CSE 341 Final — June 13, 2002

Your name:

This exam is closed book and notes. 10 points per question — 130 points total.

1. Consider static vs. dynamic typing. Discuss one advantage of each.

2. Suppose that we had a version of Miranda, called V-Miranda, that used **call by value** rather than lazy evaluation. Would V-Miranda offer **referential transparency**? Answer "yes," "no," or "more information needed," and **explain why**.

3. Consider the following program in an Algol-like language.

```
begin
integer n;
procedure p(k: integer);
    begin
    k := k+10;
    n := n+k;
    print(n,k);
    end;
n := 20;
p(n);
print(n);
end;
```

(a) What is the output when k is passed by **value**?

(b) What is the output when k is passed by **value-result**?

(c) What is the output when k is passed by **reference**?

4. What does the following Java program print?

```
import java.awt.Point;
class Test {
    public static void main(String[] args) {
        Point p = new Point(10, 20);
        System.out.println("before resetOne: p.x = " + p.x);
        resetOne(p);
        System.out.println("after resetOne: p.x = " + p.x);

        p = new Point(10, 20);
        System.out.println("before resetTwo: p.x = " + p.x);
        resetTwo(p);
        System.out.println("after resetTwo: p.x = " + p.x);
    }

    public static void resetOne(Point q) {
        q.x = 0;
        q.y = 0;
    }

    public static void resetTwo(Point q) {
        q = new Point(0,0);
    }
}
```

(There is more space for your answer on the next page.)

5. Control structures in Java and Smalltalk.

   (a) Compare how **conditionals** are handled in Java and Smalltalk.

   (b) Compare how **iterating through a collection** is handled in Java and Smalltalk.

6. What is the principal problem with Java's type system that Pizza is designed to solve? Include an example in your description that illustrates the problem as it exists in Java, and that shows how it is solved by Pizza. (You don't need to write down code — just describe the example in English.)

7. Discuss the primary reason **inner classes** are useful for defining iterators in Java.

8. Consider the following Smalltalk class definitions of three classes.

```
Object subclass: #ClassOne
  instanceVariableNames: ''
  classVariableNames: ''
  poolDictionaries: ''

test
  Transcript show: 'test - ClassOne'.
```

```
ClassOne subclass: #ClassTwo
  instanceVariableNames: ''
  classVariableNames: ''
  poolDictionaries: ''

test
  Transcript show: 'test - ClassTwo'.
  super test.
  self snark.

snark
  Transcript show: 'snark - ClassTwo'.
```

```
ClassTwo subclass: #ClassThree
  instanceVariableNames: ''
  classVariableNames: ''
  poolDictionaries: ''

test
  Transcript show: 'test - ClassThree'.
  super test.

snark
  Transcript show: 'snark - ClassThree'.
```

What is printed when each of the following expressions is evaluated?

(a) `ClassOne new test`

(b) `ClassTwo new test`

(c) `ClassThree new test`

9. Define a **recursive** Scheme function `double-list` that takes a list of numbers as an argument, and returns a new list of numbers, each twice the number in the original. For example, `(double-list '(1 2 3))` should return `(2 4 6)`. (This version should **NOT** use `map`.)

10. Define another version of `double-list` that is **NOT** recursive, using `map` and `lambda`.

11. Suppose we evaluate the following Scheme expressions:

```
(define x '(1 2))
(define y '(10 11 12))
(define z (append x y))
(define a (cdr x))
(define b (cdr z))
```

Draw a box-and-arrow diagram of the lists that `x`, `y`, `z`, `a`, and `b` are bound to, being careful that your diagram clearly shows what parts of the lists are shared, if any.

12. Consider the following function definitions in Scheme.

```
(define x 2)
(define y 3)

(define (octopus x)
  (+ x y))

(define (squid x y)
   (octopus 10))

(define (mollusc x)
   (lambda (y) (+ x y)))

(define (crab z)
   ((mollusc 10) z))

(define (complicated x)
   ((mollusc 10) x))
```

What do the following expressions evaluate to? (They all evaluate without error.)

(a) `(+ x y)`

(b) `(octopus 20)`

(c) `(squid 100 200)`

(d) `(crab 20)`

(e) `(complicated 20)`

13. True or False? (Circle one for each.)

(a) True or False: In both Smalltalk and Java, 3 is an instance of a class.

(b) True or False: Scheme, Java, and Smalltalk all pass parameters **by reference**.

(c) True or False: It is **NOT** possible to pass a function as a parameter in Scheme, because Scheme will evaluate the function **before** it is passed.

(d) True or False: In Smalltalk, the programmer **can** modify methods of the built-in classes that come with the system.

(e) True or False: In Java, the programmer **can** modify methods of the built-in classes that come with the system.