Programación Lógica y Funcional

Práctica 4

En esta práctica haremos uso de **strings, listas por comprensión, tuplas y rangos**. Recordemos que un rango permite establecer una lista de valores sucesivos enumerando solo el primer elemento (o los dos primeros) y el último elemento (o ninguno en caso de listas infinitas). Las tuplas permiten agrupar datos de diferentes tipos y las listas por comprensión permiten construir listas describiendo las características de sus elementos.

1. Una técnica de compresión de información consiste en identificar repeticiones de datos y establecer una codificación para que los datos puedan ser descomprimidos posteriormente. Escriba la función **comprime s** que recibe un string y regresa una lista de tuplas que contienen cada carácter del string y el número de repeticiones de ese carácter. Ejemplos:

```
*Main> comprime "mmmmmmmmmmnnaaa"
[('m',10),('n',2),('a',3)]
*Main> comprime "aaaaaabbbbbb"
[('a',6),('b',6)]
*Main> comprime "aaaaaabbbbbbaaaaa"
[('a',6),('b',6),('a',5)]
```

2. El famoso problema de las **Torres de Hanoi** consiste en mover n aros de una torre inicial a una torre final utilizando una tercera torre intermedia como auxiliar. Las restricciones son que solo se puede mover un solo aro de una torre a otra y que en ningún momento puede quedar un aro sobre uno más pequeño. Escriba la función **hanoi** que recibe el número de aros y el nombre de las tres torres y regresa un string que contiene todos los movimientos necesarios para resolver el problema. Los movimientos deberán estar numerados y uno en cada línea como se muestra en el ejemplo:

```
*Main> hanoi 3 "Torre A" "Torre B" "Torre C"
1.-mueve aro 1 de Torre A a Torre C
2.-mueve aro 2 de Torre A a Torre B
3.-mueve aro 1 de Torre C a Torre B
4.-mueve aro 3 de Torre A a Torre C
5.-mueve aro 1 de Torre B a Torre A
6.-mueve aro 2 de Torre B a Torre C
7.-mueve aro 1 de Torre A a Torre C
```

Sugerencias: Utilizar la funciones **putStr** (para mostrar el string), **zip** (para la numeración) y **show** (para convertir números a strings). Utilizar el string **"\n"** para el salto de línea.

3. Una **permutación** es la variación del orden o de la disposición de los elementos de un conjunto. Así por ejemplo “abcd” es una permutación del conjunto {a,b,c,d}. Otra sería “bacd”. En general, el número de permutaciones de un conjunto de *n* elementos es *n!* (factorial de *n*). Escriba la función **permuta** que recibe una lista y regresa la lista de todas las permutaciones de esa lista. Ejemplos:

```
*Main> permuta "abcd"
["abcd","abdc","acbd","acdb","adbc","adcb","bacd","badc","bcad","b
cda","bdac","bdca","cabd","cadb","cbad","cbda","cdab","cdba","dabc
","dacb","dbac","dbca","dcab","dcba"]
*Main> permuta [1,2,3]
[[1,2,3],[1,3,2],[2,1,3],[2,3,1],[3,1,2],[3,2,1]]
```

Enviar un archivo con extensión hs que contenga TODAS las funciones necesarias para ejecutar sin problemas cada una de las funciones que se piden en la práctica**. Las funciones deberán llamarse tal y como de muestra en los ejemplos**, de lo contrario no serán tomadas en cuenta.