CSE 341 Final — Dec 11, 1998

Your name:

This exam is open book, open notes, 110 minutes, 110 points total.

1. (12 points) Consider the following program. This program will print out four values.

```
begin
integer n;
procedure p(j: integer)
    begin
    n := n+1;
    j := 100;
    print(n);
    end;
n := 0;
p(n);  print(n);
p(n);  print(n);
end;
```

   (a) What is the output if the parameters are passed by value?

   (b) What is the output if the parameters are passed by result?

   (c) What is the output if the parameters are passed by value-result?

   (d) What is the output if the parameters are passed by reference?

2. (10 points) Suppose we wanted to make Java into a pure object-oriented language. What changes should be made? What would be the effect of these changes on efficiency and

language consistency?

3. (20 points) Suppose that the following Miranda script has been filed in.

```
plus x y = x+y

append [] ys = ys
append (x:xs) ys = x : append xs ys

my_map2 f [] [] = []
my_map2 f (x:xs) (y:ys) = f x y : my_map2 f xs ys
```

What is the result of evaluating the following Miranda expressions? If there is a compile-time type error, or a run-time error, or a non-terminating computation, say so. If the result is infinite, give the first several values. If the expression is followed by `::`, then give the type, instead of the value.

(a) `plus ::`

(b) `plus 5 ::`

(c) `append ::`

(d) `append [] ::`

(e) `append [3,4] ::`

(f) `append [] [3,4] ::`

(g) `my_map2 ::`

(h) `my_map2 plus ::`

(i) `my_map2 append ::`

(j) `my_map2 plus [1..] [1..]`

4. (18 points) Consider the following definition of the `abs` and `listmax` relations in CLP($\mathcal{R}$).

```
abs(X,X) :- X>=0.
abs(X,-X) :- X<0.

listmax ([A],A).
listmax ([A|Xs],A) :- listmax(Xs,B), B <= A.
listmax ([X|Xs],A) :- listmax(Xs,A), A > X.
```

What answers are produced for the following queries? (Give all of the answers that would be produced by backtracking.)

(a) `abs(-5,A).`

(b) `abs(A,5).`

(c) `abs(A,-5).`

(d) `listmax([-1,-2],M).`

(e) `listmax([A,B],10).`

(f) `listmax([A,B],M), abs(M,10).`
(This one is harder ... you should give all of the answers that CLP($\mathcal{R}$) would, but we won't take off any points if you give the answers in a different order than CLP($\mathcal{R}$) would.)

5. (5 points) Consider the following Scheme definitions.

```
(define n 100)
(define (test n)
  (helper n))
(define (helper k)
  (+ n k))
```

(a) What is the result of evaluating `(test 5)`?

(b) Suppose that Scheme used dynamic rather than lexical scoping. In that case what would be the result of evaluating `(test 5)`?

6. (5 points) In Miranda there is a single numeric type `num`, and both 3 and 3.5 are of type `num`. In Haskell, another pure functional language, there are separate types `int` and `float`. What are the advantages and disadvantages of the Miranda design?

7. (8 points) Consider the following Java class definitions. (These compile correctly.) What is the result of compiling and executing the following Java program (using the `main` method in `Test`)?

```
class Animal {

  public void printIt() {
    System.out.println("printIt method in Animal");
  }

  static public void staticPrintIt() {
    System.out.println("StaticPrintIt method in Animal");
  }

}


class Bird extends Animal {

  public void printIt() {
    System.out.println("printIt method in Bird");
    super.printDescription();
  }

  static public void staticPrintIt() {
    System.out.println("StaticPrintIt method in Bird");
  }

}


public class Test {
  public static void main (String [ ] args) {
    Animal a = new Animal();
    Bird b = new Bird();
    Animal c = new Bird();
    a.printIt();
    b.printIt();
    c.printIt();
    Animal.staticPrintIt();
    Bird.staticPrintIt();
  }
}
```

8. (32 points) Tacky but easy-to-grade true/false questions!

   (a) In Miranda, programs would give the same answers if we replaced lazy evaluation with call-by-name.

   (b) In Miranda, programs would give the same answers if we replaced lazy evaluation with call-by-value.

   (c) Control structures are implemented in Smalltalk using message sending; there is no special syntax for such things as `for` and `while` loops beyond blocks and message sending.

   (d) Smalltalk includes static type declarations that are similar to those in Java.

   (e) Scheme, Java, and Miranda all support higher-order functions.

   (f) As soon as a Java object can no longer be accessed from any active objects, its `finalize()` method will be invoked and it will be garbage collected.

   (g) After a programmer is done using a cyclic structure in Java, it is essential that the programmer explicitly break the cycle by setting one or more fields of the objects in the cycle to null. Otherwise the structure can never be garbage collected, even if it can no longer be accessed from any active variables, since the reference counts of the objects will never drop to zero.

   (h) A Java class can extend only one other class, but can implement many interfaces.

   (i) The methods in an inner class in Java have access both to fields of the inner class and to fields of the enclosing class.

   (j) A Java program can have multiple threads, but only one thread can be running at any given time.

   (k) A static method in Java may not use the `this` reference.

   (l) Two different threads cannot simultaneously execute the same `synchronized` method in Java.

   (m) In Scheme parameters are passed by storing a copy of the actual argument in the variable for the formal parameter. This is true for parameters of all kinds, including integers and lists.

   (n) An interactive Java program must check for mouse clicks and other input events using a continuously-executing loop that repeatedly checks the state of the mouse to see if a button has been pushed.

   (o) Evaluating a tail recursive function in Scheme requires stack space proportional to the number of recursive calls.

   (p) A variable of type `Object` can store a reference to any Java data type, including `String`, `int`, and `char`.