Programación declarativa
18 de Diciembre de 2013

Apellidos : .....

Nombre : .....

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

Ejercicio 1. [**2 ptos.**]

1. Definir las siguientes funciones usando las funciones predefinidas de orden superior y procesamiento de listas más adecuadas.

   - `par_orden`, que dada una lista de pares de números, devuelve la lista de los pares en los que la primera componente es menor que la segunda. Ejemplo:

     ```
     Main> par_orden [(8,65),(67,9),(98,100),(876,654)]
     [(8,65),(98,100)]
     ```

   - `restos`, que dada una lista de números y un número, devuelve la lista de los restos de dividir cada elemento de la lista por el número dado. Ejemplo:

     ```
     Main> restos [9,8,7,6] 2
     [1,0,1,0]
     ```

2. Definir usando el patrón de plegado por la derecha la función `longitudes`, que dada una lista de listas, devuelve la lista de sus longitudes. Ejemplo:

   ```
   Main> longitudes [[1,2,3],[4,3],[7,6,5,7],[8,7]]
   [3,2,4,2]
   ```

3. Definir utilizando el patrón de recursión con acumulador y el correspondiente plegado la función `por`, que dada una lista y un predicado, devuelva el producto de los números pares (es decir, múltiplos de 2) de la lista que cumplen el predicado

   ```
   Main> por [2,4,6,7,5,3,8,40,80,65] (>32)
   3200
   ```

Ejercicio 2. [**2 ptos.**]
Consideremos el siguiente tipo, definido para manejar las **pilas de datos**

```
data Pila a = Vacia
            | Apila a (Pila a)
              deriving Eq

instance (Show a) => Show (Pila a) where
  show Vacia = "-"
  show (Apila x s) = show x ++ "|" ++ (show s)
```

Definir los procedimientos siguientes:

- `esVacia` que determina si una pila de datos está vacía.
- `desapila` como la función que, dada una pila de datos, devuelve la pila que resulta al eliminar el último dato incorporado a la misma. Si la pila no contiene datos debe indicarse con un mensaje de error adecuado.
- `cima` como la función que dada una pila devuelve el último dato incorporado a la misma. Si la pila es vacía debe mostrarse un mensaje de error con la advertencia correspondiente.

Ejercicio 3. [**3 ptos.**]

- Definir la función

  ```
  filtraPila :: (a -> Bool) -> Pila a -> Pila a
  ```

  tal que `(filtraPila p pila)` es la pila con los elementos de pila que verifican el predicado `p`, en el mismo orden. Por ejemplo,

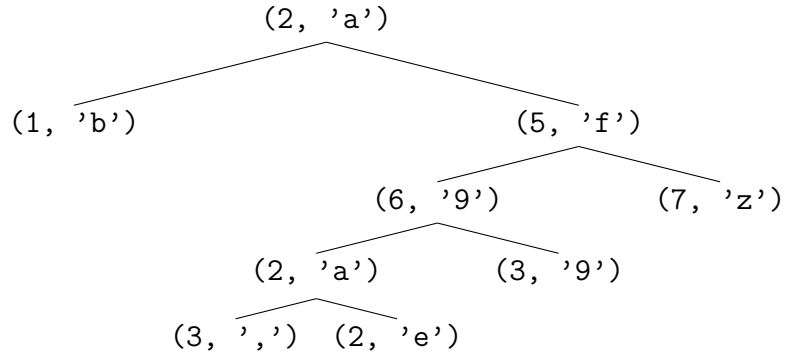
  ```
  Main> p1
  1|2|3|4|5|6|7|8|9|10|11|12|13|14|15|16|17|18|19|20|-
  Main> filtraPila even p1
  2|4|6|8|10|12|14|16|18|20|-
  ```

- Definir adecuadamente `p1` para comprobar el ejemplo anterior.

Ejercicio 4. [**3 ptos.**]

- Definir el tipo de datos `Info` para representar los pares de enteros y caracteres.
- Definir el tipo de datos `Arbol` para representar, adecuadamente, árboles binarios que, tanto en los nodos como en las hojas, contengan datos de un tipo arbitrario `a`.
- Definir la función `altura`, que calcule la altura de un árbol definido según el tipo anterior.

  Si `a1` es el siguiente árbol, entonces `altura a1 == 4`



- Definir adecuadamente `a1` para comprobar el ejemplo anterior.