

# Universidade Federal de Lavras

GAC107 – Paradigmas de Linguagens de Programação

**Profa. Ana Paula Piovesan Melchiori**

## Exercício Prático 02

1. Defina transparência referencial, no contexto de linguagens de programação.
2. O que é uma recursão de cauda?
3. Cite formas funcionais utilizadas na confecção de funções mais complexas, a partir de funções primitivas fornecidas por uma linguagem funcional.
4. Escrever uma função que encontre as raízes reais de uma equação do 2º grau ( $a \cdot x^2 + b \cdot x + c = 0$ ).
5. Fornecidos três valores diferentes entre si,  $a$ ,  $b$  e  $c$ , elaborar uma função que retorne quantos desses três números são maiores que o valor médio entre eles.
6. Calcular a soma dos números inteiros compreendidos em um intervalo  $[x,y]$ , incluindo e excluindo os limites.
7. Construa uma função que retorne o MMC (Mínimo Múltiplo Comum) entre dois números inteiros.
8. Construa uma função que retorne o MDC (Máximo Divisor Comum) de uma lista de números inteiros.
9. Defina uma função recursiva que recebe dois inteiros  $m$  e  $n$ , onde  $m < n$ , e retorne o produto de todos os números no intervalo  $[m,n]$ .
10. A raiz quadrada inteira de um número inteiro positivo  $n$  é o maior número inteiro cujo quadrado é menor ou igual a  $n$ . Por exemplo, a raiz quadrada inteira de 15 é 3, e a raiz quadrada inteira de 16 é 4. Defina uma função recursiva para calcular a raiz quadrada inteira.
11. Elaborar uma função para concatenar duas listas, sem utilizar o operador de concatenação do Haskell (++).

12. Elabore uma função que recebe uma string e remova espaços repetidos: ao encontrar dois ou mais espaços seguidos, deixa apenas um.

13. Defina uma função que, dada uma lista de números reais *A*, retorne uma lista *B* com os elementos presentes em *A* que são menores do que a média de todos os elementos de *A*.

14. Dada uma base de dados de 10 alunos matriculados em uma disciplina, construa funções que retornem:

- a. número de reprovados na disciplina. Considere como aprovado o aluno que obteve uma nota acima ou igual a 6.0;
- b. o nome do(a) aluno(a) que obteve a menor nota.

```
type Nome = String
type Curso = String
type Período = Int
type Nota = Float
type Aluno = (Nome, Curso, Período, Nota)
type Disciplina = [Aluno]

alunos :: Int -> Aluno
alunos matricula
  | matricula == 1 = ("Rodrigo", "S.Inf.", 3, 6.0)
  | matricula == 2 = ("Joao", "Eng.Comp.", 5, 5.0)
  | matricula == 3 = ("Lucas", "C.Comp.", 8, 3.5)
  | matricula == 4 = ("Ana", "C.Comp.", 5, 8.0)
  | matricula == 5 = ("Maria", "C.Comp.", 7, 9.5)
  | matricula == 6 = ("Paulo", "C.Comp.", 6, 6.0)
  | matricula == 7 = ("Jose", "S.Inf.", 8, 7.0)
  | matricula == 8 = ("Eduarda", "C.Comp.", 4, 1.0)
  | matricula == 9 = ("Carla", "Eng.Comp.", 6, 6.5)
  | matricula == 10 = ("Luiz", "C.Comp.", 7, 5.7)
```

**Exemplo de uso:**

```
Main> contar_reprovados 10
4
Main> menor_nota 10
"Eduarda"
```

15. Crie uma função *zip3* com funcionamento similar à função *zip*, mas recebendo três listas ao invés de apenas duas.

16. Construa a função de Ackermann, a qual é definida por:

```
A(m, n) = n + 1 se m = 0
A(m, n) = A(m-1, 1) se m <> 0 e n = 0
A(m, n) = A(m-1, A(m, n-1)) se m <> 0 e n <> 0
```

<> significa diferente

17. Seja o cadastro de pessoas dado pela função a seguir:

```
type Pessoa = (String, Int, Float, Char)
pess :: Int->Pessoa
pess x
  | x==1 = ("Rosa", 27, 1.66, 'F')
  | x==2 = ("João", 26, 1.85, 'M')
  | x==3 = ("Maria", 67, 1.55, 'F')
  | x==4 = ("Jose", 48, 1.78, 'M')
  | x==5 = ("Paulo", 24, 1.93, 'M')
  | x==6 = ("Clara", 38, 1.70, 'F')
  | x==7 = ("Bob", 12, 1.45, 'M')
  | x==8 = ("Rosana", 31, 1.58, 'F')
  | x==9 = ("Daniel", 75, 1.74, 'M')
  | x==10 = ("Jocileide", 21, 1.69, 'F')
  | otherwise = ("Acabou!", 0, 0.0, 'x')
```

Construa funções que retornem os seguintes dados:

- número do registro da pessoa de maior idade.
- A idade média de todas as pessoas.
- número de pessoas do sexo masculino com idade superior a 25 anos.

18. Faça uma função na linguagem Haskell, chamada `eliminaVogal`, que recebe uma `String` e retorna uma tupla (`String`, `Int`, `Int`):

- `String` - conterá a string de entrada com as vogais removidas (a, e, i, o, u)
- O primeiro `Int` corresponde ao tamanho da string de entrada
- O segundo `Int` corresponde ao tamanho da string de saída (com vogais removidas)

Exemplos:

```
>>> eliminaVogal "Paradigmas de Linguagens de Programacao"
("Prdgms d Lnggns d Prgrmc", 39, 24)
```

```
>>> eliminaVogal "PLP"
("PLP", 3, 3)
```

Observação: As strings serão compostas apenas de caracteres ASCII, ou seja, não terão acentos e nem 'ç'.