Tutorial 8: Polynomials are the coolest!

CS 135 Fall 2007

October 31-November 2, 2007

Today's tutorial covers material from lecture module 8 in the course notes, on simultaneous list processing.

To practice this, we'll be computing with polynomials represented by a list of *terms*. A term is a pair of nonzero coefficient and exponent representing a single monomial such as $5x^2$.

A single term in Scheme is represented by a `term` structure, which is defined (in the starter code t8-starter.scm) by

```
(define-struct term (coeff expon))
```

where the coefficient is not zero. A polynomial is represented by a list of terms sorted in order of increasing exponents.

So for example the polynomial

$$f(x) = 2x^{99} - 3x^{88} + 5x^{80} - x^7$$

would be represented by the Scheme expression

```
(list (make-term -1 7)
      (make-term 5 80)
      (make-term -3 88)
      (make-term 2 99))
```

1 Addition

Write a function `poly+` which consumes two polynomials and produces another polynomial representing their sum. Keep in mind that the terms are sorted, and that all terms appearing in the list must have nonzero coefficients.

2 Equality

Without using the `equal?` function, write a function `poly=?` that consumes two polynomials and returns `true` iff the two polynomials are the same.

3 Counting identical terms

We know two polynomials are exactly equal iff they have all the same terms. But we might just want to know how many identical terms two polynomials have. For this, write a function `num-same-terms` which consumes two polynomials and produces the number of terms they have in common.

4 Multiplication

Write a function `poly*` to multiply two polynomials. You'll probably want to use the function `poly+` which you wrote earlier, as well as the helper function `term-mul` provided for you in the starter code.

5 Evaluation

To evaluate a polynomial $f(x)$ at a point, we simply replace the value of the point for x in the expression (just like evaluating a Scheme function). So write a function `poly-eval` which consumes a polynomial and a number and evaluates the polynomial at that number.

For this question, you may not use the built-in Scheme function `expt` (because that would just take all the fun out of it). But you may use the helper function `divide-by-x` which is provided in the starter code.

6 Composition

The composition of two polynomials (or any two functions) is simply the first function applied to the result of the second. So the composition of two polynomials $f(x)$ and $g(x)$ is just $f(g(x))$, for example.

Write a function `compose` which consumes two polynomials and produces the single polynomial which represents their composition.

7 Divisibility

As a bonus, write a function `divides?` which consumes two polynomials and produces `true` iff the second polynomial is exactly divisible by the first.

Warning: it may be difficult to write this function, as no one in the world knows how to accomplish this in a reasonable amount of time when polynomials are given in our chosen representation. If you come up with a fast solution, please email it to Dan Roche (mailto:droche@cs.uwaterloo.ca) so that I can take credit for your work and become famous.