Programming Languages
Final Exam

Please write all answers in the blue book. Keep your answers brief!

1. (a) Define the following terms: imperative language, functional language, object oriented language.

   (b) Give an example of a term in the lambda calculus that will be reduced to a normal form in fewer steps (applications of β-reduction) using applicative order evaluation than using normal order evaluation. Show the reduction sequences using applicative order evaluation and using normal order evaluation (should be short!).

   (c) What would the following program print if it were written in a language that used pass-by-name parameter passing?

```
program exam;
  i: integer := 0;
  function f(x: integer):Integer
     sum: integer := 0;
  begin
     while (i < 10) do
       sum := sum + x;
       i : = i + 1;
     end while;
     return sum;
  end f;
begin (* program *)
  print(f(i+3));
end exam;
```

2. (a) Write a Scheme function `maxlist` that finds the largest number stored anywhere in a list. Assume the list can contain sublists, but the only atomic elements will be numbers. For example, `(maxlist '(1 (9 2 (10 5) 6) 8))` should return 10.

   (b) In your Scheme interpreter, why wasn't `apply` defined in your library file, but instead was inserted into `*global-env*`? [NOT RELEVANT TO SPRING 2015 CLASS]

   (c) Write a Scheme function `f` that, given a list `L`, returns a function that, given a list `L2`, returns `L` if `(maxlist L)` is greater than `(maxlist L2)`, and returns `L2` otherwise. For example,

```
> (define g (f '(1 2 (3 4))))
> (g '(1 3 5))
(1 3 5)
> (g '(1 (2 3)))
(1 2 (3 4))
```

3. (a) Define a function in ML whose type is `('a -> 'b) -> ('a -> 'b)`.

   (b) What is the type of the following function?

```
fun foo f g x y = let fun bar z = f(g(x,y), z)
                  in bar 3
                  end
```

   (c) Give an intuitive description of how the type you gave in your previous answer would be inferred by the compiler.

(d) Define in ML a polymorphic datatype, `'a mylist`, whose values are either an empty list, written `nil`, or a non-empty list, created by writing an expression of the form `cons(x,xs)`, where x is of type `'a` and `xs` is of type `'a mylist`. For example, once you have defined the `mylist` datatype, you could write,

```
val L = cons(3, cons(4, nil))
```

(e) Write a polymorphic function `max` whose type is `('a * 'a -> bool) -> 'a mylist -> 'a` that computes the maximum element of an `'a mylist`, where the first parameter is a "greater than" operator (and should be used as `>` within `max`). Assume that `max` will not be called on the empty list (so you don't have to handle that case).

4. (a) Give a simple example in Java of the overriding of a method (in a child class), as well as the dynamic dispatching of that method.

   (b) Why doesn't Java allow subtyping between instances of a generic class? Give an example in Java where such subtyping, if it were allowed, would cause a problem. Briefly explain your example.

   (c) Suppose there is a Java generic class, `C<T>`, that implements the `Collection<T>` interface, so that the method `boolean add(T x)` adds a `T` object to the collection. Write a static method `insert()` that is as polymorphic as possible and takes two parameters:

   - a `Car` object (where `Car` is derived from `Vehicle`), and
   - any `C<>` object into which a `Car` object can be inserted.

   Your `insert()` method should add the `Car` object to the `C<>` object using the `add()` method.

   (d) Suppose class `Porsche` is derived from `Car`. Given your definition of `insert()`, and given a variable `c` of type `C<Car>` and a variable `p` of type `Porshe`, would it be possible to call `insert(c,p)`? Explain why or why not.

5. (a) Explain how function subtyping in Scala, where if `B<:A` then `A=>B <:B=>A`, satisfies the subset interpretation of subtyping.

   (b) Explain what the following code means:

   ```
   trait myTrait[T <: Ordered[T]] {
     def value: T
     def foo(other: myTrait[T]):Boolean
   }
   ```

   where `Ordered[]` is a trait in Scala supporting the comparison operators.

   (c) Define two unrelated Scala generic classes, `myClass[T]` and `yourClass[T]`, which both implement the `myTrait[T]` trait. The `foo()` method in both classes should use both the "this" object and the "other" object to compute its result.

   (d) Can the `foo()` method of `myClass[T]` be passed an object of type `yourClass[T]` and vice versa? Explain.

6. (a) What is the advantage of using a heap pointer to allocate heap structures rather than a free list?

   (b) Why is copying GC used in conjunction with a heap pointer and mark/sweep garbage GC generally used in conjunction with a free list?

   (c) In a scenario in which the heap is large, but the number of live objects (and the amount of space occupied by those objects) is small, which of mark/sweep or copying garbage collection would you expect to consume less time? Explain?