Haskell — Unfoldr P23089_ca

El mòdul Data.List de Haskell ofereix una funció **unfoldr** :: $(b \to \textbf{Maybe}\ (a, b)) \to b \to [a]$ que és un *dual* de **foldr**. Aquesta és la seva documentació:

While **foldr** reduces a list to a summary value, **unfoldr** builds a list from a seed value. The function takes the element and returns **Nothing** if it is done producing the list or returns **Just** (m, n), in which case, m is prepended to the list and n is used as the next element in a recursive call.

1. Definiu recursivament una funció *myUnfoldr* :: $(b \to \textbf{Maybe}\ (a, b)) \to b \to [a]$ que funcioni com **unfoldr**.

   Si no us en sortiu, podeu fer la resta dels apartats fent *myUnfoldr* = **unfoldr** i incloent un `import Data.List (unfoldr)` al principi del programa.

2. Definiu, utilitzant *myUnfoldr*, una funció *myReplicate* :: $a \to \textbf{Int} \to [a]$ de manera que *myReplicate x n* retorni una llista amb n cops el valor x.

3. Definiu, utilitzant *myUnfoldr*, una funció *myIterate* :: $(a \to a) \to a \to [a]$ que funcioni com **iterate**.

4. Definiu, utilitzant *myUnfoldr*, una funció *myMap* :: $(a \to b) \to [a] \to [b]$ que funcioni com **map**

5. Considereu la definició següent del tipus *Bst* per arbres binaris de cerca, juntament amb una funció *add* que hi afegeix valors:

   data *Bst a* = *Empty* | *Node a* (*Bst a*) (*Bst a*) **deriving Show**

   add :: **Ord** $a \Rightarrow a \to$ (*Bst a*) $\to$ (*Bst a*)

   add x Empty = *Node x Empty Empty*
   add x (*Node y l r*)
   | $x < y$ = *Node y* (*add x l*) *r*
   | $x > y$ = *Node y l* (*add x r*)
   | **otherwise** = *Node y l r*

   Feu que els arbres binaris de cerca siguin instància de **Show**, mostrant-se segons els exemples.

6. Definiu una funció *adder* :: **Ord** $a \Rightarrow$ (*Bst a*, $[a]$) $\to$ **Maybe** (*Bst a*, (*Bst a*, $[a]$)) de manera que *myUnfoldr adder* (t, xs) retorni una llista que mostri, pas a pas, la construcció d'un arbre binari de cerca inserint seqüencialment els valors de xs en t. Vegeu l'exemple.

El Jutge dóna puntuacions parcials, 15 punts per apartat i 10 per l'exemple públic.

Observació

A l'hora de corregir es tindrà en compte la correcció, senzillesa, elegància i eficiència de la solució proposada.

Exemple d'entrada

```
myUnfoldr (\x -> if x == 0 then Nothing else Just (x, x - 1)) 5
myReplicate 7 4
myReplicate '*' 4
take 8 $ myIterate (*2) 1
take 4 $ myIterate ('*' :) ""
myMap (*2) [1..10]
take 4 $ myMap even [1..]
show (Empty :: Bst Int)
show $ add 30 Empty
show $ add 20 $ add 10 $ add 50 $ add 30 Empty
myUnfoldr adder (Empty, [3, 1, 4, 5])
```

Exemple de sortida

```
[5,4,3,2,1]
[7,7,7,7]
"****"
[1,2,4,8,16,32,64,128]
["","*","**","***"]
[2,4,6,8,10,12,14,16,18,20]
[False,True,False,True]
"."
"(30 . .)"
"(30 (10 . (20 . .)) (50 . .))"
[(3 . .),(3 (1 . .) .),(3 (1 . .) (4 . .)),(3 (1 . .) (4 . (5 . .)))]
```

Informació del problema

Autor : Jordi Petit
Generació : 2022-11-04 14:41:58