CIS 705 — Programming Languages — Spring 2009

Assignment 3

Due by 2:30 p.m. on Thursday, March 12

The context for this assignment is Chapter 2 of *DAWOC*, the core and standard OCaml libraries, and Assignment 2 (including the model answers `balanced.ml` and `balanced-inefficient.ml`).

Exercise 1 (25 Points)

Using a few well-chosen examples, explain how the versions of

```
val balanced : int list -> int list list
```

defined by `balanced-inefficient.ml` and `balanced.ml` work, and compare their relative efficiencies.

Exercise 2 (35 Points)

The function

```
val insert : 'a -> 'a list -> 'a list
```

of `balanced.ml` runs out of stack space when used with lists containing millions of elements, resulting in the error message

```
Stack overflow during evaluation (looping recursion?).
```

Reimplement `insert` to meet the same specification, but to be usable with lists of millions of elements. Try to minimize the stack space it uses, as well as the number of `::`'s and comparisons that it does; minimizing stack space should have highest priority. You may use as many auxiliary functions as you wish, and these functions may be defined either inside or before `insert`'s definition.

Program in a strictly functional style, without using the imperative features of Chapter 3 of *DAWOC*. Format your program in a way that makes its structure clear. Choose meaningful names for functions, and choose other identifiers with care. Document your functions by abstractly explaining their input/output behavior.

Exercise 3 (40 Points)

Prove that the functions

```
val extend   : int -> int list -> int list option * int list
val balanced : int list -> int list list
```

of `balanced.ml` meet their specifications. You may assume that the auxiliary and library functions they use are correct, as well as that the auxiliary function `bal` of `balanced`'s definition terminates on all inputs meeting its precondition.

(A function meets its specification iff, for all inputs to the function, if those inputs meet the function's precondition, then the function returns a value meeting the function's postcondition. When proving a recursively defined function correct, one may assume that a recursive call whose arguments meet the precondition returns a value that meets the postcondition.)

Submission

Submit your solutions to Exercises 1 and 3 on paper, not electronically.

Email your solution to Exercise 2 to me, being sure to keep an electronic copy. I will acknowledge receiving it. Your submission should *not* include any testing or extraneous code. It should begin with a comment including your name.