Homework 7

Due Friday, 10/22/10

Please turn in your homework solutions online at `http://www.dci.pomona.edu/tools-bin/cs131upload.php` before the beginning of class on the due date.

1. (10 points) **Let expressions**

   Suppose we extend the grammar for typed PCF (see lecture 19) to include "let" expressions:

   ```
   e::= let x: T = e in e end
   ```

   Thus we could write

   ```
   let x: Int = 17 in succ(succ x) end
   ```

   (a) Please write down the type-checking rule for the let expression in the same form as Lecture 19. That is, the bottom line (below the long horizontal line) should read

   ```
   E |- let x:T = e1 in e2 end : U
   ```

   Write down the complete rule including all of the hypotheses necessary to prove the typeing of the let expression.

   (b) Please write down the evaluation rule for the new let expression. As in Lecture 19, the bottom line should read:

   ```
   (let x:T = e1 in e2 end, env) => v
   ```

   Again, write down the entire rule, including the hypotheses that generate that conclusion.

2. (20 points) **Scala Programming**

   Be sure to have a computer with Scala installed (all Macs in the Pomona CS lab have Scala installed with Eclipse). If you use Scala with Eclipse (my recommendation) then go to `http://www.scala-ide.org/` andfollow the instructions on how to download and install the right version of Eclipse and how to download and install Scala from within Eclipse. Then go to user documentation and then the "Tutorial Example" for information on how to write a program in Scala using Eclipse. Do read and follow the directions carefully. If you omit steps then Eclipse will not be able to find your `main` method.

   If you would rather, you can just edit your program using emacs and run scalac to compile your code using scalac. See the on-line tutorial for further information on running programs and to find more info on the language.

   (a) The scala library class `List`, in `scala.collection.immutable`, provides immutable lists. You can create a list of numbers just by writing `List(1,2,3,4)`. Notice that "new" is not required (though normally it would be to create objects). You can find more information about the class and its operations by going to the general library documentation at

`http://www.scala-lang.org/api/current/index.html` and then selecting `scala.collection.immutable` from the packages on the left and then clicking on `List` beneath it.

The element `Nil` is a built in object, while "::" can be used to create a new list by adding a new first element. For example, `5::List(6,7,8,9)` results in a new list containing the numbers from 5 to 9.

Please write a function that takes a list as input and reverses it. Write a main method that tests it.

(b) Look up information about methods `map` and `foreach`. Both apply a function to all the arguments of the list, but `map` takes the results and puts them in a list.

Use `foreach` to write a method `printall(lst:List[Int])` that prints out all of the elements in a list of integers. Write a sample program that creates a list of integers and calls `printall` to print out all of its values.

(c) Do problem 4a from homework 4. It asks for you to code up a list comprehension operator. You may use the map and filter functions from the library class `List` from package `scala.collection.immutable`.

(d) Write a binary search tree class over integers. [Don't bother to write a parameterized class, just have it use integers.] Write methods that will allow you to add new elements to the tree, compute the size of the tree, and to traverse the tree inorder, applying a function f to all elements of the tree and gathering the results in a list. That is the declaration of this method will look like:

```
def inorder(f:int => int):List[Int] {...}
```

If an in-order traversal of the tree would produce $[a_1, \ldots, a_n]$ then this method should provide $[f(a_1), \ldots, f(a_n)]$. Write a program that tests this on a few simple trees and some simple functions.