \quad \{Q\}}$$

where N ($N \geq 0$) is the maximum positive integer value that can be represented + 1.

   (a) (5 points) With the rule above, find the weakest precondition P that makes the following Hoare triple valid:

   $\{P\} \quad x := N - 1;\ x := x + 1 \quad \{x = 0\}$

   where N denotes the maximum positive integer value + 1 as in the axiom above.

   (b) (5 points) Find the weakest precondition P that makes the following Hoare triple valid:

   $\{P\} \quad x := x + 1 \quad \{x > 0\}$

8. **bonus question** (15 points)
Derive the weakest precondition of the Indian algorithm for the computation of powers:

z **:= 1;**
p **:=** x**;**
n **:=** y**;**
while $n \neq 0$ **do**
    if even(n) **then**
        p **:=** $p * p$**;**
        n **:=** $\frac{n}{2}$**;**
    else
        z **:=** $z * p$**;**
        n **:=** $n - 1$**;**
end
$\{z = x^y\}$

The loop invariant is $z * p^n = x^y \,\&\, n \geq 0$. Show how you derived your answer by annotating the program with pre- and postconditions for all statements.