



## (Paradigmas de) Linguagens de Programação

### Exercício 6

Esta atividade deve utilizar recursão, mas nada de composição iterativa e composição sequencial de comandos. As funções devem ser definidas utilizando as primitivas *map*, *reduce*, *all*, *any*, *filter* *next*, podendo também recorrer a outras funções sobre listas que considere necessárias..

Obs: Lembre que *reduce* precisa do import (`from functools import reduce` )

1. Defina a função *soma\_nat* que recebe como argumento um número natural *n* e devolve a soma de todos os números naturais até *n*.

**Exemplo:** *soma\_nat(10) = 1+2+3+4+5+6+7+8+9+10=55*

2. Defina a função *quadrados* que recebe como argumento um número natural *n* devolve a lista dos *n* primeiros quadrados perfeitos.

**Exemplo:** *quadrados(5) = [1, 4, 9, 16, 25]*

3. Defina a função *quadrados\_inv* que recebe como argumento um número natural *n* devolve a lista dos *n* primeiros quadrados perfeitos, por ordem decrescente.

**Exemplo:** *quadrados\_inv(5) = [25, 16, 9, 4, 1]*

4. Defina a função *num\_perf* que recebe como argumento um número inteiro positivo e devolve True se esse número for um número perfeito e False em caso contrário. Lembre que um número perfeito é um número natural que é igual à soma de todos os seus divisores próprios, isto é, a soma de todos os divisores excluindo o próprio número. Pode-se definir funções auxiliares, desde que definidas no paradigma funcional.

**Exemplo:** *num\_perf(6) = TRUE ... (1+2+3=6)*

*num\_perf(5) = FALSE*

5. Defina a função *inverte\_lista* que recebe como argumento uma lista e devolve essa lista invertida.

**Exemplo:** *inverteLista([1,2,3]) = [3, 2, 1]*

6. Defina a função *indices\_pares* que recebe como argumento uma lista de números inteiros *w* e devolve a lista dos elementos de *w* em posições pares.

**Exemplo:** *indices\_pares([4,3,7,1,2,9]) = [4, 7, 2]*



7. Defina a função *triangulo* que recebe como argumento um número natural  $n$  e devolve uma lista em que o primeiro elemento é a lista  $[1]$ , o segundo elemento é a lista  $[1, 2]$  e assim sucessivamente até  $n$ .

**Exemplo:** *triangulo*(4) =  $[[1], [1, 2], [1, 2, 3], [1, 2, 3, 4]]$

8. Defina a função *prod\_lista* que recebe como argumento uma lista de números inteiros e devolve o produto dos seus elementos.

**Exemplo:** *prod\_lista*([2,4,6]) = 48  
*prod\_lista*([]) = 1

9. Defina a função *conta* que recebe como argumentos uma lista de números inteiros  $w$  e um número inteiro  $x$  e devolve o número de ocorrências de  $x$  em  $w$ .

**Exemplo:** *conta*([1,2,3,2,1,2],1) = 2

10. Defina a função *contem\_parQ* que recebe como argumento uma lista de números inteiros  $w$  e devolve True se  $w$  contém um número par e False em caso contrário.

**Exemplo:** *contem\_parQ*([3,5,7,9,11]) = *False*  
*contem\_parQ*([3,4,7,9,11]) = *True*

11. Defina a função *todos\_imparesQ* que recebe como argumento uma lista de números inteiros  $w$  e devolve True se  $w$  contém apenas números ímpares e False em caso contrário.

**Exemplo:** *todos\_imparesQ*([3,5,7,9,11]) = *True*  
*todos\_imparesQ*([3,4,7,9,11]) = *False*

12. Defina a função *pertenceQ* que recebe como argumentos uma lista de números inteiros  $w$  e um número inteiro  $n$  e devolve True se  $n$  ocorre em  $w$  e *False* em caso contrário.

**Exemplo:** *pertenceQ*([1,2,3,4],5) = *False*  
*pertenceQ*([1,2,3,4],5) = *True*

13. Defina a função *int\_listaQ* que recebe como argumento uma lista e devolve True se a lista for constituída exclusivamente por números inteiros e False em caso contrário.

