Programación Declarativa
Ingeniería en Informática

13 de septiembre de 2012

Apellidos: .... **Nombre:** .....

Instrucciones

- Resuelve el examen en dos archivos que tengan por nombre `examen-DNI-NOMBRE.hs` y `examen-DNI-NOMBRE.pl`, sustituyendo `DNI` por tu número de dni o pasaporte y `NOMBRE` por tus apellidos y nombre (separados por guiones).
- Escribe también lo siguiente en las primeras líneas de esos archivos: dni, apellidos y nombre, nombre del ordenador desde el que estás realizando el examen.

NOTAS:

- Salvo para el ejercicio 1 y el ejercicio 2, es **obligatorio** especificar, de la forma más general posible, el tipo de las *funciones solicitadas*. Para las demás funciones auxiliares que se definan no es necesario.
- Cuando el enunciado de un ejercicio indica que se reciben dos listas de la misma longitud, **no es necesario tratar** el caso de recibir dos listas de distinta longitud, es decir, se supone que la longitud va a ser la misma.

La *distancia de Hamming*, $dist(x, y)$, entre dos listas x e y de la misma longitud es el número de posiciones en las que esas dos listas contienen elementos distintos. Por ejemplo, la distancia de Hamming entre dos listas vacías es 0, y la distancia de Hamming entre las listas [2,4,3] y [3,4,4] es 2 (ya que ambas listas difieren en 2 posiciones, la primera y la tercera).

Ejercicio 1
Definir, **por recursión sobre listas**, una función `distanciaHamming` que recibe dos listas de la misma longitud y devuelve la distancia de Hamming entre esas dos listas.

La *distancia mínima*, $dist(C)$, de un conjunto C de listas de la misma longitud es la mínima distancia de Hamming entre cualesquiera dos listas distintas de C.

Si el valor de $dist(C)$ está fijado, ¿cuál es la mayor cantidad de listas que puede contener C? Se puede demostrar que si C es un conjunto de listas de ceros y unos de longitud n y $dist(C) \geq \delta n$, entonces $|C| \leq n2^{(1-H(\tau))n}$, donde $\tau = \frac{1-\sqrt{1-2\delta}}{2}$ y $H(x) = -x\log_2 x - (1-x)\log_2(1-x)$.
Por ejemplo, para $n = 4$ y $\delta = 0.5$ la cota que se obtiene es 4.0, mientras que para $n = 8$ y $\delta = 0.25$ la cota que se obtiene es 73.16043289187104.

Ejercicio 2
Definir una función `cotaCardinalC` que recibe los valores n y δ y devuelve el valor de la cota que se obtiene según la fórmula anterior. **Se debe** utilizar la cláusula *where* de manera adecuada para evitar repetir cálculos y para descomponer la fórmula en expresiones simples.
Nota: considérense las funciones $(**)$, **sqrt** y **logBase** de Haskell.

Consideremos el alfabeto q-ario $A = \{0, 1, \ldots, q-1\}$, con $q \geq 2$. La distancia de Lee entre dos listas $x = [x_1, x_2, \ldots x_n]$ e $y = [y_1, y_2, \ldots y_n]$ de la misma longitud sobre el alfabeto A se define como

$$dist(x, y) = \sum_{i=1}^{n} \text{mín}(|x_i - y_i|, q - |x_i - y_i|)$$

Problema: definir una función que recibe el tamaño q del alfabeto A y dos listas de la misma longitud sobre ese alfabeto y devuelve la distancia de Lee entre esas dos listas.
Por ejemplo, si $q = 5$, la distancia de Lee entre las listas [3,1,4,0] y [2,4,4,3] es $1 + 2 + 0 + 2 = 5$.

Ejercicio 3
Resolver el problema de la distancia de Lee utilizando funciones recursivas.

Ejercicio 4
Resolver el problema de la distancia de Lee utilizando listas por comprensión.

Ejercicio 5
Resolver el problema de la distancia de Lee utilizando funciones de procesamiento de listas (**map**, **filter**, **foldl**, ...).

El cuatro en raya es un juego de dos jugadores que se juega en un tablero con seis filas y siete columnas. Un jugador dispone de 21 fichas blancas y el otro de 21 fichas negras que dejan caer una a una, por turno, en las columnas del tablero. Gana quien consiga realizar primero una línea, horizontal, vertical o diagonal, de cuatro fichas de su color.

