# INE5416 - Paradigmas da Programação (2015/2)

***Relatório 9: Listas em Haskell***

**Caique Rodrigues Marques 13204303**

# Parte 1

- *Lazy evaluation* ou avaliação preguiçosa é uma técnica onde se avalia uma dada expressão apenas quando ela for necessária, evitando possíveis erros futuros ou computações desnecessárias. Esta técnica também acaba sendo útil para a construção de estruturas de dados infinitas, por exemplo usando `[1,2,..]` onde se vai alocando espaço até não ter mais memória. Em linguagens de programação funcionais, como Haskell e Miranda, implementam *lazy evaluation* por padrão, enquanto em outras linguagens existem formas para providenciar a técnica.

- Várias linguagens de programação implementam a função *map*, a função de mapeamento, que dada uma função, se uma lista de operações for mapeada para esta função, ela retorna uma lista com as respostas. Funções auxiliares podem ser usadas para complementar o mapeamento, como `list` e `filter`. Em Python temos os exemplos:

  - `list(map(lambda x:  x+1, range(10)))`: Mapeia uma função simples de soma para uma lista com elementos com inteiros de 0 a 9, a saída será uma lista com cada elemento somado a 1.
  - `list(map(chr, range(65, 91)))`: Função que mapeia cada caractere ao seu correspondente em ASCII e depois as listando as letras do alfabeto maiúsculas, de 'A' (65) a 'Z' (91).

- O módulo Data.List do Haskell provém uma série de operações a serem realizadas com listas, como por exemplo, coletar o primeiro elemento, coletar o tamanho da lista, coletar a lista reversa, concatenar listas, replicar listas e entre outras coisas.

# Parte 2

- Para o cálculo de soma de termos de uma progressão aritmética, primeiro uma simples função é encontrada para achar a razão. A fórmula de uma PA pode ser implementada com mais facilidade usando as funções nativas do Haskell, para coletar o tamanho da lista (o número de termos) e o operador `div` garante uma divisão inteira.

```
ratio n = n!!1 - head(n)
sum_ap n =
    ( (length n) * (2*head(n) + ((length n - 1) * ratio n)) ) `div` 2
```

- A implementação do cálculo de produto de termos de uma PA foi considerada que para um $n$ qualquer é tal que $n \in \mathbb{Z}^+$, logo, a determinação do valor na função gama é da seguinte forma:

$$P_n = r^n \frac{\Gamma(\frac{a_1}{r} + n)}{\Gamma(\frac{a_1}{r})} \qquad \Gamma(n) = (n-1)!$$

  Em Haskell é necessário especificar a função `fromIntegral` para converter um valor inteiro para um valor em ponto flutuante (`length` sempre retorna inteiro). Visto a limitação computacional, o cálculo do fatorial na função gama é dado de um valor máximo de 1000.

```
gamma n = sum([product[1..n-1]])
prod_ap n = (ratio n)** fromIntegral (length n) *
    ( (gamma ( (head(n) / ratio n) + fromIntegral (length n) ))
    / gamma (head(n) / ratio n) )
```