Ejercicios de Programación Declarativa

Curso 2021/22

Hoja 4

1. Supongamos que definimos el tipo `data Pila = P [a]` para representar pilas como instancia de las clases `Show` y `Eq`, heredando los métodos por defecto. Define las siguientes funciones en Haskell:

   - `creaPila` para crear una pila vacía.
   - `esPilaVacia p` para determinar si la pila `p` está vacía o no.
   - `apilar x p` para apilar un elemento `x` en la pila `p`.
   - `cima p` para consultar la cima de una pila no vacía `p`.
   - `desapilar p` para eliminar la cima de una pila no vacía `p`.

   Determina el significado de la siguiente definición:

   ```
   r :: [a] -> [a]
   r xs = ys where P ys = foldl (\p x -> apilar x p) creaPila xs
   ```

2. Define una función `mayorQueCumple::Integral a => (a->Bool)->a->a->Maybe a`, que dada una propiedad `p` y dos números enteros `n` y `m` devuelva devuelva el mayor entero entre `n` y `m` que cumpla la propiedad `p`. Devuelve `Nothing` en el caso de que ninguno la cumpla.

3. Define un tipo de datos estructurado `Cj` para representar conjuntos de elementos del mismo tipo. Recuerda que en un conjunto no puede haber elementos repetidos y que el orden de los elementos no importa. Define las siguientes funciones:

   - `elemCj x c` para determinar si `x` es un elemento del conjunto `c` o no.
   - `bienDefCj c` para eliminar las posibles repeticiones de elementos en el conjunto `c`.
   - `contenido c1 c2` tras eliminar las posibles repeticiones de los conjuntos `c1` y `c2`, determina si `c1` está contenido en `c2`, es decir todos los elementos de `c1` son elementos de `c2`.

   Declara el tipo `Cj` como instancia de las clases `Eq` y `Ord`, teniendo en cuenta que un conjunto `c1` es menor que otro `c2` si `c1` está contenido en `c2`, y son iguales si `c1` está contenido en `c2` y `c2` está contenido en `c1`.

4. Dada la declaración:

   ```
   data Temp = Kelvin Float | Celsius Float | Fahrenheit Float
   ```

   para representar temperaturas en diferentes escalas, escribe una función para realizar conversiones de una escala a otra y otra para determinar la escala en la que está representada una temperatura. El nuevo tipo tiene que ser instancia de las clases `Ord` y `Eq`. Define adecuadamente los métodos `==` y `compare` para la nueva estructura de datos.

5. Declara un tipo de datos para representar árboles binarios de búsqueda con valores en los nodos pero no en las hojas. Programa en Haskell la ordenación de una lista por el algoritmo `treeSort`, consistente en ir colocando uno a uno los elementos de la lista en un árbol binario de búsqueda inicialmente vacío. A continuación devuelve la lista resultante de recorrer el árbol en *inOrden*.