Universidade de Brasília
Departamento de Ciência da Computação

# Linguagens de Programação - Turma B - Lista Haskell 4

Prof. Marcos Fagundes Caetano - 2016/1

| Matrícula | Nome |
|---|---|
| | |

1. Utilizando *list comprehension*, forneça a expressão que calcule a soma $1^2 + 2^2 + \ldots + 100^2$.

2. Generalize a expressão anterior de tal forma que o calculo seja feito para os $n$ primeiros elementos. A função geral soma apresenta a seguinte definição: $soma \ :: \ Int \ \rightarrow Int$.

3. Em maneira similar a função *length*, mostre como a função $replicate \ :: \ Int \rightarrow a \rightarrow [a]$, que produz a lista de elementos idênticos, pode ser definida utilizando *list comprehension*. Exemplo:

```
> replicate 3 True
[True,True,True]
```

4. O trio $(x, y, z)$ de inteiros positivos é pitagórico (*pythagorean*) se $x^2 + y^2 = z^2$. Utilizando *list comprehension*, defina a função:

$$pyths \ :: \ Int \ \rightarrow \ [(Int, Int, Int)]$$

   Esta função mapeia um inteiro $n$ para todos os trios com componentes em $[1..n]$. Por exemplo:

```
> pyths 5
[(3,4,5),(4,3,5)]
```

5. Um inteiro positivo é *perfeito* se seu valor é igual a soma dos seus fatores, excluindo seu próprio valor. Utilizando *list comprehension* e a função $factors$, defina a função $perfects \ :: Int \rightarrow [Int]$, que retorna a lista de todos números perfeitos até um dado limite. Exemplo:

```
> perfects 500
[6,28,496]
```

   **Dica:** a função **init** ($init \ :: \ [a] \rightarrow [a]$) retorna a lista original sem o último elemento.

6. Mostre como a simples *comprehension* $[(x, y) \mid x \leftarrow [1, 2, 3], \ y \leftarrow [4, 5, 6]]$ com dois geradores pode ser rescrito utilizando duas *comprehension* com apenas um gerador.

```
> [(x,y) | x <- [1,2,3], y <- [4,5,6]]
[(1,4),(1,5),(1,6),(2,4),(2,5),(2,6),(3,4),(3,5),(3,6)]
```

   **Dica:** Utilize a função **concat** para unir as duas *comprehension*.

7. O produto escalar entre duas listas de inteiros $xs$ e $ys$ de tamanho $n$ é dada pela soma do produto entre os números inteiros correspondentes:

$$\sum_{i=0}^{n-1}(xs_i * ys_i) \tag{1}$$

Mostre como *list comprehension* pode ser utilizada para definir a função *scalarproduct* :: $[Int] \rightarrow [Int] \rightarrow Int$ que retorna o produto escalar entre duas listas. Por exemplo:

```
> scalarproduct [1,2,3][4,5,6]
32
```

**Obs:** 1*4 + 2*5 + 3*6 =¿ 32