Homework 5 — ML — assigned Monday 1 March — due Tuesday 9 March

In this homework, you may use the ML module language to structure your code. If you do, each structure and signature should be in a separate file (e.g., `expressions.sig` and `expressions.sml`). You should only use the SML/NJ top-level to load all the modules with a sequence of `use` invocations and then execute your tests. Do not use `open`. Submit a `LOG` file containing the trace of your SML/NJ session.

Reading assignment

Read Chapter 5, Sections 5.1–5.11 (through page 191) of *ML for the Working Programmer*.

5.1 Parser for a simple expression evaluator (40pts)

We can use the following data type declaration to introduce a language of simple arithmetic expressions:

```
datatype expr = Num of int
              | Add of expr * expr
              | Mul of expr * expr
```

Write a function `parse`, with type `string -> expr`. The function should return an expression corresponding to the text of the string, in accordance with the standard precedence of arithmetic. The input string may consist of digits, + and * signs, parentheses, and white space. For instance,
`parse " 5+3 * 4"` should evaluate to `Add (Num 5, Mul (Num 3, Num 4))`.
In case the input string is not well-formed (for example, the string `"5 + 3 ) * 4"` is not), the function `parse` should raise an exception. *Be sure to formalize the rules for what strings are well-formed, as part of your specification.*

5.2 Boolean formulae: writing recursive functions over algebraic datatypes (30pts)

We can use the following declaration to introduce a language of Boolean formulae:

```
datatype expr = Const of bool
              | Var of int
              | And of expr list
              | Or of expr list
              | Not of expr
```

For instance, the Boolean formula $(\neg x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2)$ is represented by the ML term `And [Or [Not (Var 1), Var 2, Var 3], Or [Var 1, Not (Var 2)]]`.

5.2.1 Simple Boolean evaluator (10pts)

Write a function `eval`, with type `env -> expr -> bool`, which computes the Boolean value of a formula.

The type `env` is the type of environments; an environment is simply an assignment of Boolean values to variables x_i. Choose your own ML representation for the type `env`.

5.2.2 More on Boolean formulae: satisfiability checker (10pts)

Write a function `satisfiable: expr -> bool`, which determines if the given formula is satisfiable, i.e., true for some assignment of Boolean values to the variables that appear in the formula.

5.2.3 More on Boolean formulae: tautology checker (10pts)

Write a function `tautology: expr -> bool`, which determines if the given formula is a tautology, i.e., true for *all* possible assignments of Boolean values to the variables that appear in the formula.

5.3 ⋆Sets: writing recursive functions over algebraic datatypes (30pts)

Devise an ML representation for sets, using an ML datatype. The representation *must* be such that it is possible to do *all* of the following:

- (10pts) Define an appropriate *fold* function.
- (10pts) Write an ML function `powerset` that takes a set S and returns its powerset 2^S.
- (10pts) Let $f(0) = \emptyset$ and $f(k+1) = 2^{f(k)}$ for $k = 0, 1, \ldots$. Write an ML function `f` that implements the mathematical function f. Compute the sets $f(k)$ for $k = 0, 1, 2, 3, 4$.

How to turn in

Make sure that you have thoroughly tested your code, and include all your test runs!

Turn in your code by running

˜clint/handin your-file

on a regular UNM CS machine. You should use whatever filename is appropriate in place of your-file.

Include the following statement with your submission, signed and dated:
I pledge my honor that in the preparation of this assignment I have complied with the University of New Mexico Board of Regents' Policy Manual, including Section 4.8, Academic Dishonesty.