Programación para la Inteligencia Artificial
Práctica 8 - Ejercicio de Autoevaluación

La **codificación por longitud**, o **compresión RLE** (*Run-length encoding*)[1], es una compresión de datos en la que se almacenan secuencias de datos con el mismo valor consecutivas usando un único valor más su recuento. Por ejemplo, la cadena

```
BBBBBBBBBBBBNBBBBBBBBBBBBNNNBBBBNBBB
```

se codifica por

```
12B1N12B3N4B1N3B
```

Esto se interpreta como 12 letras B, 1 letra N , 12 letras B, 3 letras N, etc.
Se pide:

A) Crear un módulo, denominado `ListaPar.hs`, y en él:

1. Definir un **tipo de dato nuevo**, `ListaP a`, que consista en una lista formada por parejas del tipo `(Int,a)`.
2. Declarar este tipo como una instancia de la clase `Show`, de manera que una lista que contenga, por ejemplo, los elementos
   `[(12,'B'),(1,'N'),(12,'B'),(3,'N'),(19,'B')]`
   se visualice como:
   `12->'B'`
   `1->'N'`
   `12->'B'`
   `3->'N'`
   `19->'B'`
3. Declarar `ListaP2 a` como un **renombramiento** para el mismo tipo de dato, es decir, para una lista de `(Int,a)`.
4. Definir la función
   ```
   comprimida ::  [a] -> ListaP2 a
   ```
   tal que `(comprimida xs)` es la lista obtenida al comprimir por longitud la lista `xs`. Por ejemplo:
   ```
   ghci> comprimida [1,1,7,7,7,5,5,7,7,7,7]
   [(2,1),(3,7),(2,5),(4,7)]
   ghci> comprimida "BBBBBBBBBBBBNBBBBBBBBBBBBNNNBBBBBBBBB"
   [(12,'B'),(1,'N'),(12,'B'),(3,'N'),(9,'B')]
   ```
5. Definir la función
   ```
   expandida ::  ListaP2 a -> [a]
   ```
   tal que `(expandida ps)` es la lista expandida correspondiente a `ps` (es decir, es la lista `xs` tal que la comprimida de `xs` es `ps`). Por ejemplo:

[1] Ejercicios extraídos de http://www.glc.us.es/~jalonso/

```
ghci> expandida [(2,1),(3,7),(2,5),(4,7)]
[1,1,7,7,7,5,5,7,7,7,7]
```

6. Exportar de este módulo sólo aquello que se use en la definición de las funciones incluidas en el fichero principal siguiente.

B) Usar el módulo anterior en un nuevo fichero en el que además se debe:

1. Definir por **renombramiento**, a partir de `ListaP2 a`, un tipo `ListaPCh`, que consista en una lista de `(Int,Char)`.
2. Definir la función

   ```
   listaAcadena ::  ListaPCh -> String
   ```

   tal que `(listaAcadena xs)` es la cadena correspondiente a la lista de pares de `xs`. Por ejemplo:

   ```
   ghci> listaAcadena [(12,'B'),(1,'N'),(12,'B'),(3,'N'),(19,'B')]
   "12B1N12B3N19B"
   ```

3. Definir la función

   ```
   cadenaComprimida ::  String -> String
   ```

   tal que `(cadenaComprimida cs)` es la cadena obtenida comprimiendo por longitud la cadena `cs`. Por ejemplo,

   ```
   ghci> cadenaComprimida "BBBBBBBBBBBBNBBBBBBBBBBBBNNNBBBBBBBBBBNNN"
   "12B1N12B3N10B3N"
   ```

4. Definir la función

   ```
   cadenaAlista ::  String -> ListaPCh
   ```

   tal que `(cadenaAlista cs)` es la lista de pares correspondientes a la cadena `cs`. Por ejemplo,

   ```
   ghci> cadenaAlista "12B1N12B3N10B3N"
   [(12,'B'),(1,'N'),(12,'B'),(3,'N'),(10,'B'),(3,'N')]
   ```

5. Definir la función

   ```
   cadenaExpandida ::  String -> String
   ```

   tal que `(cadenaExpandida cs)` es la cadena expandida correspondiente a `cs` (es decir, es la cadena `xs` que al comprimirse por longitud da `cs`). Por ejemplo:

   ```
   ghci> cadenaExpandida "12B1N12B3N10B3N"
   "BBBBBBBBBBBBNBBBBBBBBBBBBNNNBBBBBBBBBBNNN"
   ```

6. Definir una función

   ```
   main ::  IO()
   ```

   que solicite la opción codificar/decodificar, a continuación la cadena, y devuelva el resultado correspondiente.