Ejercicio 6
Definir el tipo de datos `Ficha` como un sinónimo de un valor lógico. Definir el tipo de datos `Columna` como un sinónimo de una lista de fichas. Este tipo de datos representa las fichas colocadas en una de las columnas del tablero.

Definir un nuevo tipo de datos `CuatroEnRaya` que tenga un único constructor con un argumento, una lista de columnas. Este tipo de datos representa la configuración actual del tablero. Siempre asumiremos que al construir un valor de este tipo de datos la lista de columnas proporcionada es correcta.

Definir la función `colocarFicha` que recibe una Ficha, un **Int** (la columna) y un tablero CuatroEnRaya y devuelve el tablero CuatroEnRaya resultante de colocar la ficha en la columna especificada. Siempre asumiremos que la ficha, la columna y la configuración actual del tablero recibidas son correctas.

Por ejemplo, dada la configuración inicial del tablero, [[],[],[],[],[],[],[]], si el jugador que inicia la partida juega con fichas `True` y decide colocar en la tercera columna, el tablero quedaría [[],[],[True],[],[],[],[]]. Si tras algunas jugadas el tablero se encuentra en una configuración [[False,False,False],[],[True],[],[True],[True],[]] y el jugador que juega con `True` decide poner en la primera columna, el tablero debería quedar con [[True,False,False,False],[],[True],[],[True],[True],[]].

Consideremos la siguiente definición de un nuevo tipo de datos que representa de manera recursiva las colas con prioridad de números enteros:

```
type Prioridad = Int
data Cola a = ColaVacia | Encolar a Prioridad (Cola a)
            deriving Show
```

Es decir, una cola con prioridad de números enteros es la cola vacía o la cola obtenida añadiendo un número entero con una cierta prioridad (siempre de tipo **Int**) a una cola ya existente.

Ejercicio 7
Definir la función `primerElemento` que recibe una Cola y devuelve un par con el primer elemento de la cola (es decir, de todos los elementos de la cola con la máxima prioridad, aquel que se introdujo primero) y su prioridad. La función no estará definida para la cola vacía.

Así, por ejemplo, el primer elemento de la cola resultante de encolar el elemento 4 con prioridad 5 en la cola vacía, devolvería el par compuesto por el elemento 4 y la prioridad 5. En el caso de que tuviéramos como entrada la cola resultante de encolar 3 con prioridad 10 en la cola resultante de encolar 6 con prioridad 8 en

la cola resultante de encolar 9 con prioridad 10 en la cola vacía, devolvería el par compuesto por el elemento 9 y la prioridad 10.

Según la pseudociencia de la numerología, para calcular el número de nacimiento de una persona basta sumar los números que componen su fecha de nacimiento, volver a hacerlo con el resultado obtenido y así reiteradamente hasta obtener un solo guarismo entre el uno y el nueve.

Ejercicio 8
Definir la función `númeroNacimiento` que pida desde el teclado la fecha de nacimiento de una persona, en el formato *ddmmaaaa*, y escriba en pantalla el número de nacimiento que le corresponde.

Por ejemplo, el número de nacimiento que corresponde a la fecha 10061981 es el 8.

Ejercicio 9
Se considera el siguiente programa escrito en Prolog:

```
p(X) :- q(2, X).                    %R1
q(Y, X) :- Z is Y**2, Z > X, !.     %R2
q(Y, X) :- Z is mod(X, Y), Z =:= 0. %R3
q(Y, X) :- Z is Y + 1, q(Z, X).     %R4
```

1. Escribe el árbol de resolución que se obtendría para el programa anterior y la consulta `?- p(6).`
2. ¿Cómo habría que modificar la regla R3 para que el predicado `p` se correspondiera con la propiedad *ser primo* de los números enteros?

Ejercicio 10

1. Definir en Prolog un predicado `distanciaHamming(+L1,+L2,-N)` que calcule la distancia de Hamming entre dos listas, según se ha definido en el ejercicio 1.
2. Haciendo uso de los predicados predefinidos findall y min_list(+L, −M), definir en Prolog un predicado `mínimaDistancia(+C, -M)` que calcule la mínima distancia de una lista de listas, según se ha definido en el ejercicio 2.

Por ejemplo, la mínima distancia de la lista de listas [[1,1,0,0,0], [1,0,0,1,0], [0,0,1,0,1]] es 2, ya que *dist*([1,1,0,0,0], [1,0,0,1,0]) = 2, *dist*([1,1,0,0,0], [0,0,1,0,1]) = 4 y *dist*([1,0,0,1,0], [0,0,1,0,1]) = 4.