CSE 305 Spring 2015

Assignment #5: Polymorphic Types and Functional Programming in ML

Assigned: April 14, 2015

(see **red** highlight for minor corrections)

Due: Mon, April 27, 2015 (11:59 pm)

(Assignment #6 will be given on April 24 and will be due on May 7.)

Note: This assignment may be done by a pair of students.

The need for polymorphic types in programming can be seen from the C program bfirst_A5.c posted on Piazza. This program requires two different monomorphic types for lists, one for a list of integers and another for a list of trees, in addition to two variations for each function operating on lists – icons, tcons, iaddback, and taddback. In this assignment, you are to create in ML a functional version of the above program, in order to demonstrate the conciseness of polymorphic typing as well as the power of type inference in providing static typing checking without declaring types for any variables.

Problem 1.

Assume the following ML type for binary trees (***** was accidentally omitted in initial posting):

```
datatype 'a tree = leaf | node of 'a * 'a tree * 'a tree;
```

Complete the definition below for `insert(i, tr)` which will insert a value `i` into `tr` so as to maintain the *binary search tree* property.

```
fun insert(i, leaf) =  __________________
  | insert(i, node(v,left,right)) =
                    if i = v then __________________________
                    else if i < v then _____________________
                         else ______________________________;
```

In the above code, ML will assume `v` to be of type `int` by default. In completing the definition of `insert`, note that you do not update the input tree; rather, you need to construct a new tree in which the value to be inserted is placed in the correct position. Test your definition by running the following function, making sure to place the code for `testcase` **after** `insert`:

```
fun testcase() =
        let val t1 = node(100,leaf,leaf);
            val t2 = insert(50, t1);
            val t3 = insert(150, t2);
            val t4 = insert(200, t3);
            val t5 = insert(125, t4);
            val t6 = insert(175, t5);
            val t7 = insert(250, t6);
            val t8 = insert(25, t7);
            val root = insert(75, t8)
         in root
        end;
```

Submit online a file **insert.sml** containing the type definition for `tree` as well as the code for `testcase and insert.`

Problem 2.
Define a function that returns a list of integers that we would get from a breadth-first traversal of a BST. Develop your answer by completing the definition below for `bfirst`.

```
fun bfirst(tr) =
        let fun bf([], ans) = ____________
              | bf(leaf::t, ans) = _____________________
              | bf(node(v,t1,t2)::t, ans) = __________________
         in bf(________,__________)
        end;
```

Note that function `bf` will be tail-recursive and the second parameter, `ans`, is used to accumulate the final answer. Test out `bfirst` by running the following function:

```
fun test_bfirst() = bfirst(testcase());
```

Submit online a file **bfirst.sml** containing the type definition for `tree` and the code for **`insert`**, `testcase` and `bfirst.`

Problem 3.
Consider the following depth-first ("in order") traversal of a BST.

```
fun dfirst(leaf) = []
  | dfirst(node(v,t1,t2)) = dfirst(t1) @ [v] @ dfirst(t2);
```

Following the strategy of `bfirst`, develop a tail-recursive version of `dfirst`, called `dfirst2`. That is, the function `dfirst2` should make use of a tail-recursive helper function, say `df` (analogous to `bf`), which will use an accumulator-passing style in order to construct the answer. Test out `dfirst2` by running the following function:

```
fun test_dfirst2() = dfirst2(testcase());
```

Submit online a file **dfirst.sml** containing the type definition for `tree` and the code for **`insert`**, `testcase and dfirst2.`

Team-work and On-line Submission:

1. This assignment may be done by a team of at most two students. Write both student names at the top of all program files when submitting them.

2. It is fine to do the assignment solo. Write your name at the top of each file and submit it.

End of Assignment #5