Assignment 6

COMP 302 Programming Languages and Paradigms
Prakash Panangaden

Due Date: 11th April 2016

This assignment has two programming questions **only**. Put the programming assignment in a file with the extension .fs as usual. Both questions involve stream programming. The file assignment6.fs on the course web page has some predefined functions that you may find useful.

Question 1[10 points] The decimal expansion of a rational number may be infinite though it will have to repeat itself at some point of course. In this question I want you to compute the whole possibly infinite sequence of digits obtained. We will take the radix of the number system as a parameter. Please write a function `expand` that does this. The type and a couple of examples are shown below.

```
val expand : num:int -> den:int -> radix:int -> seq<int>
> let oneSeventh = expand 1 7 10;;
> prefix 10 oneSeventh;;
val it : int list = [1; 4; 2; 8; 5; 7; 1; 4; 2; 8]
> let threeSeventh = expand 3 7 10;;
val threeSeventh : seq<int>
> prefix 10 threeSeventh;;
val it : int list = [4; 2; 8; 5; 7; 1; 4; 2; 8; 5]
> let oneSeventeenth = expand 1 17 10;;
val oneSeventeenth : seq<int>
> prefix 20 oneSeventeenth;;
val it : int list =
  [0; 5; 8; 8; 2; 3; 5; 2; 9; 4; 1; 1; 7; 6; 4; 7; 0; 5; 8; 8]
> prefix 10 (expand 1 31 10);;
val it : int list = [0; 3; 2; 2; 5; 8; 0; 6; 4; 5]
```

Question 2[30 points] In this question you will define some simple functions to create and work with *power series*. A power series is a representation of a function in the form

$$f(x) = c_0 + c_1x + c_2x^2 + c_3x^3 + c_4x^4 + \ldots$$

where the coefficients c_i are arbitrary real numbers and x is a real variable. Functions that have such a power series representation are called *analytic functions*. For some values of x the series may converge and for others it may diverge. We will not worry about these subtleties and will happily write series that may be divergent. These are called *formal* power series. A famous example is the exponential series

$$e^x = 1 + x + x^2/(2!) + x^3/(3!) + \ldots + x^n/(n!) + \ldots$$

This series actually converges for all x. It also satisfies the remarkable identities

$$\frac{\mathrm{d}}{\mathrm{d}x}\, e^x = e^x \text{ and } \int e^x \,\mathrm{d}x = e^x.$$

We will use the latter equation to construct the exponential series recursively. But first things first.

We define a type of terms:

```
type term = Term of float * int
```

F# provides a function `pown` to raise a float to an integer power. Using this one can code a function `evalTerm`; I have done this for you.

1. Write a function `integrateTerm` that computes the indefinite integral of a term symbolically using the standard formula $\int cx^n \mathrm{d}x = \frac{cx^{n+1}}{n+1}$.
2. We represent a power series as an infinite stream of terms. Write a function that computes the integral of such a stream lazily producing an infinite stream as result. Here is the type

   ```
   val integrateSeries : sigma:seq<term> -> seq<term>
   ```

3. Using (co)-recursion and the second equation above for the exponential series, create an infinite stream representation of the exponential series.
4. Write a function `sumSeries` that computes the sum of the first n terms of a power series evaluated at x. The type is shown below

   ```
   val sumSeries : sigma:seq<term> -> x:float -> n:int -> float
   ```

Here are some examples of the code in action:

```
val evalTerm : t:term -> x:float -> float
> let t1 = Term (2.0,5);;
val t1 : term = Term (2.0,5)
> evalTerm t1 1.1;;
val it : float = 3.22102
val integrateTerm : t:term -> term
> t1;;
```

```
val it : term = Term (2.0,5)
> integrateTerm t1;;
val it : term = Term (0.3333333333,6)

val integrateSeries : sigma:seq<term> -> seq<term>
let sigma1 = Seq.initInfinite (fun i -> Term(1.0,i))
> prefix 5 sigma1;;
val it : term list =
  [Term (1.0,0); Term (1.0,1); Term (1.0,2); Term (1.0,3); Term (1.0,4)]
let sigma2 = integrateSeries sigma1
val sigma2 : seq<term>
> prefix 5 sigma2;;
val it : term list =
  [Term (1.0,1); Term (0.5,2); Term (0.3333333333,3); Term (0.25,4);
   Term (0.2,5)]
>
val sumSeries : sigma:seq<term> -> x:float -> n:int -> float

> sumSeries sigma1 1.0 5;;
val it : float = 5.0
> sumSeries sigma2 1.0 7;;
val it : float = 2.592857143

warning FS0040: This and other recursive references to the object(s) being
defined will be checked for initialization-soundness at runtime through the
use of a delayed reference. This is because you are defining one or more
recursive objects, rather than recursive functions. This warning may be
suppressed by using '#nowarn "40"' or '--nowarn:40'.

val expSeries : seq<term>

> prefix 5 expSeries;;
val it : term list =
  [Term (1.0,0); Term (1.0,1); Term (0.5,2); Term (0.1666666667,3);
   Term (0.04166666667,4)]
> sumSeries expSeries 1.0 10;;
val it : float = 2.718281526
> #quit;;
```