Proyecto 4

Algoritmos y Estructuras de Datos I
Laboratorio

2 de octubre de 2006

El proyecto consta de cuatro ejercicios. Los ejercicios no se pueden especificar en el formalismo del teórico pero pueden ser resueltos aplicando el razonamiento inductivo como se hizo en el ejercicio de las torres de Hanoi y en el proyecto anterior.

1. Programar la función

   ```
   inics :: :: [a] -> [[a]]
   ```

   donde `inics xs` devuelve la lista de todos los tramos iniciales de la lista `xs`. Por ejemplo:

   ```
   inics [2,4,1] = [[],[2],[2,4],[2,4,1]]
   inics [3]     = [[],[3]]
   inics []      = [[]]
   ```

   Ayuda: pensar inducción sobre la lista. O sea, ver que devuelve la función para el caso en que la lista sea vacía y para el caso en que no lo sea. Para el caso no vacía es posible que necesite crear una función auxiliar. Para ella aplique también el razonamiento inductivo.

2. Hacer el ejercicio anterior sin utilizar una función auxiliar.

   Ayuda: use el funcional predefinido `map` del proyecto anterior.

3. Programar la función

   ```
   perms :: [a] -> [[a]]
   ```

   donde `perms xs` devuelve la lista de todas las permutaciones posibles de `xs`. Por ejemplo:

   ```
   perms [4,1]   = [[4,1],[1,4]]
   perms [2,4,1] = [[2,4,1],[4,2,1],[4,1,2],[2,1,4],
                    [1,2,4],[1,4,2]]
   perms [4]     = [[4]]
   perms []      = [[]]
   ```

Ayuda: la misma ayuda que el primer ejercicio.

Ayuda: para la función auxiliar ver si se puede usar la funcion `intercal`.

4. Hacer el ejercicio anterior solamente con la función `intercal` como función auxiliar.

   Ayuda: investigue (en el Prelude.hs) y use el funcional predefinido `concatMap` de Haskell.