Haskell — Parcial 2019-11-06 P24239_es

1. Números romanos (con recursividad)

Define una función *roman2int* :: **String** → **Int** que convierta un número romano en su entero equivalente *usando recursividad*.

Recuerda que los números romanos se escriben con los símbolos `I`, `V`, `X`, `L`, `C`, `D` y `M`, con valores 1, 5, 10, 50, 100, 500 y 1000 respectivamente. En este sistema, para obtener el número representado, se suman los valores de los símbolos, excepto los símbolos situados a la izquierda de un símbolo de valor mayor, que se restan.

2. Números romanos (sin recursividad)

Define una función *roman2int'* :: **String** → **Int** que hace lo mismo que la función anterior pero *sin usar recursividad*: usa una o más funciones de orden superior.

3. Raíces

La serie de Taylor para calcular $\sqrt{x}$ es:

$$f_1(x) = x$$
$$f_n(x) = \frac{1}{2}\left(f_{n-1}(x) + \frac{x}{f_{n-1}(x)}\right)$$

Define una función *arrels* :: **Float** → [**Float**] que, dado un real x, retorna la lista infinita de los términos del desarrollo de Taylor de $\sqrt{x}$.

4. Más raíces

Escribe una función *arrel* :: **Float** → **Float** → **Float** que a partir de una x y un ϵ, aproxime la raíz de x con un error inferior o igual a ϵ utilizando la lista infinita anterior. El error en el término t_i de la serie (con $i > 1$) es la diferencia en valor absoluto entre t_i y t_{i-1}.

5. Escritura de árboles

Considera el siguiente tipo genérico *LTree a* de árboles binarios con valores en las hojas:

data *LTree a* = *Leaf a* | Node (LTree a) (LTree a)

Haz que los árboles sean ("instance") de la clase **Show** visualizándolos según los ejemplos.

6. Creación de árboles equilibrados

Haz una función *build* :: [*a*] → *LTree a* que, dada una lista no vacía, construye el *LTree* equilibrado (a la izquierda) que contiene los elementos de la lista en el mismo orden de izquierda a derecha. Decimos que un árbol está equilibrado a la izquierda si todos los subárboles tienen el hijo izquierdo con la misma profundidad que el hijo derecho o la misma más 1.

7. Mónadas y árboles

Define una función *zipLTrees* :: *LTree a* → *LTree b* → **Maybe** (*LTree* (*a*,*b*)) que combine los valores de las hojas de dos árboles con la misma estructura.

Si las estructuras de los dos árboles no encajan, retorna **Nothing** y, si encajan, retorna **Just** del árbol que tiene en cada hoja el par con el primer elemento del primer árbol y el segundo del segundo árbol en la misma posición.

Utiliza la notación **do**.

Ejemplo de entrada 1

```
roman2int "I"
roman2int "IV"
roman2int "MCCCXIX"
roman2int "MMXVIII"
```

Ejemplo de salida 1

```
1
4
1319
2018
```

Ejemplo de entrada 2

```
roman2int' "I"
roman2int' "IV"
roman2int' "MCCCXIX"
roman2int' "MMXVIII"
```

Ejemplo de salida 2

```
1
4
1319
2018
```

Ejemplo de entrada 3

```
take 10 $ arrels 4.0
take 10 $ arrels 100.0
```

Ejemplo de salida 3

```
[4.0,2.5,2.05,2.0006099,2.0,2.0,2.0,2.0,2.0,2.0]
[100.0,50.5,26.240099,15.02553,10.840435,10.032578,10.000053,10.0,10.0,10.0]
```

Ejemplo de entrada 4

```
arrel 4.0 0.00001
arrel 100.0 0.1
```

Ejemplo de salida 4

```
2.0
10.000053
```

Ejemplo de entrada 5

```
Node (Leaf 3) (Node (Leaf 8) (Leaf 7))
Node (Leaf 1) (Node (Node (Leaf 3) (Leaf 4)) (Node (Leaf 8) (Leaf 7)))
Node (Leaf "Albert") (Node (Leaf "Gerard") (Leaf "Jordi"))
Leaf 'x'
```

Ejemplo de salida 5

```
<{3},<{8},{7}>>
<{1},<<{3},{4}>,<{8},{7}>>>
<{"Albert"},<{"Gerard"},{"Jordi"}>>
{'x'}
```

Ejemplo de entrada 6

```
build [3, 2, 5]
build [3, 2, 8, 5, 1]
build ['a', 'b', 'c', 'd']
build [[1, 2, 3]]
```

Ejemplo de salida 6

```
<<{3},{2}>,{5}>
<<<{3},{2}>,{8}>,<{5},{1}>>
<<{'a'},{'b'}>,<{'c'},{'d'}>>
{[1,2,3]}
```

Ejemplo de entrada 7

```
let t1 = Node (Leaf "a") (Node (Leaf "b") (Leaf "c"))
let t2 = Node (Leaf 0) (Node (Leaf 1) (Leaf 2))
let t3 = Node (Node (Leaf 1) (Leaf 2)) (Leaf 0)
zipLTrees t1 t2
zipLTrees t1 t3
```

Ejemplo de salida 7

```
Just <{("a",0)},<{("b",1)},{("c",2)}>>
Nothing
```

Información del problema

Autoría: Jordi Petit, Albert Rubio, Gerard Escudero
Traducción: Albert Rubio

Generación: 2026-02-03T17:06:35.926Z