Projeto 2

ATE meia noite de 2/9

Compressão e descompressão de listas

dada a lista (de caracteres neste exemplo)

```
"aaabbaasxbbbb"
```

vamos definir uma lista comprimida cujos elementos são pares `(item, quantidade)` onde `quantidade` é o número de vezes que o `item` aparece sequenciamente na lista. Assim, a compressão dessa lista seria:

```
[('a',3),('b',2),('a',2),('s',1),('x',1),('b',4)]
```

- Implemente a função `comprime :: Eq a => [a] -> [(a,Int)]`

```
comprime [3,3,3,4,5,6,5,5,5,5,7]

=> [(3,3),(4,1),(5,1),(6,1),(5,4),(7,1)]
```

- Implemente a função `descomprime :: Eq a => [(a,Int)] -> [a]` que é o inverso de comprime

```
descomprime [(3,3),(4,1),(5,1),(6,1),(5,4),(7,1)]

==> [3,3,3,4,5,6,5,5,5,5,7]
```

Restrições

Nesse projeto voce pode usar qualquer função já predefinida no prelude do Haskell https://hackage.haskell.org/package/base-4.20.0.1/docs/Prelude.html#g:13 mas nao pode usar funções definidas nos modulos

Comentários

eu acho que nao é claro como usar programação de alto nivel (funções que operam em funções) no problema de comprimir. Acho que seria uma recursao tradicional. Na **minha cabeça** um `foldr` é mais claro nesse problema, mas vc pode resolver como quiser

O descomprime é muito mais próximo de uma abordagem usando programação de alto nivel. Cada elemento da lista comprimida , algo como `(5,4)` precisa ser transformado em `[5,5,5,5]`. Nesse primeiro passo vc obtem uma lista de listas. Mas veja o que a função `concat`, já definida no prelude faz:

```
ghci> concat [[1],[3,4,9],[],[5,6,7,10],[],[4]]
[1,3,4,9,5,6,7,10,4]
```