Programming Exercises 3: Package

Submissions only via Moodle (due to 23:55, Sep. 19)

September 11, 2014

In this assignment, you will have to implement a "package" of polynomials in one variable using the `list` data type.

```
type polynomial = float list;;
```

A polynomial $p(x)$ of degree n in a single variable x is of the form:

$$c_n x^n + c_{n-1} x^{n-1} + \ldots c_1.x + c_0$$

- The (real) numbers c_i are called *co-efficients* of x^i.
- The powers of x in each term are called the *exponents*

We can also write $p(x)$ as

$$c_0.x^0 + c_1.x^1 + \ldots + c_n.x^n = \Sigma_{i=0}^{n} c_i.x^i$$

The polynomial $p(x)$ can be modelled as a list $[c_0; c_1; \ldots; c_n]$.

- The variable is implicit i.e. we only have to store the coefficients in the list.
- The exponent of each term is given by the position in the list i.e. if we have a list `[0.5; 0.0; 2.1]`, we can interpret it as the polynomial $2.1x^2 + 0.5$ since 0.5 is in the 0th position in the list and 2.1 is in the 2nd position. The 1st position contains 0.0 and so the coefficient of x is 0 in this polynomial.
- Only the coefficient c_i is listed at position i in the list.
- $c_n \neq 0.0$ i.e. given a polynomial of degree n, the list representing that polynomial must have size exactly $n + 1$. Any of these $k + 1$ entries may be 0.0 *except* the last one i.e. c_n. This corresponds to the standard way of writing polynomials e.g. if we consider $2x^3 + 3$, it can be thought of as the same polynomial as $0x^5 + 2x^3 + 3$ but we write it only as $2x^3 + 3$.

Naively evaluating $p(x)$ at x_0 involves about $n^2/2$ multiplications and n additions. Instead we can use *Horner's Rule*. Consider:

$$p(x_0) = (\ldots((c_n.x_0 + c_{n-1}).x_0 + c_{n-2}).x_0 + \ldots + c_1).x_0 + c_0$$

We can perform a tail recursive computation involving only n multiplications and n additions, by maintaining a partial product *pp*. At each stage, we multiply *pp* with x_0 and add the next lower coefficient..

You will need to write programs to perform the following operations on polynomials

1. Evaluation or finding out the value of a polynomial $p(x)$ for some given value x_0 for x, using Horner's Rule. `evaluate: polynomial -> float -> float` finds the value of a given polynomial $p(x)$ for the value x_0. For example, the polynomial $3x^2 - 4x + 6$ has the value 4.75 at $x = 0.5$.

2. Addition. A function `addpoly: polynomial -> polynomial -> polynomial` which takes the representations of polynomials $p_1(x)$ and $p_2(x)$ and returns the representation of $p_1(x) + p_2(x)$. The two polynomials need not have the same degree. For example, the addition of $3x^4 - x + 2$ and $2x^2 + x - 7$ yields $3x^4 + 2x^2 - 5$.

3. Multiplication. A function `multpoly: polynomial -> polynomial -> polynomial` which takes the representations of polynomials $p_1(x)$ and $p_2(x)$ and returns the representation of $p_1(x) \times p_2(x)$. The two polynomials need not have the same degree. For example, the product of $(x + 3)$ and $4x^2 - 2$ yields $4x^3 + 12x^2 - 2x - 6$.

4. Differentiation. A function `derivpoly: polynomial -> polynomial` that takes the representation of a polynomial $p(x)$ and returns the representation of the polynomial $\frac{d}{dx}p(x)$ which is the derivative of $p(x)$ with respect to variable x. For example, the derivative of $3x^2 - 4x + 3$ is $6x - 4$.

5. (*Bonus question. This part is not compulsory. Extra marks will be given for it if you attempt it correctly.*) A function `dividepoly: polynomial -> polynomial -> polynomial * polynomial` takes the representations of polynomials $p_1(x)$ and $p_2(x)$ and returns the representation of the quotient and remainder of polynomial division $p_1(x)/p_2(x)$. The two polynomials need not have the same degree.