Name:
SSN :

CSE 428

Fall 1998

Final Exam

16 December 1998

The exam consists of 10 problems on 9 pages, totaling 200 points. Read each question carefully and use your time judiciously.

Write your name/number on every page.

1. For each of the following grammars, (20 pts)

   - state whether or not it is ambiguous; if it is ambiguous, give a string with more than one parse tree;
   - state any operator precedences which are enforced;
   - state any operator associativities which are enforced;

   Note that even if a grammar is ambiguous, it can still enforce operator precedences and associativities.

   (a)

$$\begin{array}{lcl} E & ::= & E\text{“}*\text{”}F \mid F \\ F & ::= & E\text{“}+\text{”}F \mid F\text{“}-\text{”}G \mid G \\ G & ::= & N \mid Id \mid \text{“(”}E\text{“)”} \end{array}$$

   (b)

$$\begin{array}{lcl} E & ::= & E\text{“}*\text{”}F \mid F \\ F & ::= & F\text{“}+\text{”}G \mid F\text{“}-\text{”}G \mid G \\ G & ::= & N \mid Id \mid \text{“(”}E\text{“)”} \end{array}$$

2. Consider the following program fragment: (20 pts)

```
program main
  x : integer;

  procedure p(a : integer)
  begin
    ??? := ???
  end p;

begin main
  x := 1;
  p(x);
  write(x);
end main;
```

(a) Replace `??? := ???` above with one assignment statement which yields a program that has *different* behaviors (values for `x` printed out) for *call-by-value* and *call-by-reference* parameter-passing modes, but *identical* behaviors for *call-by-value-result* and *call-by-reference* parameter-passing modes.

(b) Give a simple, but general, description for code which can be inserted as the body of `p` such that the resulting program has *identical* behavior under *call-by-value*, *call-by-value-result*, and *call-by-reference* parameter passing. Give your description in terms of the sets of variables that can be read from and written to. Your description should be general enough to cover the most cases of code which could be inserted in `p` and yield identical programs.

3. Give the *most general* types for each of the following function declarations. (20 pts)

(a) `fun com (f,g) x = f(g(x));`

(b) `fun mix(x,y) (f,g) = (f(x),g(y));`

(c) `fun Curry3 f x y z = f(x,y,z);`

(d) `fun unCurry3 f (x,y,z) = f x y z;`

4. Recall the definition of iterators from Assignment 6: (20 pts)

```
fun iter(f,v) 0 = v
   |iter(f,v) n = f(iter(f,v) (n-1), n);

fun gen_iter(f,g,h) (0,a) = g(a)
   |gen_iter(f,g,h) (n,a) = f(gen_iter(f,g,h) (n-1,h(n,a)), n,a);
```

For each of the following functions, give a brief explanation of what it does. (Each performs a function which can be described in a few words.)

(a) `val a = iter(fn(x,y)=>x+(y*y), 0);`

(b) `val b = iter(fn(x,y)=>2*x, 1);`

(c) `val c = gen_iter(fn(x,y,z)=>x, fn(x)=>x, fn(x,y)=>x*y);`

5. Consider the following informal description of IS-trees. (30 pts)

A tree is either (i) a leaf containing a *string*; or (ii) a node containing an *integer* and two IS-trees.

(a) Give an inductive definition of IS-trees.

(b) Give an ML datatype declaration for IS-trees.

(c) Give the principle of induction for proving properties about IS-trees. (I.e., to prove a property P holds for all IS-trees, what must be shown.)

6. An implementation of a *statically-scoped*, block-structured language might use activation records containing both *static links* and *dynamic links*. **Briefly** explain why a dynamically-scoped, block-structured language would not need static links. Describe what is needed in an activation record in this case and how these things are used. (10 pts)

7. Call-by-value and call-by-reference are two common modes of parameter passing. For each mode, give an advantage it has over the other mode: (10 pts)

   (a) Advantage of CBV over CBR:

   (b) Advantage of CBR over CBV:

8. Consider the following typing rule for `select` expressions: (30 pts)

$$\frac{\Gamma \vdash e_0 : \tau_0 \qquad \Gamma[x : \tau_0] \vdash e_1 : \tau \qquad \Gamma[y : \tau_0] \vdash e_2 : \tau}{\Gamma \vdash \texttt{select}(e_0,\, x.e_1,\, y.e_2) : \tau}$$

(a) Below is a fragment the SML typechecker for expressions (from `Analyze1.sml`) which has been extended with an appropriate constructor for `select` expressions. Fill in the case for typechecking `select` expressions. (Recall that the type of `Cxt` is `((string * tp) list)`.)

```
datatype tp = tpint | tpbool | bad;
datatype expression = IConstant of int | BConstant of bool
                    | Plus of (expression * expression) | ...
                    | Select of (expression * string * expression *
                                 string * expression)

fun typeof(IConstant n, Cxt) = tpint
   |typeof(BConstant b, Cxt) = tpbool
   |typeof(Plus(e1,e2), Cxt) =
         if (typeof(e1,Cxt)=tpint andalso typeof(e2,Cxt)=tpint)
         then tpint else bad
    ...

   |typeof(Select(e0,x,e1,y,e2), Cxt) =
```

(b) Below is a fragment of the Prolog typechecker for expressions (from `hw9-Starter.pl`). Fill in the case for typechecking `select` expressions.

```
infer(_,N,int) :- integer(N).
infer(G,plus(E1,E2),int) :- infer(G,E1,int), infer(G,E2,int).
...

infer(G,select(E0,X,E1,Y,E2)) :-
```

9. Each of the following Prolog programs defines a relation `p` between two lists. In each case, briefly and concisely describe the relationship between lists `xs` and `ys` if `p(xs,ys)` is provable. For example, an answer might be *"if* `p(xs,ys)` *is provable then* `ys` *is the reverse of* `xs`. (Note: each of these programs is independent of the others.) (20 pts)

(a)
```
p([],Ys).
p([X|Xs],[X|Ys]) :- p(Xs,Ys).
```

(b)
```
p([],Ys).
p([X|Xs],[Y|Ys]) :- p(Xs,Ys).
```

(c)
```
p([],[]).
p([X|Xs],[X|Ys]) :- p(Xs,Ys).
```

(d)
```
p(Xs,Ys).
p([X|Xs],[X|Ys]) :- p(Xs,Ys).
```

10. Consider the following Prolog program. (20 pts)

```
p :- q, s.
p :- q, r.
q :- t, u.
q :- t.
r :- r, v.
r :- w.
s :- t.
t.
t :- v.
u :- w, u.
v.
w :- fail.
x :- y,p.
y :- x,q.
```

(The proposition `fail` is never provable.)

For each of the following, state whether a standard Prolog implementation (like SWI Prolog) will answer `yes`, `no`, or do something else (not terminate or report an error when runtime space is exceeded). Write `yes`, `no`, or `other` to indicate the behavior.

(a) `p.`

(b) `q.`

(c) `r.`

(d) `u.`

(e) `x.`

11. If your life was a game of *Jeopardy!* what would the categories be? (0 pts)

Merry Christmas! Happy Hanukkah! Happy Kwanzaa! Happy Holidays! Frohe Weihnachten! ¡Feliz Navidad! Joyeux Noël! Glædig Jul! Buon Natale!