Introduction to Functional Programming

Exercises on functions and lists

1. Write a functional to compute the minimum value $\min_{k=0}^{\ell} f(k)$ of a function f on the integers. Use the functional to express the two dimensional minimum $\min_{i=0}^{m} \min_{j=0}^{n} g(i,j)$ of an integer valued function g of two integer arguments.

2. Write a function of type `'a list list -> bool` that checks whether or not its input is a matrix. Give versions with list functionals and without.

3. Give implementations for all the operations on sets that you can think of (including membership testing, intersection, union, complement, inverse and direct images, characteristic function, *etc.*) for the following two representations

   (a) `type 'a set = 'a -> bool`

   (b) `type 'a set = 'a list`

   of sets.

4. Consider the following type

   ```
   type ('i,'o,'s) IODAutomata' = 's * 'i -> 'o * 's option
   ```

   of input/output deterministic automata.

   Write simple recursive and tail recursive versions of a function

   ```
   val run = fn :
         ('i,'o,'s) IODAutomata' -> 's option -> 'i list -> 'o list
   ```

   such that

   ```
   run A s i
   ```

   `run`s the automata `A` from the initial state `s` with the input sequence `i` producing the corresponding output sequence.

   Repeat the exercise for the following type of automata:

   ```
   type ('i,'o,'s) IODAutomata = 's * 'i -> ('o * 's) option
   ```

5. Compare the (evaluation of) the simple recursive, tail recursive, and the following continuation-passing style version

   ```
   fun CPSfact n
     = let
         fun auxCPSfact( n , k )
           = if n = 0 then k(1)
             else auxCPSfact( n-1 , fn x => k(n*x) )
       in
         auxCPSfact( n , fn x => x )
       end ;
   ```

   of the factorial function.