CSE 341 Final — Autumn 1998 — Answer Key

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? 1 1 2 2

   (b) What is the output if the parameters are passed by result? 1 100 101 100

   (c) What is the output if the parameters are passed by value-result? 1 100 101 100

   (d) What is the output if the parameters are passed by reference? 100 100 100 100

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?

   The principal change would be to replace all of the primitive types — boolean, char, int, and so forth — with real objects. The wrapper classes Boolean, Character, Integer, and so forth, could be replaced by classes for these new objects. One could extend these classes just as with any others. This would significantly improve language consistency — one could put any kind of datatype in a collection, for example. It would negatively affect efficiency, unless there were some quite clever compilation techniques, because the primitive types are handled more efficiently than real objects. (This is the only part of this answer required for full credit.)

   Some other changes might be to make operators the same as any other method, so that they can be used in user-defined methods, and to make classes first-class objects, rather than having a special reflection mechanism.

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 ::`
    `num -> num -> num`

(b) `plus 5 ::`
    `num -> num`

(c) `append ::`
    `[*] -> [*] -> [*]`

(d) `append [] ::`
    `[*] -> [*]`

(e) `append [3,4] ::`
    `[num] -> [num]`

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

(g) `my_map2 ::`
    `(* -> ** -> ***) -> [*] -> [**] -> [***]`

(h) `my_map2 plus ::`
    `[num] -> [num] -> [num]`

(i) `my_map2 append ::`
    `[[*]] -> [[*]] -> [[*]]`

(j) `my_map2 plus [1..] [1..]`
    `[2,4,6,8,10, ...`

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).`
    `A=5`

(b) `abs(A,5).`
    `A=5;`
    `A=-5`

(c) `abs(A,-5).`
    `no`

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

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

(f) `listmax([A,B],M), abs(M,10).`
```
M=10, A=10, B<=10
M=-10, A=-10, B<=-10
M=10, A<10, B=10
M=-10, A<-10, B=-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)`?
105

(b) Suppose that Scheme used dynamic rather than lexical scoping. In that case what would be the result of evaluating `(test 5)`?
10

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?

The advantage of the Miranda design is that it is simpler — there is just one type, and as a result the arithmetic operators such as + and * are not overloaded. In fact there is no overloading in Miranda, which significantly simplifies the type system. (Miranda actually cheats a bit here, in that the "show" operator is overloaded, but it's built-in and not user definable. Also "=" is not defined for function types, but this is checked at runtime. We aren't expecting you to include this though in your answer.)

The disadvantage is that the type system does not differentiate between integers and floats. As a result, we lose some of the benefits of static type checking for functions that require integers or floats specifically. For example, the `mod` function should only be applied to integers, but since in Miranda it just has type `num -> num -> num` the check to make sure its arguments are integers rather than floats is only done at run time.

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.printIt();
  }

  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();
  }
}
```

```
  printIt method in Animal
  printIt method in Bird
  printIt method in Animal
  printIt method in Bird
  printIt method in Animal
  StaticPrintIt method in Animal
  StaticPrintIt method in Bird
```

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. True.

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

(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. True.

(d) Smalltalk includes static type declarations that are similar to those in Java. False. (There aren't any type declarations in Smalltalk.)

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

(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. False.

(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. False.

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

(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. True.

(j) A Java program can have multiple threads, but only one thread can be running at any given time. False. (Consider multi-processor machines.)

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

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

(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. False. (I was assuming that this meant that the list was copied, but some people interpreted this as meaning the pointer was copied — so we're throwing this question out.)

(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. False.

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

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