Functional Programming and Specification
Practical 2

This is an assessed practical exercise, to be completed by 4pm on **Friday 4th March**. It will be marked but the mark will not contribute to the overall mark for the course. Please use Moscow ML to test your programs before you submit them, using the DICE `submit` command, like so:

> For UG3 students: `submit cs3 fps-3 2` *myfile*$_1$ *myfile*$_2$ ... *myfile*$_n$
> For UG4/MSc students: `submit msc fps-4 2` *myfile*$_1$ *myfile*$_2$ ... *myfile*$_n$

⋆

Your task is to modify an interpreter for a very small subset of an imperative programming language. The Standard ML code for the interpreter is taken from chapter 6 of the book

> Chris Reade. *Elements of Functional Programming*, Addison Wesley (1989).

and can be found in `http://homepages.inf.ed.ac.uk/dts/fps/pract2/reade.sml`. The interpreter is *sectional* in the sense that related types and functions appear close together, but it is not *modular*. There are no signatures or structures; it is a core language program.

Split the interpreter into modules in order to make explicit the necessary dependencies of some parts upon others. You aim should be to identify re-usable entities within the interpreter while also encapsulating information. Each structure (functor) should have an explicit (output) signature.

Some test data is included with the source code of the interpreter. This is for your convenience, and need not be included in your modular version of the interpreter.

For this exercise, you may (but are not required to) work together with one other person of your choice. If you do this, please submit a joint solution bearing both names rather than two separate copies.

⋆

If you want to learn more:

- Extend the expression and command forms in the language that the interpreter implements. Add a simple type system. Modify the parser to allow for error reporting.
- Design a type to represent simple machine instructions and a simple machine language (e.g. lists of machine instructions). Write an interpreter for your machine language. Write a compiler which converts commands to machine language.
- Learn about mosmllex and mosmlyacc — see chapters 17 and 18 of the *Moscow ML Owner's Manual*, available from the Moscow ML website — and use them to replace the lexer and parser of the interpreter.
- Learn about the extensions to the Standard ML module language that Moscow ML implements (in particular: higher-order, first-class and recursive modules)— see chapter 11 of the *Moscow ML Owner's Manual* — and think about how they might be used to structure the interpreter. (These features are not covered by this course; please DO NOT use them in your solution to the exercise.)