Name:
SSN :

CSE 428

Fall 1999

Final Exam

14 December 1999

The exam consists of 8 problems on 8 pages, totaling 200 points. Read each question carefully and use your time judiciously.

Write your name/number on every page.

1. Consider the following program: (20 pts)

```
main() {
 int a,b,c;

  function Rosencrantz(int x, int y) {
    a = 4;
    b = x*y;
    return (x+y);
  }

  a = 1;
  b = 2;
  c = Rosencrantz(a,b);
}
```

What is the final value of c under each of the following conditions:

(a) x is passed by value and y is passed by reference

(b) x is passed by reference and y is passed by value

(c) both x and y are passed by value

(d) both x and y are passed by reference

2. Consider the following three grammars: (20 pts)

Grammar 1

```
E ::= E * F | F
F ::= F + G | F - G | G
G ::= N | Id | ( E )
```

Grammar 2

```
E ::= E + F | E - G | F
F ::= F * G | G
G ::= N | Id | ( E )
```

Grammar 3

```
E ::= E + F | E - F | E * F
F ::= N | Id
```

Given the following sentences (where "a", "b" and "c" can be generated by `Id`), indicate which of the above grammars can generate each sentence.

(a) `(a - b * 2 + c)`

(b) `a * b + ( 2 - c )`

(c) `a + b * c - 2`

(d) `a * b - c`

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

(a) `fun Curry f x y = f(x,y);`

(b) `fun unCurry f (x,y) = f x y;`

(c) `fun swapArgs f x y = f y x;`

(d)
```
fun reversemap (f, nil) = nil
  | reversemap (f, x::xs) = xs@[f x];
```

4. Consider the following program: (20 pts)

```
program main
{
  int x,y,z;

  procedure Othello(int m)
  {
     int z;
     z = m + x;
     y = (x + y + z) * 2;
     print(x,y,z,m);
  }

  procedure Iago(int n)
  {
     int y;
     y = z * 2;
     Othello(n * y);
     print(x,y,z,n);
  }


  x = 1;
  y = 2;
  z = 3;
  Iago(y);
  print(x,y,z);
}
```

(a) Write the output of the program under static scoping call-by-value rules.

(b) Write the output under dynamic scoping call-by-value rules.

5. Consider the following inference rules for typing values (where N is any integer): (20 pts)

$$\frac{}{\mathsf{senv} \vdash N : \mathsf{int}} \qquad \frac{}{\mathsf{senv} \vdash \mathsf{true} : \mathsf{bool}} \qquad \frac{}{\mathsf{senv} \vdash \mathsf{false} : \mathsf{bool}}$$

$$\frac{\mathsf{senv} \vdash v_1 : \tau_1 \quad \mathsf{senv} \vdash v_2 : \tau_2}{\mathsf{senv} \vdash (v_1, v_2) : (\tau_1 \times \tau_2)}$$

Indicate whether each of the following is true or false.

(a) There is an inductively defined set S such that $(v, \tau) \in S$ if and only if $v : \tau$ is provable using only the rules above.

(b) The following definition is equivalent to the rules above:

1. Every integer N has type int.
2. Boolean constants true and false have type bool.
3. If v_1 has type τ_1 and v_2 has type τ_2 then (v_1, v_2) has type $(\tau_1 \times \tau_2)$.

(c) The following grammar generates exactly the typeable values specified by the above rules:

$$V \ ::= \ N \mid \mathsf{true} \mid \mathsf{false} \mid (V, V)$$

(d) The following grammar generates exactly the types which can occur in a provable judgment $v : \tau$ using the above rules:

$$T \ ::= \ \mathsf{int} \mid \mathsf{bool} \mid \mathsf{int} \times \mathsf{int} \mid \mathsf{bool} \times \mathsf{bool} \mid (T)$$

(e) The above inference rules are ambiguous.

6. Recall the definition of `permute` in Prolog: (20 pts)

```
insert(X,L,[X|L]).
insert(X,[Y|L],[Y|K]) :- insert(X,L,K).

permute([],[]).
permute([X|L], L2) :- permute(L,L1), insert(X,L1,L2).
```

Which of the following goals are provable using this set of clauses? If the goal contains variables, give the solutions (instantiations) that Prolog will provide *first*.

(a) `permute([1,2,3],[1,2,3])`

(b) `insert([1],[2,3],[1,2,3])`

(c) `permute(X,Y)`

(d) `permute([1,2,3],Z)`

Assume we reverse the order of the clauses for `insert` in our program:

```
insert(X,[Y|L],[Y|K]) :- insert(X,L,K).
insert(X,L,[X|L]).
```

and give Prolog the following goal:

```
?- permute([a,b],L).
```

Give in order all solutions generated by Prolog (assuming we keep asking for more solutions).

7. Define a class `Semaphore` with the following methods: (20 pts)

   (a) A constructor method with an integer parameter n, representing the initial value of the semaphore.

   (b) A method `down()` which decrements the semaphore if it is positive, and suspends otherwise.

   (c) A method `up()` which increments the semaphore.

8. For each of the following properties/conditions, state (**yes** or **no**) whether it can always be detected *statically* (at compile time) in a statically typed, statically scoped, multi-threaded, higher-order programming language which does not allow implicit type coercions, but does support C++-style generic modules. (20 pts)

   (a) A function can be called with the wrong number of arguments.

   (b) Deadlock cannot occur.

   (c) An arithmetic exception (e.g., overflow, divide by zero) cannot occur.

   (d) Generic modules are always instantiated with appropriate types.

   (e) An undeclared variable is referenced.