Haskell — Parcial 2018-04-11 P91910_ca

Apartat 1: Llista infinita

Escriviu una funció *multEq* :: **Int** → **Int** → [**Int**] que, donats dos nombres positius x i y diferents de zero, genera la llista infinita ordenada creixentment que conté els nombres formats per la multiplicació de la mateixa quantitat de x que de y.

Apartat 2: Selecció

Escriviu una funció *selectFirst* :: [**Int**] → [**Int**] → [**Int**] → [**Int**] que, donades tres llistes *l1*, *l2* i *l3* retona els elements de *l1* que apareixen a *l2* en una posició menor estrictament que a *l3*. Si un element apareix a *l2* i no a *l3* es considera que apareix en una posició anterior.

Apartat 3: *iterate* amb *scanl*

Definiu una funció *myIterate* :: $(a \to a) \to a \to [a]$ que faci el mateix que **iterate**, però implementada en termes d'**scanl**.

Apartat 4: Taula de símbols

Considereu una taula de símbols genèrica que converteix textos (**String**s) en valors de tipus *a* definida per **type** *SymTab a* = **String** → **Maybe** *a*.

La taula de símbols retorna un **Maybe** *a* i no un *a* perquè poder indicar cerques sense èxit.

Les operacions sobre la taula de símbols són:

empty :: *SymTab a*
get :: *SymTab a* → **String** → **Maybe** *a*
set :: *SymTab a* → **String** → *a* → *SymTab a*

on *empty* crea una taula de símbols buida, *get* retorna el valor d'un text a la taula de símbols (amb **Just** si hi és o **Nothing** si no hi és), i *set* retorna una nova taula de símbols definint un nou valor per a un símbol (i sobrescrivint el valor antic si el símbol ja era a la taula).

Implementeu aquestes tres operacions sobre el **type** donat (que no podeu canviar).

Apartat 5: Expressions amb símbols

Considereu el següent tipus genèric per a expressions de tipus *a* amb variables:

```
data Expr a
  = Val a
  | Var String
  | Sum (Expr a) (Expr a)
  | Sub (Expr a) (Expr a)
```

| *Mul* (*Expr a*) (*Expr a*)
deriving Show

Escriviu una funció *eval* :: (**Num** *a*) ⇒ *SymTab a* → *Expr a* → **Maybe** *a* que, evaluï una expressió utilitzant una taula de símbols, retornant **Nothing** si alguna variables no està definida a la taula.

Exemple d'entrada 1

```
take 6 $ multEq 2 3
take 5 $ multEq 3 7
```

Exemple de sortida 1

```
[1,6,36,216,1296,7776]
[1,21,441,9261,194481]
```

Exemple d'entrada 2

```
selectFirst [] [] []
selectFirst [8,4,5,6,12,1] [] [8,6,5,4,1]
selectFirst [8,4,5,6,12,1] [4,5,6,2,8,12] []
selectFirst [8,4,5,6,12,1] [4,5,6,2,8,12] [8,6,5,4,1]
```

Exemple de sortida 2

```
[]
[]
[8,4,5,6,12]
[4,5,12]
```

Exemple d'entrada 3

```
take 10 $ myIterate (+1) 0
take 10 $ myIterate (*2) 1
take 10 $ myIterate ('a':) []
take 8 $ myIterate (++"y") "x"
```

Exemple de sortida 3

```
[0,1,2,3,4,5,6,7,8,9]
[1,2,4,8,16,32,64,128,256,512]
["","a","aa","aaa","aaaa","aaaaa","aaaaaa","aaaaaaa","aaaaaaaa","aaaaaaaaa"]
["x","xy","xyy","xyyy","xyyyy","xyyyyy","xyyyyyy","xyyyyyyy"]
```

Exemple d'entrada 4

```
get (set empty "a" 1) "a"
get (set empty "a" 1) "b"
get (set (set empty "a" 1) "b" 2) "a"
get (set (set empty "a" 1) "b" 2) "b"
get (set (set empty "a" 1) "b" 2) "c"
get (set (set empty "a" 1) "a" 2) "a"
```

Exemple de sortida 4

```
Just 1
Nothing
Just 1
Just 2
Nothing
Just 2
```

Exemple d'entrada 5

```
let st1 = set (set empty "a" 1) "b" 2
let st2 = set (set empty "a" 4) "b" 3
let e1 = Mul (Val 5) (Sum (Var "a") (Var "b"))
let e2 = Mul (Val 5) (Sum (Var "a") (Var "c"))
let e3 = Sub (Var "a") (Var "b")
eval st1 e1
eval st2 e1
eval st1 e2
eval st2 e2
eval st1 e3
eval st2 e3
```

Exemple de sortida 5

```
Just 15
Just 35
Nothing
Nothing
Just (-1)
Just 1
```

Informació del problema

Autor : Jordi Petit i Albert Rubio
Generació : 2024-05-03 08:51:05