COMS W4115
Programming Languages and Translators
Homework Assignment 1

Prof. Stephen A. Edwards
Columbia University

Due October 17th, 2012
at 11:59 PM

Submit a single file, "hw1.ml," electronically via Courseworks. Modify the hw1.ml template provided on the class webpage.

Do this assignment alone. You may consult the instructor or a TA, but not other students.

All the problems ask you to use O'CAML. You may download the compiler from caml.inria.fr.

The hw1.tar.gz file on the class webpage contains a Makefile that will compile your solutions; if you do not have the benefit of *make*, the sequence of commands is

```
ocamllex scanner.mll
ocamlyacc parser.mly
ocamlc -c ast.mli
ocamlc -c parser.mli
ocamlc -c scanner.ml
ocamlc -c parser.ml
ocamlc -c hw1.mli
ocamlc -c hw1.ml
ocamlc -c runtests.ml
ocamlc -o runtests scanner.cmo parser.cmo \
   hw1.cmo runtests.cmo
```

The file *expected-output* shows what a working version of hw1.ml should produce from *runtests*.

1. In O'Caml, write a function "rle" that takes a list and returns a run-length-encoded version of the list. The result should be a list of pairs: the first member of each pair should be the item being repeated; the second member is an integer representing the number of times it appeared. This is a bit like the "uniq" command in Unix with the "count" option.

   For example, `rle [1;1;1;2;3;3]` should return `[(1,3);(2,1),(3,2)]` and `rle ['a';'a';'b';'b';'b']` should return `[('a',2);('b';3)]`.

   Hint: my solution is seven lines and includes a helper function that measures the length of a single run.

2. Write a program that takes a Boolean expression such as `(A+B)(!A+!B)(A!B+A)` and displays its truth table:

   ```
   00 0
   01 0
   10 1
   11 0
   ```

   I've written the scanner, parser, and expression data structure for you.

   (a) Write a function `free_vars` that takes an `Ast.expr` and returns a *sorted list* of all the unique variables in the expression. Hint: Collect them using a CharSet.

   (b) Write an `eval` function that takes an environment—a `CharMap` that maps variable names (characters) to Boolean values—and an expression and returns the *Boolean value of the expression in the environment*. You may assume every variable that appears in the expression will be in the `CharMap`.

   (c) Use your solutions for parts (a) and (b) to write a function `string2tt` that takes a Boolean expression string and returns a string containing its truth table. Use the `binary_envs` function I've provided to create a list for the rows of this truth table.

   Make sure the result string is formatted as shown above: each line should start with the values of the variables (in alphabetical order) followed by a space followed by the value of the expression for those variables. The string should not have a final newline.

   Hint: use `String.concat` to assemble strings and newlines.