Harvey Mudd College
Computer Science 65
Fall 2008

Assignment 7
Unicalc, Part I
Due. 11:59 p.m., Wed., 5 November 2008

Construct in Java a Scheme-like interpreter for Unicalc, a computational model supporting units. From the website, you may download a zipped project that contains the basic Openlist classes in source form, as well as some rudimentary Unicalc classes. These files are contained in Java packages openlist and unicalc respectively. The main program is a read-eval-print loop that tries to interpret input as a unit and translate that unit to a Unicalc **Quantity** (a numeric coefficient with numerator and denominator lists of units) using a database that is provided.

The objective of Part I is to extend the read-eval-print loop to an interpreter supporting Scheme syntax for evaluating arithmetic expressions with units. These are the input expressions that are to be support by your program:

1. Bare numerals, such as `123`, `45.6`, `7.8e-9`, *etc*.

2. Bare symbols, such as `cm`

3. Arithmetic expressions, where E1 and E2 are either numerals, symbols, or non-atomic expressions.

   (* E1 E2 . . .) (/ E1 E2) (+ E1 E2 . . .) (- E1 E2)

For example, an input expression could be

```
(/ (* foot pound) (* inch inch))
```

The result of evaluation will always be a *normalized* quantity where possible, e.g.

```
214.29947166656584 (kg) / (meter)
```

A Quantity is said to be *normalized* when (a) there are no units common to the numerator and denominator, and (b) it contains only basic units. A unit is said to be *basic* if there is no specified conversion for it. For example, in the database provided, `meter`, `kg`, and `second` are examples of basic units.

It is noted that the output syntax is slightly different from the input syntax. This is intentional, because in part II, we will replace the S expression input syntax with something friendlier to the average user.

Expressions with only * and / are always normalizable.

Expressions with + and – can be normalized only if the individual arguments can be normalized into the same basic units. If this is not possible, the result should be the string `incompatible-units`. Note that units defined in the database don't necessarily yield a normalized definition. Normalization is part of the problem.

In addition to expressions to be evaluated, there are special forms:

(**add** unit expression)

adds *expression* to the database as the value associated with *unit*. If the unit is already defined in the database, a warning should be issued, and the old value printed out in the input syntax, in case the user wants to re-define it back. For example:

```
Redefined symbol acre, was (* 43560 (foot foot))
```

(**remove** unit)

removes the unit from the database. As above, a message is printed:

```
Removed symbol acre, was (* 43560 (foot foot))
```

(**set** symbol expression)

defines the symbol as a *variable* for subsequent usage within the session.

Units in the database cannot be treated as variables. An error should be printed:

```
Cannot treat a unit as a variable: acre
```

Changes to the database are in effect for the current session. Do not write out the database file.