Haskell — Expressions lògiques P41359_ca

Considereu aquesta definició de tipus per a expressions lògiques amb variables o valors booleans a les fulles:

```
data LogicExpression
  = Or  LogicExpression LogicExpression
  | And LogicExpression LogicExpression
  | Not LogicExpression
  | Var String
  | Val Bool
```

1. Feu que *LogicExpression* sigui instància de **Show** de forma que les expressions compostes apareguin totalment parentitzades en forma infixa, tot seguint el format dels exemples.

2. Feu una funció *pushNegations* :: *LogicExpression* → *LogicExpression* que, donada una expressió lògica, retorni una expressió lògica equivalent en la que les negacions només apareguin aplicades a variables i que elimini les dobles negacions consecutives.

3. Escriviu un llista infinita *bits* :: [[[**Int**]]] que contingui totes les llistes d'n bits per a tota $n \geq 0$. No podeu fer servir recursivitat ni importar cap llibreria. Penseu en l'operador <∗> de les llistes considerades com a aplicatius. S'espera una solució extremadament concisa. (Aquest apartat és independent dels anteriors.)

El Jutge dóna puntacions parcials per cada apartat: un punt per cada joc de proves públic i un punt per cada joc de proves privat. La puntuació del problema per part del professor és independent d'aquesta puntuació.

Exemple d'entrada 1

```
Val False
Val True
Var "a"
And (Val True) (Var "b")
And (Val False) (Val False)
Or (And (Val True) (Var "x")) (Not (Or (Var "y") (Var "x")))
Not (Or (And (Val True) (Var "x")) (Not (Or (Var "y") (Var "x"))))
```

Exemple de sortida 1

```
0
1
a
(1 and b)
(0 and 0)
((1 and x) or (not (y or x)))
(not ((1 and x) or (not (y or x))))
```

Exemple d'entrada 2

```
pushNegations $ Val True
pushNegations $ Var "x"
pushNegations $ Not (Val True)
pushNegations $ And (Val True) (Var "x")
pushNegations $ Or (And (Val True) (Var "x")) (Not (Or (Var "y") (Var "x")))
pushNegations $ Not (Or (And (Val True) (Var "x")) (Not (Or (Var "y") (Var "x"))))
pushNegations $ Not $ Not $ Not $ Var "a"
```

Exemple de sortida 2

```
1
x
0
(1 and x)
((1 and x) or ((not y) and (not x)))
((0 or (not x)) and (y or x))
(not a)
```

Exemple d'entrada 3

```
take 3 bits
bits !! 0
bits !! 1
bits !! 2
bits !! 3
bits !! 4
take 16 $ map length bits
```

Exemple de sortida 3

```
[[[]],[[0],[1]],[[0,0],[0,1],[1,0],[1,1]]]
[[]]
[[0],[1]]
[[0,0],[0,1],[1,0],[1,1]]
[[0,0,0],[0,0,1],[0,1,0],[0,1,1],[1,0,0],[1,0,1],[1,1,0],[1,1,1]]
[[0,0,0,0],[0,0,0,1],[0,0,1,0],[0,0,1,1],[0,1,0,0],[0,1,0,1],[0,1,1,0],[0,1,1,1],[1,0,0,0],[1,0,
[1,2,4,8,16,32,64,128,256,512,1024,2048,4096,8192,16384,32768]
```

Informació del problema

Autoria: Jordi Petit

Generació: 2026-02-03T17:01:52.271Z