Tutorial 3: Structs and Symbols and Predicates (oh my!)

CS 135 Fall 2007

September 26-28, 2007

Tutorial this week attempts to cover most of the material from lecture module 3 ("New Types of Data") in preparation for your first midterm. The pace may be a little quicker than previous weeks, so be prepared, and be sure to work out problems on your own and check the solutions when they are posted on Friday afternoon. Good luck on your first exam!

Remember the new first step in the design recipe, Data Analysis and Design. It'll be necessary for most of the examples here.

1 Dates

A simple way to represent a date in some unspecified year is by a pair of numbers for the month and the day. The following functions should work with dates specified in this way.

1.1 Comparison

Write a function `before?` which consumes two dates and produces true iff the first date occurs before the second one (in the same year).

1.2 'Tis what season?

Write a function `season` which consumes a single date and produces a symbol for what season that date is in (one of `'fall`, `'winter`, `'spring`, or `'summer`). For the purposes of this question, say fall is from September 23 to December

21, winter is from December 22 to March 19, spring is from March 20 to June 19, and Summer is from June 20 to September 22.

1.3 Sequential order

Write a function `in-order?` which consumes three dates and returns true iff they occur in that order — that is, starting at the first date and moving forward in time, the second date occurs before the third date does. So for example the dates September 20, January 4, March 10 are in order, but the dates August 15, May 29, December 12 are not. For simplicity, you may assume all three dates given are distinct.

2 Numbers with infinity

Although they have many nice properties, the numbers represented by Scheme do have the limitation that they are finite. To overcome this limitation, define the symbols `'infty+` and `'infty-` to be plus and minus infinity, respectively. The following functions will deal with the mixed data type called "unbounded" which is either a number or one of the two symbols for infinity.

2.1 Squaring

The square of positive or negative infinity is always positive infinity. Using this fact, write a function `square` which consumes an "unbounded" number and produces that number squared.

2.2 Comparison

Now write two functions, `less-than?` and `less-equal?` which consume two unbounded numbers and determine (respectively) if the first is less than or less than or equal to the second.

2.3 Addition

Write a function `sum` which consumes two unbounded numbers and produces their sum. Note that the sum of positive and negative infinity is undefined; your function should produce the symbol `'undef` in this case.

3 Weather readings

For the purposes of this tutorial, a weather reading consists of four pieces of information: air temperature (in degrees Celcius), wind speed (in km/hr), humidity (as a percentage), and an indication of whether or not precipitation is currently falling from the sky.

3.1 Type of precipitation

This is entirely inaccurate, but assume that the type of precipitation which falls depends only upon the air temperature, as follows: Under -1 degrees, it will be snow, greater than 2 degrees it will be rain, and in between it will be sleet. So write a function `precip-type` which consumes a weather reading and produces a symbol indicating the type of precipitation which is falling (one of `'snow`, `'sleet`, or `'rain`), or the symbol `'none` if no precipitation is falling.

3.2 How does it feel?

In Canada, two measures are used to indicate how hot or cold it "feels" outside when the air temperature is not sufficient. These are called the humidex and the wind chill. If the air temperature is T, the humidity is H and the wind speed is V, then the wind chill $W$and humidex U are calculated as follows:

$$W = w_1(T) + w_2(T)V^{0.16}$$
$$U = u_1(T) + u_2(T)H$$

The functions w_1, w_2, u_1 and u_2 are coded in Scheme and given as follows:

```
(define (w1 T)
  (+ 13.12
     (* 0.6215 T)))

(define (w2 T)
  (- (* 0.3965 T)
     11.37))

(define (u1 T)
```

```
  (- T
     (* (/ 5 9) 10)))

(define (u2 T)
  (* (/ 5 9)
     6.112
     (expt 10 (/ (* 7.5 T)
 (+ 237.7 T)))
     (/ 1 100)))
```

Now design and write a function `feels-like` which consumes a weather reading and produces the temperature it "feels like" according to the following rules: If the air temperature is below 10 degrees, report the wind chill; if it is above 20 degrees, report the humidex; and otherwise just report the air temperature.

3.3 Lost data

Some weather data has been lost and we're trying to recover it. We have the actual air temperature and the temperature it "felt like" at the time (according to the function you just wrote), but the complete weather reading has been lost.

So write a function `recover-data` which consumes the actual air temperature and the temperature it "felt like", and produces a weather reading which is consistent with those values. Note that there will always be some values in the weather reading which you can't determine; use the symbol `'unknown` for these.

Also, some values wouldn't make sense, such as a humidity less than 0 or greater than 100, or a wind speed less than 0. If there is no weather reading that could possibly be consistent with the values given, return the symbol `'impossible` instead of a weather reading.