CSE 505 Spring 2017

Assignment 5: Functional Programming in ML

Assigned: Tues, April 18, 2017
(Correction in red, posted on April 21)
Due: Fri, April 28, 2017 (11:59 pm)
Note: This assignment may be done by a pair of students.

Problem 1. Consider the following ML type for representing integer binary search trees:

```
datatype bstree = leaf | node of int * bstree * bstree;
```

Write a function **insert(v, tr)** which will insert an integer value **v** into tree **tr** so as to maintain the binary search tree property. Test your program using the following code:

```
fun testcase1() = reduce(insert, leaf, [50,30,20,40,60]);

testcase1();
```

where **reduce(f, b, l)** is the standard higher-order function on lists. Lecture 18, slides 11-13 provides guidance on how **insert** should behave.

Problem 2. Below is a type definition for an n-ary tree, which generalizes a binary tree so that each internal node has a **list of zero of more subtrees** and each leaf node holds a value:

```
datatype ‘a ntree = leaf of ‘a  |  node of  ‘a ntree list;
```

a. Using the **map(f, l)** higher-order function, define a function **subst(tr,v1,v2)** which returns a new ntree in which all occurrences of **v1** in **tr** are replaced by **v2**. For example,

```
subst(node([leaf(“x”), node([leaf(“y”), leaf(“x”), leaf(“z”)])]), “x”, “w”) =
                              node([leaf(“w”), node([leaf(“y”), leaf(“w”), leaf(“z”)])])
```

b. Using the **reduce(f, b, l)** function, define a function **cat(tr)** which returns the concatenation of all strings at the leaf nodes of **tr**, adding a space between each value. For example,

```
cat(node([leaf(“x”),node([leaf(“y”),leaf(“x”),leaf(“z”)])])) = “x y x z”
```

Incorporate the above test cases as two functions:

```
fun test_subst() = subst(node([leaf(“x”), node([leaf(“y”), leaf(“x”), leaf(“z”)])]), “x”, “w”);

fun test_cat() = cat(node([leaf(“x”),node([leaf(“y”),leaf(“x”),leaf(“z”)])]));

test_subst();
```

```
test_cat();
```

Problem 3. Consider the following depth-first (“in order”) traversal of a binary search tree.

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

Write a tail-recursive version of dfirst, called **dfirst2**. Define **dfirst2** in terms of a helper (inner) function **df**: ‘a tree list * ‘a list → ‘a list, which uses an accumulator-passing style in order to construct the answer. Test **dfirst2** using the **testcase1**() function of problem 1:

```
fun test_dfirst2() = dfirst2(testcase1());

test_dfirst2();
```

Problem 4. Consider an infinite list of strings of the form:

```
[ “Lf.Lx.(f x)”,
  “Lf.Lx.(f (f x))”,
  “Lf.Lx.(f (f (f x)))”,
  “Lf.Lx.(f (f (f (f x))))”,   …  ]
```

These strings represent the numbers 1, 2, 3, 4 … in the pure λ-calculus. Here, L stands for λ. Each string is called a *Church numeral* – in honor of Alonzo Church who invented the λ-calculus.

Refer to the **infinite list** ML type discussed in Lecture 19:

```
datatype 'a inf_list = lcons of 'a * (unit -> 'a inf_list)
```

Define a function **church**: string -> string inf_list which generates an infinite list of Church numerals starting from 1. Test **church** by executing the following main program:

```
fun take(0, _) = []
  | take(n, lcons(h, thk)) = h :: take(n-1, thk());

take(5, church(“x”));
```

WHAT TO SUBMIT:

Prepare a file named *A5_UBITId1_UBITId2.sml* if the assignment is done by two students; otherwise, name it as *A5_UBITId.sml* if the assignment is done solo. (Order the *UBITId*’s in alphabetic order, in the former case.) In this file, place the definitions for all datatypes and functions in the following order:

bstree, insert, testcase1,
ntree, map, reduce, subst, cat, test_subst, test_cat,
dfirst2, test_dfirst2,
inf_list, church, take

Also include the code shown that invokes the various tester functions.

Submit file using the `submit_cse505` command. For more details regarding online submission, see

`Resources` → `Homeworks` → `Online_Submission_2017.pdf`.

End of Assignment #5

P.S. If you cut and paste code from this assignment into your SML file, the quotation marks might not come out correctly and this could cause an error from the SML compiler.