Programación para la Inteligencia Artificial
Haskell - Práctica 4 (parte I)
Manejo de listas ordenadas por recursión

1. Definir por recursión la función

   $ordenada :: Ord\ a => [a] -> Bool$

   tal que `(ordenada xs)` se verifica si `xs` es una lista ordenada de menor a mayor. Por ejemplo,

   ```
   ordenada [2,3,5] == True
   ordenada [2,5,3] == False
   ```

2. Definir por recursión la función

   $borrar :: Eq\ a => a -> [a] -> [a]$

   tal que `(borrar x xs)` es la lista obtenida borrando una ocurrencia de `x` en la lista `xs`. Por ejemplo,

   ```
   borrar 1 [1,2,1] == [2,1]      borrar 3 [1,2,1] == [1,2,1]
   ```

3. Definir la función `insertar` tal que `insertar e l` inserte `e` en la lista ordenada `l`.

4. Definir la función `ordInsercion` que tome una lista no ordenada y la ordene usando la función anterior.

5. Definir la función `minimo` que calcule el mínimo elemento en una lista no necesariamente ordenada, usando la función anterior.

6. Definir por recursión la función

   $mezcla :: Ord\ a => [a] -> [a] -> [a]$

   tal que `(mezcla xs ys)` es la lista ordenada obtenida mezclando las listas ordenadas `xs` e `ys`. Por ejemplo,

   ```
   mezcla [2,5,6] [1,3,4] == [1,2,3,4,5,6]
   ```

7. Definir la función

   $mitades :: [a] -> ([a],[a])$

   tal que `(mitades xs)` es el par formado por las dos mitades en que se divide `xs` tales que sus longitudes difieren como máximo en uno. Por ejemplo,

   ```
   mitades [2,3,5,7,9] == ([2,3],[5,7,9])
   ```

8. Definir por recursión la función

   $ordMezcla :: Ord\ a => [a] -> [a]$ tal que `(ordMezcla xs)` es la lista obtenida ordenando `xs` por mezcla (es decir, considerando que la lista vacía y las listas unitarias están ordenadas y cualquier otra lista se ordena mezclando las dos listas que resultan de ordenar sus dos mitades por separado). Por ejemplo,

   ```
   ordMezcla [5,2,3,1,7,2,5] == [1,2,2,3,5,5,7]
   ```

9. Definir por recursión la función

   $esPermutacion :: Eq\ a => [a] -> [a] -> Bool$

   tal que `(esPermutacion xs ys)` se verifica si `xs` es una permutación de `ys`. Por ejemplo,

   ```
   esPermutacion [1,2,1] [2,1,1] == True
   esPermutacion [1,2,1] [1,2,2] == False
   ```