Homework 8

Due Thursday, 11/6/2014

Please turn in your homework solutions online at `http://www.dci.pomona.edu/tools-bin/cs131upload.php`.

1. (10 points) **Activation records**

   Draw the stack of activation records for the following Ada program (a) after the first call to procedure b; (b) after the second call to procedure b. Show the static (access) and dynamic (control) links for each activation record. (c) Indicate how x is found in procedure c.

```
procedure env is
   integer x = 12;
   procedure a is
      integer y = 2;
      procedure b(z) is
         procedure c is
         begin
            b(x);
         end c;
      begin
         c;
      end b
   begin
      b(x+y);
   end a;
begin
   a;
end env
```

2. (15 points) **Function Calls and Memory Management**

   Please do problem 7.12 from Mitchell, page 198.

3. (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). The best source of information on Scala is `http://www.scala-lang.org/` Information on how to install Scala on your own computer with Eclipse is available at `http://scala-ide.org`.

   If you wish to run your programs through Eclipse, see the documentation at `http://scala-ide.org/docs/index.html`. Do read and follow the directions carefully. If you omit steps then Eclipse will not be able to find your `main` method. The older eclipse implementations were a bit buggy, but the new ones seem better. If you wish, you can just run it on the command line. In that case you can just edit your program using a text editor and run scalac to compile your code using scalac.

(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`, 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. (Do NOT use the built-in reverse method! Write it using the primitives mentioned above.) 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 5a 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.

4. (10 points) **Exceptions** (I understand that some of you don't know ML, but the programs below behave identically to Haskell – aside from exceptions. Let me know if you have any problems understanding the ML code below.)

Consider the following functions, written in ML:

```
exception Excpt of int;
fun twice(f,x) = f(f(x)) handle Excpt(x) => x;
fun pred(x) = if x = 0 then raise Excpt(x) else x-1;
fun dumb(x) = raise Excpt(x);
fun smart(x) = (1 + pred(x)) handle Excpt(x) => 1;
```

What is the result of evaluating each of the following expressions?

(a) twice(pred,1);

(b) twice(dumb,1);

(c) twice(smart,0);

In each case, be sure to describe which exception gets raised and where.