Harvey Mudd College
Computer Science 42
Fall 2009

Assignment 2
Unicalc API
Due. 11:59 p.m., Tue., 15 September 2009

This is the first part of a 2-part assignment. **API** stands for "Application Programming Interface", essentially the visible functions in a library used to build complete applications. **Unicalc** is a calculator that includes physical and other units, rather than just numbers.

Units are important to computation because they eliminate an element sometimes left to human interpretation. In the area of engineering, failure to interpret numbers without their units specified has been known to lead to failure of space missions for example. In the wikipedia article about NASA's Mars Climate Orbiter, http://en.wikipedia.org/wiki/Mars_Climate_Orbiter, it is stated:

> "The Mars Climate Orbiter was intended to enter orbit at an altitude of 140–150 km (460,000-500,000 ft.) above Mars. However, a navigation error caused the spacecraft to reach as low as 57 km (190,000 ft.). The spacecraft was destroyed by atmospheric stresses and friction at this low altitude. The navigation error arose because a NASA subcontractor (Lockheed Martin) used Imperial units (pound-seconds) instead of the metric system."

A **Unicalc Quantity** consists of three parts:

- A number, called the **multiplier**
- A flat list of symbols, called the **numerator**
- A flat list of symbols, called the **denominator**

Our API specifies that a Quantity is represented by a list of these components, in this order, for example:

```
'(2.5 (kg meter) (second second))
```

For this problem, Quantities will be entered as test cases in this way. In a future assignment, we will construct a Scheme-like interpreter that accepts arithmetic expressions containing quantities.

On the main course page, you will find **unicalc-db.scm**. This is a Scheme source file for the Unicalc **database**. The database is, in effect, a set of equations defining single symbols, called **units**, in terms of Quantities. As a Scheme structure, it is an association list, **unicalc-db**. You will want to download this file and load it with your solution to this assignment: (load "unicalc-db.scm").

The API is a set of functions that you will construct, as described below, along with other helper functions. To describe our functions, we need a couple more definitions.

A symbol that is defined in the database (as the first element of an element of the association list) is called a **defined unit**. A symbol that is not defined is called a **basic unit**. Examples of defined units are: mile, day, pound. Examples of basic units are: kg, second, meter.

Defined units can be **expanded** into equivalent Quantities consisting of only basic units in one or more steps. (You can assume for now that there are no circular definitions in the database.) Also, basic units can be converted into quantities by making them the only element in the numerator list, accompanied by a multiplier of 1 and an empty denominator list.

A Quantity is called **normalized** provided that the following two conditions are true:

- The numerator and denominator consist of only basic units.
- The numerator and denominator are *sorted* in ascending alphabetic order.

We don't require the **user** to provide normalized Quantities, but it helps make the processing more efficient if we use them exclusively inside our functions.

Here are the functions you need to provide in the API. Note that **divide** effectively achieves conversion from one kind of normalized Quantity to another.

Function Call Form	Meaning
(**normalize-unit** Unit)	Returns a normalized Quantity for a single unit.
(**normalize** Quantity)	Converts any Quantity to a normalized Quantity.
(**multiply** Quantity1 Quantity2)	Multiplies two normalized Quantities, returning a normalized Quantity.
(**divide** Quantity1 Quantity2)	Divides normalized Quantity1 by normalized Quantity2, returning a normalized Quantity.
(**add** Quantity1 Quantity2)	Adds normalized Quantity1 to normalized Quantity2, returning a normalized Quantity, provided the quantities are interconvertible. Returns (error "illegal add" Quantity1 Quantity2) value if not. If the
(**subtract** Quantity1 Quantity2)	Subtracts normalized Quantity2 from normalized Quantity1, returning a normalized Quantity, provided the quantities are interconvertible. Returns (error "illegal subtract" Quantity1 Quantity2) value if not.

Note that the database, unicalc-db, is global and behind the scenes, rather than being passed as an argument. Test cases will be provided as **uncialc-tests.scm**.