Programming Languages
CSCI-GA.2110.001 Fall 2014

Homework 2
Due Thursday, December 11

You should write the answers using word, latex, etc., and upload them as a PDF document. **Important: You must turn this in by 11:55pm on Thursday, December 11. I will be posting the solutions prior to the review session on December 12.**

1. Write the following in Scheme (you don't have to implement it on a computer):

   (a) The function `(listfromTo a b)` which, given two integers `a` and `b`, returns a list of all the numbers between `a` and `b`, inclusive.

   (b) The function `(removeMult a L)` which removes all multiples of the integer `a` from the list `L` of integers. You can use the built-in function `(modulo x y)` which returns the remainder of `x/y`.

   (c) The function `(sieve n)` which returns the list of all prime numbers less than or equal to `n` using the Sieve of Eratosthenes. The algorithm starts with the list of numbers from 2 to `n` and, starting from the beginning of the list, for each element of the list encountered, it removes all subsequent multiples of that element from the list (before progressing to the next element). Be sure to write a purely functional version of it.

2. Suppose you have written a Scheme interpreter where the main evaluation function is called `my-eval`.

   (a) Suppose the following new construct was added to Scheme, which your interpeter had to interpret:

   `(longest (`exp_1 $result_1$`) ... (`exp_N $result_N$`))`

   The `longest` construct has to evaluate each of exp_1 through exp_N, which should each return a list. If expression exp_i returns the longest list (as determined by the usual `length` function), then the value of $result_i$ is returned (and none of the other *result* expressions are evaluated). For example,

   ```
   (longest ('(1 2 3) (+ 1 2 3))
            ((cdr '(1 2 3 4 5)) 'yes)
            ((cons 8 (list 9 10)) 'no) )
   ```

   would return the symbol `yes`, because `(cdr '(1 2 3 4 5))` is a longer list than `(1 2 3)` or the result of `(cons 8 (list 9 10))`. Write a function `handle-longest`, which you could insert into your interpreter, which takes an entire expression `(longest ....)` and an environment and interprets the expression.

   (b) Explain why, in your Scheme interpreter, you were able to use `(let ...)` in your code that handled `(let ..)` (e.g. in your `handle-let` function) and were able to use `(cond ...)` in your `handle-cond` function. Even if you didn't, explain why you could have. In other words, explain why, if your interpreter needed to implement some construct (such as `let` and `cond`), you were still able to use that construct in your interpreter.

3. (a) In ML, why do all lists have to be homogeneous (i.e. all elements of a list must be of the same type)?
   (b) Write a function in ML whose type is `('a -> 'b) -> ('b -> 'c) -> 'a -> 'c`.
   (c) What is the type of the following function (try to answer without running the ML system)?

```
fun foo f (op >) (x,y) z =
   let fun bar a = if x > y then z else a
   in  bar [1,2,3]
   end
```

   (d) Provide an intuitive explanation of how the ML type inferencer would infer the type that you gave as the answer to the previous question.

4. Consider the following package specification for an Ada package that implements a queue of integers.

```
package queue is
  function extract return integer;
  function insert(x: integer);
end queue;
```

   (a) Why would this package not be said to implement an abstract data type (ADT) for a queue?
   (b) Modify the above package specification, and implement a simple package body (that performs no error checking), so that a queue is an ADT.

5. (a) As discussed in class, what are the three features that a language must have in order to considered object oriented?
   (b) i. What is the "subset interpretation of suptyping"?
       ii. Provide an intuitive answer, and give an example, showing why class derivation in Java satisfies the subset interpretation of subtyping.
       iii. Provide an intuitive answer, and give an example, showing why subtyping of functions in Scala satisfies the subset interpretation of subtyping.
   (c) Consider the following Scala definition of a tree type, where each node contains a value.

```
abstract class Tree[T <: Ordered[T]]
case class Node[T <: Ordered[T]](v:T, l:Tree, r:Tree) extends Tree[T]
case class Leaf[T <: Ordered[T]](v:T) extends Tree[T]
```

   Ordered is a built-in trait in Scala (see `http://www.scala-lang.org/api/current/index.html#scala.math.Ordered`). Write a Scala function that takes a Tree[T], for any ordered T, and returns the maximum value in the tree. Be sure to use good Scala programming style.
   (d) In Java generics, subtyping on instances of generic classes is invariant. That is, two different instances `C<A>` and `C<B>` of a generic class `C` have no subtyping relationship, regardless of a subtyping relationship between `A` and `B` (unless, of course, A and B are the same class).

i. Write a function (method) in Java that illustrates why, even if `B` is a subtype of `A`, `C<B>` should not be a subtype of `C<A>`. That is, write some Java code that, if the compiler allowed such covariant subtyping among instances of a generic class, would result in a run-time type error.

ii. Modify the code you wrote for the above question that illustrates how Java allows a form of polymorphism among instances of generic classes, without allowing subtyping. That is, make the function you wrote above be able to be called with many different instances of a generic class.

(e) i. In Scala, write a generic class definition that supports covariant subtyping among instances of the class. For example, define a generic class C[E] such that if class B is a subtype of class A, then C[B] is a subtype of C[A].

ii. Give an example of the use of your generic class.

(f) i. In Scala, write a generic class definition that supports contravariant subtyping among instances of the class. For example, define a generic class C[E] such that if class B is a subtype of class A, then C[A] is a subtype of C[B].

ii. Give an example of the use of your generic class.

6. (a) What is the advantage of a mark-and-sweep garbage collector over a reference counting collector?

(b) What is the advantage of a copying garbage collector over a mark and sweep garbage collector?

(c) Write a brief description of generational copying garbage collection.

(d) Write, in the language of your choice, the procedure `delete(x)` in a reference counting GC system, where `x` is a pointer to a structure (e.g. object, struct, etc.) and `delete(x)` reclaims the structure that `x` points to. Assume that there is a free list of available blocks and `addToFreeList(x)` puts the structure that `x` points to onto the free list.