Logistics

This homework is officially due in class on **Wednesday, February 26**. However, it comes with an automatic extension: anything submitted by **5pm on Friday, February 28** will be accepted as being on time.

You should work singly or in pairs on this assignment.

You should turn in a hard copy of your source code and a transcript demonstrating convincingly that your code is correct. You should also submit your code electronically.

Exercises (due Wednesday, February 26)

1. Every polynomial \( a_0 + a_1x + a_2x^2 + \cdots + a_nx^n \) can be represented as a list \([a_0, a_1, a_2, \ldots, a_n]\). For example, \( 1 + 4x^2 + 2x^3 + 6x^4 \) can be represented as the list \([1,0,4,2,6]\).

   For the purposes of the assignment, we’ll limit ourselves to polynomials whose coefficients are integers. Thus we can introduce the following type synonym:

   ```haskell
type Poly = [Int]
```

   The effect of this piece of code (be sure to include it in your file) is to introduce a new name for an already existing type. Here, we’re introducing the new name `Poly` for the already existing type `[Int]`: the two types are completely identical, and either one can be used in place of the other.

   (a) Write a Haskell function

   ```haskell
   evalPoly :: Int -> Poly -> Int
   such that evalPoly m p returns the value of the polynomial p when x is m. For example, evalPoly 3 [1,0,4,2,6] should return 577 (1 + 0 * 3^2 + 4 * 3^3 + 2 * 3^4 + 6 * 3^5 = 577).
   ```

   (b) Write a Haskell function

   ```haskell
   addPoly :: Poly -> Poly -> Poly
   such that addPoly p q returns the sum of the polynomials p and q. For example, addPoly [3, 2, 0, 7, 3] [1, 0, 6, 5] should return [4, 2, 6, 12, 3].
   ```

   (c) Recall that the derivative of a polynomial \( a_0 + a_1x + a_2x^2 + \cdots + a_nx^n \) is the polynomial

   \( a_1 + 2a_2x + 3a_3x^2 + \cdots + na_nx^{n-1} \).

   Write a Haskell function

   ```haskell
   derivPoly :: Poly -> Poly
   such that derivPoly p returns the derivative of the polynomial p. For example, derivPoly [1, 0, 3, 4] and derivPoly [5, 0, 3, 4] should both return [0, 6, 12], while derivPoly [2, 5, 1, 0, 2] should return [5, 2, 0, 8].
   ```

   (d) Write a Haskell function

   ```haskell
   multPoly :: Poly -> Poly -> Poly
   ```
such that \texttt{multPoly} \( p \) \( q \) returns the sum of the polynomials \( p \) and \( q \). For example, \texttt{multPoly \([1,0,7,3]\) \([5,2,3]\)} should return \([5,2,38,29,27,9]\).

**Hint:** You should make use of two facts about polynomial multiplication:

i. Multiplication is distributive over addition: for example,

\[
(3x+7x^2+8x^4)(5+4x+10x^5) = 3x(5+4x+10x^5)+7x^2(5+4x+10x^5)+8x^4(5+4x+10x^5).
\]

ii. Multiplying a polynomial by \( x^n \) “shifts” all exponents in that polynomial by \( n \). For example, multiplying \( 3x + 7x^2 + 8x^4 \) by \( x^2 \) results in the polynomial \( 3x^3 + 7x^4 + 8x^6 \); in terms of our Haskell representation, \([0, 3, 7, 0, 8]\) is shifted by 2, resulting in \([0, 0, 0, 3, 7, 0, 8]\).

2. One string is a substring of another string if all of the characters in the first appear in the second, in the same order. For example, "music" is a substring of "computer science", but "cis" is not. Note that the empty string "" is a substring of every string.

(a) Write a Haskell function

\[
\texttt{isSubstring :: String -> String -> Bool}
\]

such that \texttt{isSubstring s1 s2} returns \texttt{True} if \( s1 \) is a substring of \( s2 \), and returns \texttt{False} otherwise.

(b) Write a Haskell function

\[
\texttt{substrings :: String -> [String]}
\]

such that \texttt{substrings s} returns a list containing all substrings of \( s \). For example, it should have the following behavior (note that the order of elements in the list is unimportant to me):

\[
\texttt{Main> substrings "abcd"}
\]

\[
[",",",c",",cd",",b",",bd",",bc",",bcd",",a",",ad",",ac",",acd",",ab",",abd",",abc",",abcd"]
\]

Two notes: First, despite the common theme, this question has very little to do with the previous one. Second, this question does not require a lot of code, it just requires a little careful thought.