Programación Declarativa
Ingeniería en Informática

1 de febrero 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 ese archivo: dni, apellidos y nombre, nombre del ordenador desde el que estás realizando el examen.

NOTA: 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.

Ejercicio 1
Una *ecuación de tercer grado* con una incógnita y con coeficientes reales es una ecuación de la forma $ax^3+bx^2+cx+d=0$, donde a, b, c y d son números reales, con $a \neq 0$. Dividiendo la ecuación por el coeficiente a, se obtiene una nueva ecuación simplificada $x^3 + b'x^2 + c'x + d' = 0$.

Se define entonces el discriminante de la ecuación como

$$\Delta = 4p^3 + 27q^2, \qquad \text{donde} \qquad p = c' - \frac{b'^2}{3} \qquad \text{y} \qquad q = \frac{2b'^3}{27} - \frac{b'c'}{3} + d'$$

Por ejemplo, para la ecuación $27x^3 + 27x^2 - 396x - 242 = 0$ el discriminante es $\Delta = -13068$.

Definir una función `discriminante` que recibe los coeficientes a, b, c y d de una ecuación de tercer grado y devuelve el discriminante de esa ecuación. **Se debe** utilizar la cláusula *where* de manera adecuada para evitar repetir cálculos y para descomponer la fórmula en expresiones simples.

Ejercicio 2
Definir, **por recursión sobre listas**, una función `múltiplos` que recibe un número entero y una lista de números enteros y devuelve una lista con los elementos de la lista recibida que sean múltiplos del número recibido.

El dígito de control de una serie de Códigos de Barra numéricos se calcula como sigue: se suman los dígitos del código, teniendo en cuenta que los que ocupan posición par hay que multiplicarlos antes por 3 y se toma el último dígito (que se calcula tomando el resto de dividir entre 10) del resultado. Por ejemplo, el dígito de control del código 978038795475 es 8, ya que:

$$9 + (7 \times 3) + 8 + (0 \times 3) + 3 + (8 \times 3) + 7 + (9 \times 3) + 5 + (4 \times 3) + 7 + (5 \times 3) = 138$$

Problema: definir una función que dada una lista con los dígitos de un código de barras devuelva el dígito de control correspondiente.

Ejercicio 3
Resolver el problema del dígito de control utilizando funciones recursivas.

Ejercicio 4
Resolver el problema del dígito de control utilizando listas por comprensión.

Ejercicio 5
Resolver el problema del dígito de control utilizando funciones de procesamiento de listas (**map**, **filter** , **foldl**, ...).

Ejercicio 6
El *puzle de la serpiente* consta de una cuadrícula $n \times n$ (donde las casillas tienen coordenadas (x, y) según la representación habitual de los ejes de coordenadas). Sobre esta cuadrícula se desplaza una serpiente de m segmentos que ocupan casillas adyacentes desde la cabeza hasta la cola. La cabeza de la serpiente puede desplazarse a cualquier casilla adyacente en horizontal o vertical, arrastrando consigo el resto de segmentos y con las únicas restricciones de no poder atravesar los bordes de la cuadrícula y no poder superponerse a otros segmentos de la serpiente.

Definir el tipo de datos `Serpiente` como un sinónimo de una lista de tuplas de dos números de tipo **Int**. Este tipo de datos representa las casillas ocupadas por la serpiente, desde la cabeza hasta la cola. Definir el tipo de datos `Movimiento` como un sinónimo de cadena.

Definir un nuevo tipo de datos `PuzleSerpiente` que tenga un único constructor con dos argumentos, uno de tipo Serpiente y otro de tipo **Int**. Este tipo de datos representa la configuración actual del puzle, es decir, las casillas ocupadas por la serpiente y el número de movimientos realizados. Siempre asumiremos que al construir un valor de este tipo de datos la lista de casillas ocupadas por la serpiente y el número de movimientos son correctos.

Definir la función `moverSerpiente` que reciba un puzleSerpiente y un Movimiento y devuelva el puzleSerpiente resultante de mover la serpiente en la dirección especificada. Siempre asumiremos que la configuración actual del puzle recibida es correcta, que el movimiento recibido es una de las cadenas "Arriba", "Abajo", "Izquierda" o "Derecha", y que el movimiento se puede realizar.

Ejercicio 7
Consideremos la siguiente definición de un nuevo tipo de datos que representa de manera recursiva los polinomios con coeficientes enteros:

```
type Termino a = (a, a)
data Polinomio a = PolCero | Pol (Termino a) (Polinomio a)
                   deriving Show
```

Es decir, un polinomio con coeficientes enteros es el polinomio cero ($0x^0$) o un polinomio obtenido añadiendo un nuevo término con un cierto coeficiente entero y un cierto grado a un polinomio ya existente. Siempre asumiremos que no añadimos un término de grado igual a uno del polinomio ya existente, con la única excepción de que el término sea de grado 0, que se podrá añadir si el único término de grado 0 del polinomio ya existente es $0x^0$. Por otra parte, los términos no tienen por qué añadirse en orden creciente de grado.

Definir la función `derivada` que reciba un Polinomio y devuelva la derivada de ese polinomio. Siempre asumiremos que el polinomio recibido está construido de manera correcta.

Ejercicio 8
Para obtener la letra del Documento Nacional de Identidad basta calcular el resto de dividir el número por 23 y usar la siguiente tabla de asignación:

0 - T	1 - R	2 - W	3 - A	4 - G	5 - M	6 - Y	7 - F
8 - P	9 - D	10 - X	11 - B	12 - N	13 - J	14 - Z	15 - S
6 - Q	17 - V	18 - H	19 - L	20 - C	21 - K	22 - E	

Definir la función `letraDNI` que pida desde el teclado un número de DNI y escriba en pantalla la letra que le corresponde.

Ejercicio 9
Se considera el programa lógico:

```
p([X],X).                                  %R1
p([X,Y|L],Z) :- X =< Y,!,p([Y|L],Z).       %R2
p([X,Y|L],Z) :- p([X|L],Z).                %R3
```

Escribe el árbol de resolución y las respuestas para el programa anterior y la consulta `?- p([2,6,4],N).`

Ejercicio 10
Definir en Prolog un predicado `multiplos(+L1,+N,-L2)` que resuelva el ejercicio 2 anterior. Por ejemplo,

```
?- multiplos([1,2,3,4,5,6],3,S).
   S = [3,6]
```

1. Da una definición *recursiva*.
2. Da una definición usando el predefinido findall.