COMS W4115
Programming Languages and Translators
Homework Assignment 1

Prof. Stephen A. Edwards
Columbia University

Due February 17, 2016
at 2:40 PM

Submit your assignment on paper (e.g., printouts) at the beginning of class. **Include demonstrations that your solutions work.** E.g., for each problem, include a printout showing how your solution compiles and runs a few test cases.

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

All the problems ask you to use OCaml. You may download the compiler from ocaml.org.

1. Write a function that subtracts positive integers represented as lists of decimal digits. For example,

```
subl [2;5;3] [5;7] = [1;9;6]
subl [1;0;0;0;0;0;0;0;0;0;0;0]
       [4;2;0;0;0;0;0;0;0;0;0] =
     [0;5;8;0;0;0;0;0;0;0;0;0]
```

   Your algorithm may assume the first number is larger than the second. Arbitrary-precision arithmetic packages use a similar technique but with a much larger radix.

2. Write a word frequency counter starting from the following ocamllex program (wordcount.mll) that gathers all the words in a file and prints them.

```
{ type token = EOF | Word of string }

rule token = parse
  | eof { EOF }
  | ['a'-'z' 'A'-'Z']+ as word { Word(word) }
  | _ { token lexbuf }

{
 let lexbuf = Lexing.from_channel stdin in
 let wordlist =
   let rec next l = match token lexbuf with
                      EOF -> l
                    | Word(s) -> next (s :: l)
   in next []
 in
 List.iter print_endline wordlist
}
```

   Replace the List.iter line with code that builds a string map of (word, count) pairs, uses StringMap.fold to convert the map to a list of (count, word) pairs, sorts the pairs using List.sort, and prints them with List.iter.

   Sort the list of (count, word) pairs using

```
 let wordcounts =
   List.sort (fun (c1, _) (c2, _) ->
              Pervasives.compare c2 c1)
     wordcounts in
```

   Compiling and running my (20-more-line) solution:

```
$ ocamllex wordcount.mll
4 states, 315 transitions, table size 1284 bytes

$ ocamlc -o wordcount wordcount.ml

$ ./wordcount < wordcount.mll

9 word
7 map
7 let
7 StringMap
6 in
...
```

3. Extend the three-slide "calculator" example shown at the end of the Introduction to OCaml slides (the source is available on the class website) to accept the variables named a through z, assignment to those variables, and sequencing using the "," operator. For example,

```
a = b = 3, b = b + 3, a * b + 2
```

   should print "20"

   Use an array of length 26 initialized to all zeros to store the values of the variables. You'll need to add tokens to the parser and scanner for representing assignment, sequencing, and variable names.

   The ocamllex rule for the variable names, which converts the letters a–z into a VARIABLE token, is

```
| ['a'-'z'] as lit
  { VARIABLE(int_of_char lit - 97) }
```

   The new ast.mli file is

```
type operator = Add | Sub | Mul | Div
type expr =
    Binop of expr * operator * expr
  | Lit of int
  | Seq of expr * expr
  | Asn of int * expr
  | Var of int
```