COMP 302: Assignment 2, Summer 2008.
SML datatypes, closures

Due Date: June 6th in my office (McConnell 106) or on June 5th in class.

Guidelines for submission: For this assignment, please print out a few test cases to demonstrate the correctness of your work.

Question 1: (20 points) Here is a solution to question 3(b) of the first assignment:

```
fun mult(X,Y) =
    let
       fun timesten(L1) = 0::L1;
       fun mult1h(nil, y:int, 0) = nil
          |mult1h(nil, y:int, rem) = rem::nil
          |mult1h(x::xt, y:int, rem) =
            ((x*y + rem) mod 10)::mult1h(xt, y,(x*y+rem) div 10);
       fun mult1(X, y) = mult1h(X, y, 0);
    in
       case Y of
             y::nil => mult1(X, y)
            |y::yt => add(mult1(X,y),mult(timesten(X),yt))
    end;
```

Assuming that `mult1h` correctly computes the product of an an n-digit bignum `X` with a one-digit number `y`, and assuming that `timesten` correctly multiplies a bignum by ten, prove the correctness of mult.

Note: This should be taken as an exercise in formal justification. Your goal is to explain formally to the reader why the algorithm is correct.

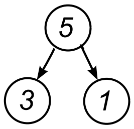
Question 2: (20 points) Construct a function which takes as input two input functions `f:real->real` and `g:real->real`, and outputs three functions. The first function, call it `f'`, will be such that `f'(x) = f(x)` for all inputs `x`. The second function `g'(x)`, will likewise satisfy `g'(x) = g(x)`. The third function `log:unit -> string` will output a string of `f` and `g` characters representing a log of the calls to `f'` and `g'`. The ith letter of the string should represent the function called on the ith application of `f'` and `g'`. What is output of log after evaluating the expression `f'(f'(3)) + g'(f'(g'(3) + f'(4)))`?

Question 3: (20 points) We will use the following declaration for trees:

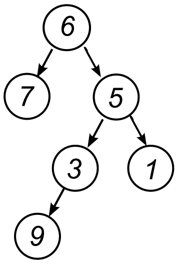
```
datatype 'a tree = Empty | Node of 'a tree * 'a * 'a tree
```

Let T_1 and T_2 be rooted trees labeled with ints. We will say that T_1 occurs within T_2 if there exists a connected subset S of vertices of T such that the rooted tree induced by S is isomorphic to T_1. For example we would say tree (a) occurs within tree (b) in the diagram below:

(a)



(b)



You will construct a function `isin` which takes an `int tree` T_1 as input, and returns a second function which takes as input an `int tree` T_2, and returns `true` if T_1 occurs within T_2 and `false` otherwise. The function `isin` should have type `(int tree)->(int tree)->bool`. Furthermore, the function should be written so that the code of the second function is optimized for fixed T_1.

Question 4: (40 points) The next questions will be about Huffman encodings. The wikipedia page is a good reference for this material.

a) Suppose that we have read a collection of text files and we have counted the number of occurrences of each character in an `(char*int) list`. Write a function `huffman` which takes as input this list and constructs a tree representing the Huffman encoding for these frequency measurements. This tree should be of the following type:

```
datatype htree = Leaf of int * char |
                 Node of htree * int * htree;
```

If a tree is a leaf, then the `int * char` pair is meant to encode the number of times that a character occurs within the sample text file. The value at an interior node should be the sum of the values at the leaf nodes.

b) Construct a function `hcode` which takes as input a tree such as the one constructed above and outputs a function which maps characters to their binary encodings represented as strings. A simple strategy would be to collapse the tree into to a list and have the function find the letter's encoding in the list. In this question it will not be necessary to optimize the function. It is possible to construct a final function which generates the code in O(1) time, but it would require some features of SML that we haven't introduced yet.