

CSE 341 - Programming Languages

Final exam - Winter 2010

Your Name:

(for recording grades):

Total (max 106):

1. (max 10)

2. (max 5)

3. (max 20)

4. (max 10)

5. (max 10)

6. (max 8)

7. (max 6)

8. (max 10)

9. (max 8)

10. (max 5)

11. (max 4)

12. (max 10)

Open book and notes. No laptop computers, PDAs, or similar devices. (Calculators are OK, although you won't need one.) Please answer the problems on the exam paper — if you need extra space use the back of a page.

1. (10 points) Write a Haskell function `until_false` that takes a function `f` and a list `xs`. It returns a list of the elements of `xs` for which `f` evaluates to true — but as soon as `f` evaluates to false, it stops and doesn't examine any more elements. Here are some examples:

```
until_false even [2,4,6,7,10,12] => [2,4,6]
until_false even [2..] => [2]
until_false even [] => []
```

2. (5 points) What is the most general possible type for your `until_false` function from Question 1?

3. (20 points) The original Scheme metacircular interpreter doesn't include `or`.

(a) Write a Scheme function `or->combination` that implements `or` as a derived expression. Assume that the argument to `or->combination` is a legal `or` expression.

(b) Using your definition, what is the value of the following expressions?

i. `(or->combination '(or))`

ii. `(or->combination '(or (f 10)))`

iii. `(or->combination '(or (= x 5) (<y 10) (member x xs)))`

Hints: remember that for derived expressions, the Scheme interpreter takes a list representing the expression in one form, in this case as a list `(or ...)`, and returns another list representing an equivalent expression that either doesn't use `or`, or that uses a simpler `or` expression (i.e., a recursive case). You can use `if` or `cond` in your equivalent expression. Also remember the semantics of `or`: expressions are evaluated left to right, and the value of the first expression that evaluates to something other than `#f` is returned. Any remaining expressions are not evaluated. If all the expressions evaluate to `#f`, return `#f`. If there are no expressions, return `#f`. (Never evaluate an expression more than once, of course.)

If you need a brand-new symbol, you can generate one using `(gensym)`.

4. (10 points)

- (a) Write one or more rules to define the `positives` constraint in CLP(R). `positives` takes two arguments. It should succeed if both arguments are lists of numbers, and the second argument contains all the positive numbers in the first list (in the same order as in the first list), and only positive numbers. Don't use cut in your rules. For example, these goals should succeed:

```
positives([4,50,-1,-10,2], [4,50,2]).  
positives([], []).
```

and these should fail:

```
positives([4,50,-1,-10,2], [4,50]).  
positives([4,50,-1,2], [4,50,-1]).  
positives([], [10]).
```

- (b) What are all of the answers returned for the following goals, using your rules? (If there are infinitely many, give the first 3.)

- `positives([3,-4,0], B)`.

- `positives([3,10], B)`.

- `positives(A, [])`.

5. (10 points) Consider the `member` rule in CLP(R):

```
member(X, [X|Xs]) .  
member(X, [_|Xs]) :- member(X, Xs) .
```

Draw the simplified derivation trees for the following goals. If the tree is infinite, say that, and include at least the first 3 answers.

- (a) `member(A, [2, 3])` .
- (b) `member(2, As)` .

Continue on the back of this page as needed.

6. (8 points) Consider a version of the CLP(R) `member` rule with a cut:

```
member_cut(X, [X|Xs]) :- !.  
member_cut(X, [_|Xs]) :- member_cut(X, Xs).
```

What are all of the answers returned for the following goals? If there are an infinite number of answers, give the first three.

```
?- member_cut(10, [1, 2, 3]).
```

```
?- member_cut(A, [1, 2, 3]).
```

```
?- member_cut(A, []).
```

```
?- member_cut(10, As).
```

7. (6 points) Consider the following example in an Algol-like language.

```
begin  
  integer n;  
  procedure p(j: integer);  
    begin  
      j = j+n;  
      n := 2*n+j;  
      print(n);  
      print(j);  
    end;  
  n := 10;  
  p(n);  
  print(n);  
end;
```

- (a) What is the output when `j` is passed by value?
- (b) What is the output when `j` is passed by value result?
- (c) What is the output when `j` is passed by reference?

8. (10 points) In Ruby, `true` and `false` are objects: `true` is an instance of `TrueClass`, and `false` is an instance of `FalseClass`. `and`, `or`, and `!` (not) are built-in, but they could actually be user-defined methods. Write methods for `and`, `and_sc`, `or`, `or_sc`, and `not` for both `TrueClass` and `FalseClass` (so 10 methods in all). `and` should always evaluate its argument, while `and_sc` should do short-circuit evaluation (so it should take a block rather than a regular argument); and similarly for `or` and `or_sc`. Here some example expressions and the result of evaluating them:

```
(1==2).not => true
(1==2).and(1/0==1) => ZeroDivisionError exception
(1==2).and_sc {1/0==1} => false
```

9. (8 points) Consider the following Java programs Test1, Test2, Test3, and Test4. In each case, does the program compile correctly? If so, does it execute without error, or is there an exception?

```
/****** Test1 *****/
import java.awt.Point;
class Test1 {
    public static void main(String[] args) {
        Point[] a;
        a = new Point[10];
        a[0] = new Point(10,20);
        test(a);
    }
    public static void test(Object[] b) {
        b[1] = "clam";
    }
}
/*******/
```

```
/****** Test2 *****/
import java.awt.Point;
import java.util.LinkedList;
class Test2 {
    public static void main(String[] args) {
        LinkedList<Point> a = new LinkedList<Point>();
        a.add(new Point(10,20));
        test(a);
    }

    public static void test(LinkedList<?> b) {
        b.add(new Point(20,30));
    }
}
/*******/
```

```
/****** Test3 *****/
import java.awt.Point;
import java.util.LinkedList;
class Test3 {
    public static void main(String[] args) {
        LinkedList<Point> a = new LinkedList<Point>();
        a.add(new Point(10,20));
        test(a);
    }

    public static void test(LinkedList<?> b) {
        b.add(null);
    }
}
/*******/
```

```

/***** Test4 *****/
import java.awt.Point;
import java.util.LinkedList;
class Test4 {
    public static void main(String[] args) {
        LinkedList<Point> a = new LinkedList<Point>();
        a.add(new Point(10,20));
        test(a);
    }

    public static void test(LinkedList<Object> b) {
        b.add(new Point(30,40));
    }
}
/*****/

```

10. (5 points) What is the result of evaluating the following Ruby expressions? (Hint: String is a subclass of Object.)

- (a) "squid".class
- (b) "squid".class.class
- (c) "squid".class.class.class
- (d) "squid".class.superclass
- (e) "squid".class.superclass.superclass

11. (4 points) Consider the following Scheme program.

```

(define n 10)
(define (squid n)
  (clam n))
(define (clam k)
  (+ n k))

```

- (a) What is the result of evaluating (squid 0)?
- (b) Suppose that Scheme used dynamic rather than lexical scoping. In that case what would be the result of evaluating (squid 0)?

12. (10 points) Write a count method for the Enumerable mixin in Ruby. The count method should take any object as a parameter, and return the number of occurrences of that object in the collection. (What counts as an occurrence should be determined by the == test). For example, ["clam", "squid", "clam"].count("clam") should evaluate to 2.