CS 457/557 Homework 5 – due 10am, Tuesday, May 6, 2003

Hand in your solutions on paper and email them to cs457acc@cs.pdx.edu. All the programs should be placed in a single .hs file, which can be the body of your email message or an attachment. It is not necessary to show evidence that you have loaded and tested your programs, but this is of course the only sensible way to make sure that you have found correct answers!

1. Do Hudak Exercise 7.1. Hint: Give your function the type

   foldTree :: (a -> a -> a) -> (b -> a) -> Tree b -> a

2. Do Hudak Exercise 7.4, using InternalTree.

3. Do Hudak Exercise 7.5. You’ll want to extract and modify the existing code from Trees.lhs (rather than trying to import it). Hint: Define your revised version of evaluate in terms of an auxiliary function

   evaluate’ :: [(String,Float)] -> Expr -> Float

   where the first argument is a list of variable bindings to be used in evaluating the second argument. For example,

   evaluate’ [("x", 1.0),("y",2.0)] (V "x" :+: V "y")

   should yield 3.0. You may find the function Prelude.lookup to be useful.

4. Do Hudak Exercises 8.5 and 8.6. You’ll want to extract and modify the existing code from Region.lhs (rather than trying to import it). Assume that the list of vertices passed to polygon is in counter-clockwise order.

5. Just as a set containing elements of type a can be represented by a function of type

   a -> Bool

   so a dictionary (finite map) with keys of type k and values of type v can be represented by a function of type

   k -> Maybe v

   Suppose we want to define an abstract data type of such dictionaries. Complete the following implementation by giving definitions of find and insert. (Hint: Let the types be your guide!)

   type Dict a b = a -> Maybe b
   empty :: Dict a b
   empty a = Nothing
   find :: Dict a b -> a -> Maybe b
   insert :: Eq a => Dict a b -> a -> b -> Dict a b