Harvey Mudd College
Computer Science 60
Fall 2010

Assignment 5
User-Friendly CLI for Unicalc
Due. 11:59 p.m., Wed., 7 October 2010

Construct in Racket a parser for free-form Unicalc expressions. Use it to create a user-friendly CLI for Unicalc arithmetic. For features other than evaluation of expressions, you do only need to provide define, not let, let*, etc. The input grammar is shown below. Each expression must be on a single line of input. Definitions are of the form

define $variable-name = expression

Otherwise the expression is a Unicalc expression to be evaluated. The parser should call the ueval evaluator that you constructed in earlier assignments produce the latter output on the command line. **Note that whitespace is implicitly allowed between major syntactic entities, but not within identifiers, or numerals**.

Production	Meaning
L → S | **define** V = S	L is the start symbol of the grammar. A **line L** is either an expression **S** to be evaluated or a definition.
S → E { {<space> | / } E }*	An **expression** S is an elementary expression **E**, followed by 0 or more additional unit expressions separated by space (representing multiplication) or a / (representing division). Grouping is to the left.
E → P | P ^ I	An **elementary expression E** is a **primary expression P**, optionally followed by a ^ (representing exponentiation) and an **integer I**.
P → C | U | V | '(' S ')'	A **primary expression P** is a coefficient **C**, a unit U, a variable V, or a parenthesized expression.
I → { + | - | ε} D D*	An **integer numeral I** is one or more digits, possibly preceded by a + or -. λ is the empty string.
D → 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9	**D** represents a **digit**.
V → $ Letter {Letter | D}*	A **variable V** is a dollar sign, then a letter, followed by any number of letters or digits.
C	**C** represents a numeric **coefficient**, which can be any unsigned integer or floating numeral.
Letter → a | b | c | … | z | A | B | C | … | Z | _	**L** represents a **letter** (which includes underscore _).
U → Letter | Letter U	A **unit** U is any sequence of letters.

Above, braces are used to group elements of the grammar. For example { + | - | ε} means one alternative of a +, -, or nothing (the empty string). Likewise {…}* means 0 or more repetitions of the part inside the braces. For example the V production says that a variable is a $ followed by a Letter, followed by 0 or more Letters or Digits. This is meant to suggest use of *iteration* rather than recursion, although {…}* could be replaced with a new non-terminal X, and productions X → … X | ε.

Examples, showing input line and both parser and evaluator output:

```
line: 1 foot/hour
parse: (/ (* 1.0 foot) hour)
normalized quantity: 8.46651371E-5 (meter)/(second)

line: 2 mph / (foot/sec)
parse: (/ (* 2.0 mph) (/ foot sec))
normalized quantity: 2.9333333333333336

line: 1 (foot pound_force) / (newton meter)
parse: (/ (* 1.0 (* foot pound_force)) (* newton meter))
normalized quantity: 1.355793454465969

line: 3.14
parse: 3.14
normalized quantity: 3.14

line: 3.14^2
parse: (* 3.14 3.14)
normalized quantity: 9.8596

line: define $x = 10 acre
parse: (define $x (* 10.0 acre))
$x = 40467.102047438835 (meter meter)

line: $x
parse: $x
normalized quantity: 40467.102047438835 (meter meter)

line: $x / yard^2
parse: (/ $x (* yard yard))
normalized quantity: 48399.99999999999

line: 3.456 (joule/coulomb) / volt
parse: (/ (* 3.456 (/ joule coulomb)) volt)
normalized quantity: 3.456

line: 7 furlong / fortnight
parse: (/ (* 7.0 furlong) fortnight)
normalized quantity: 0.001164145635125 (meter)/(second)
```