Para os próximos exercícios, use a seguinte definição de `Arvore`:

```
data Arvore = Folha Int | Nodo Int Arvore Arvore
    deriving(Eq,Show)
```

(1) Implementar a função

`multArvore:: Int -> Arvore ->  Arvore`

que recebe um inteiro e uma árvore, e multiplica todos os valores contidos na árvore pelo inteiro

(2) Implemente a função

`contaFolhas :: Arvore -> Int`

que recebe uma árvore e conta quantas folhas existem nessa árvore

(3) Implemente a função

`contaNodos :: Árvore -> Int`

que conta quantos `Nodos` uma árvore possui

(4) Implemente a função:

`quantasVezes :: Int -> Arvore -> Int`

que recebe um inteiro e uma árvore, e conta quantas vezes esse inteiro aparece na árvore

(5) A função `max` do Haskell, recebe dois inteiros e devolve o maior entre eles:

```
Main*> max 4 33
33

Main*> max 10 10
10
```

Usando a função `max` implemente a função:

`maxArvore :: Arvore -> Int`

que encontra o maior inteiro em uma árvore

(6) Uma árvore refletida, é uma árvore com todos os seus ramos esquerdos e direitos trocados. Defina a função:

`refleteArvore :: Arvore -> Arvore`

(7) Implementar a função

`geraLista :: Arvore -> [Int]`

que transforma uma árvore em uma lista de inteiros. Não importa a ordem dos inteiros na lista, apenas que todos os inteiros dos Nodos e Folhas estejam na lista resultante