Programación declarativa
6 de Noviembre de 2013

Apellidos : ....

Nombre : ....

Nota: cada una de las definiciones debe ir acompañada del tipo más general que permita realizar la operación solicitada.

Ejercicio 1. Se dice que un número natural, `n`, es reversible si su última cifra NO es 0 y la suma de `n` y el número que se obtiene escribiendo las cifras de `n` en orden inverso tiene todas sus cifras impares.

Ejemplos:

- 36 es reversible porque $36 + 63 = 99$ y 99 tiene todas sus cifras impares.
- 243 no es reversible porque $243 + 342 = 585$ y 585 no tiene todas sus cifras impares.

Definir una función `esReversible` que determine si un número natural es, o no, reversible.

Así, `esReversible 36 == True` y `esReversible 243 == False`.

Nota: haga uso de tantas funciones auxiliares como necesite (por ejemplo, para descomponer un número en cifras).

Una matriz puede representarse como una lista de listas, donde cada una de las listas representa una fila de la matriz. Por ejemplo, la matriz

$$\begin{pmatrix} 1 & 0 & -2 \\ 0 & 3 & -1 \end{pmatrix}$$

puede representarse por `[[1, 0, 2], [0, 3, -1]]`.

Nota: esta matriz no tiene por qué ser numérica.

Ejercicio 2. Definir una función `diagonal` que devuelva una lista con los elementos de la diagonal principal de una matriz cuadrada; es decir, con el mismo número de filas que de columnas.

$$\texttt{diagonal} \begin{pmatrix} 1 & 0 & -7 & 5 \\ 3 & -1 & 4 & 2 \\ 5 & 6 & 2 & -3 \\ -8 & 0 & 9 & 4 \end{pmatrix} = [1, -1, 2, 4]$$

Así, `diagonal [[1,0,-7,5],[3,-1,4,2],[5,6,2,-3],[-8,0,9,4]] == [1,-1,2,4]`.

Ejercicio 3. Definir una función `productoMatricesC` que permita calcular el producto de dos matrices cualesquiera con dimensiones adecuadas. Utilizar para ello las listas por compresión. Lógicamente, estas matrices sí deberán ser numéricas.

Ejemplo de producto de matrices:

$$\begin{pmatrix} 1 & 0 & -2 \\ 0 & 3 & -1 \end{pmatrix} \cdot \begin{pmatrix} 0 & 3 \\ -2 & -1 \\ 0 & 4 \end{pmatrix} = \begin{pmatrix} 0 & -5 \\ -6 & -7 \end{pmatrix}$$

Nota: en caso de ser necesario se puede utilizar la función `transpose` del módulo `Data.List` para calcular la traspuesta de una matriz.

Recuerde que la forma de obtener cada posición $C_{i,j}$ *de la matriz resultante de multiplicar* A *por* B*, proviene del procesamiento de la fila* A_i *y la columna* B_j*, haciendo en cada caso el* $\Sigma_k(A_{i,k} * B_{k,j})$*. En definitiva: para obtener la primera posición, proveniente de la fila* $[1, 0, -2]$ *y la columna* $[0, -2, 0]$ *hacemos* $1 * 0 + 0 * (-2) + (-2) * 0$*, que devuelve como vemos 0.*

Ejercicio 4. Definir una función recursiva `traspuestaR` que permita calcular la traspuesta de una matriz.

Ejemplo de traspuesta de una matriz:

$$\begin{pmatrix} 0 & 3 \\ -2 & -1 \\ 0 & 4 \end{pmatrix}^t = \begin{pmatrix} 0 & -2 & 0 \\ 3 & -1 & 4 \end{pmatrix}$$

Así, `traspuestaR [[0,3],[-2,-1],[0,4]] == [[0,-2,0],[3,-1,4]]`.