CSE505 – Spring 2018

Assignment 2 – Functional Programming

Due Date for **Problems 1, 2**: **Fri, March 9** (11:59 pm, online)

Due Date for **Problem 3: Tues, March 13** (11:59 pm, online)

Problem 3: In Lecture 12, we discussed game-trees which are, in general, infinite but we consider a finite version in this question. The following ML datatype defines an *n-ary tree* in which the branching factor can vary from one node to another:

```
datatype 'a ntree = leaf of 'a | node of 'a ntree list
```

Assume that you are given an *n-ary tree* in which the leaf nodes all contain an integer representing the strength of a position in the game-tree. Write two mutually-recursive functions

```
min: int ntree → int
max: int ntree → int
```

which together carry out a 'minimax' computation. That is, max(tree) will call min(subtree) for every subtree of tree and will take the maximum of all the values obtained; likewise, min(tree) will call max(subtree) for every subtree of tree and will take the minimum of all the values obtained. Thus, invocations of min and max will alternate as the recursion goes down the levels of the game-tree until a leaf node is reached. The outline of their definitions is given below.

```
fun min(leaf(x)) = __________
  | min(node(l)) =
            let fun m ___________
             in reduce(m, _____, _____)
            end
and
   max(leaf(x)) = ___________
  | max(node(l)) =
            let fun m__________
             in reduce(m, _____, _____)
            end;
```

Complete the definitions of min and max by filling in the blanks. Note: You are not allowed to alter the form of the solution; you should just fill in the blanks suitably.

Place your code in a file called minimax.sml and submit it using the submit_cse505 command. Starter code including test cases posted at Resources→Homeworks→minimax.sml.

Make a directory called A2_P3_UBITId if doing solo; or A2_P3_UBITId1_UBITId2 if doing as a pair (give UBITId's in alphabetic order). Put minimax.sml inside the directory, compress, and submit.

End of Problem 3

Problem 1: In Lecture 10, we discussed the ML datatype for a simple binary tree

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

and the function cat shown below for concatenating all strings in a string tree, with blanks separating consecutive strings.

```
fun  cat(leaf(s) = s
   | cat(node(t1, t2)) = cat(t1) ^ " " ^ cat(t2);
```

Write a tail-recursive equivalent of cat, called cat2, such that for all string trees t, cat(t) = cat2(t), i.e., they produce the same output given the same input.

Hint: Define cat2 in terms of a tail-recursive function cat_tail: string tree list * string -> string, where the first component of the input tuple is a list of trees and the second is the accumulator.

Place your code in a file called cat.sml and submit it using the submit_cse505 command. Starter code including test cases posted at Resources→Homeworks→cat.sml.

Problem 2: Towards the end of Lecture 10, we discussed a flatten function for a two-level list. Note that a two-level ML list of type 'a list list is homogeneous and hence a list such as [1, [2], 3] is not well-typed.

We can define a heterogeneous multi-level list in ML by a type 'a mixed list by defining a type called 'a mixed with two constructors, base and nest. The use of these two constructors is illustrated below for encoding the heterogeneous list [1, [2, [3, 4]], 5, [6]]:

```
[base(1),  nest([base(2),  nest([base(3), base(4)])]), base(5), nest([base(6)])]
```

We refer to the above list as a mixed list. The constructor base encodes a basic value of any type, and the constructor nest encodes a nested mixed list of the same type of basic value.

(i) Write an ML datatype for the type 'a mixed in terms of constructors base and nest.

(ii) Write a function flatten: 'a mixed list -> 'a list which takes a mixed list as input and produces a single-level list as output. So, for the above example, the output would be [1,2,3,4,5,6].

Place your code for (i) and (ii) in a file called mixed.sml and submit it using the submit_cse505 command. Starter code and test cases posted at Resources→Homeworks→mixed.sml.

Make a directory called A2_P12_UBITId if doing solo; or A2_P12_UBITId1_UBITId2 if doing as a pair (give UBITId's in alphabetic order). Put cat.sml and mixed.sml inside the directory, compress, and submit.

End of Problems 1 and 2