**Exemplo:** *int\_listaQ*([1,2,-3,4,9]) = *True*  
*int\_listaQ*([1.1,3,-3]) = *False*

14. Defina a função *nat\_listaQ* que recebe como argumento uma lista e devolve True se a lista for constituída exclusivamente por números naturais e False em caso contrário.

**Exemplo:** *nat\_listaQ*([1,2,3,-1]) = *False*  
*natt\_listaQ*([1,2,3,4]) = *False*



15. Defina a função *int\_lista\_listaQ* que recebe como argumento uma lista e devolve *True* se a lista for constituída exclusivamente por listas de números inteiros e *False* em caso contrário.  
**Exemplo:** *int\_lista\_listaQ*([[1,2,3],[4,5,6]]) = *True*  
*int\_lista\_listaQ*([[1,2,3],["ola",3]]) = *False*  
*int\_lista\_listaQ*([1,[1,2,3],[4,5,6]]) = *False*
16. Defina a função *escolhe\_pares* que recebe como argumento uma lista de números inteiros *w* e devolve a lista dos elementos pares de *w*.  
**Exemplo:** *escolhe\_pares*([1,2,3,4,5,6,7,8]) = [2, 4, 6, 8]
17. Defina a função *int\_matrizQ* que recebe como argumento uma lista *m* e devolve *True* se *m* for uma matriz de números inteiros e *False* em caso contrário.  
**Exemplo:** *int\_matrizQ*([[1,2],[4,5],[7,8]]) = *True*  
*int\_matrizQ*([[1,2],[4],[7,8]]) = *False*  
*int\_matrizQ*([[1,2],[4],7]) = *False*
18. Defina a função *retira\_negativos* que recebe como argumento uma lista de números inteiros e devolve a lista resultante de retirar todos os números negativos.  
**Exemplo:** *retira\_negativos*([1,2,3,-4,5,-6]) = [1, 2, 3, 5]
19. Defina a função *div* que recebe como argumentos dois números naturais *m* e *n* e devolve o resultado da divisão inteira de *m* por *n*. Neste exercício não pode recorrer às operações aritméticas de multiplicação, divisão e resto da divisão inteira.  
**Exemplo:** *div*(7,2) = 3
20. Defina a função *prim\_alg* que recebe como argumento um número natural *n* e devolve o primeiro algarismo (o mais significativo) na representação decimal de *n*.  
**Exemplo:** *prim\_alg*(8935) = 8
21. Defina a função *media\_digitos* que recebe como argumento um número natural e devolve a média dos seus dígitos.  
**Exemplo:** *media\_digitos*(14276) = 4.0
22. Defina a função *permutacao* que recebe como argumentos duas listas *w1* e *w2* e devolve *True* se *w2* for permutação de *w1*, e devolve *False* em caso contrário.  
**Exemplo:** *permutacao*([1,2,3],[3,2,1]) = *True*  
*permutacao*([1,1,2,3],[3,2,1]) = *False*



23. Defina função *comprimento* que recebe como argumento uma lista e devolve o seu comprimento. Não pode, como é óbvio, recorrer à função *len*.

**Exemplo:** *comprimento*([2,3,5,2,2]) = 5

24. Defina a função *intercala* que recebe como argumentos duas listas *w1* e *w2* e devolve a lista resultante de intercalar os elementos de *w1* com os de *w2*.

**Exemplo:** *intercala*([1,3,5],[2,4,6]) = [1, 2, 3, 4, 5, 6]

*intercala*([1,3,5,7],[2]) = [1, 2, 3, 5, 7]

25. Defina a função *apaga* que recebe como argumentos uma lista de inteiros *w* e um número inteiro *k* e devolve a lista que resulta de apagar de *w* todas as ocorrências de *k*.

**Exemplo:** *apaga*([1,2,1,3,1,4,1,5],1) = [2, 3, 4, 5]

26. Defina a função *maximo* que recebe como argumento uma lista não vazia de números inteiros e devolve o seu máximo.

**Exemplo:** *maximo*([14,-1,5,19,0]) = 19

27. Defina a função *lposicoes* que recebe como argumentos uma lista de números inteiros *w* e um número inteiro *k* e devolve a lista das posições em que *k* ocorre em *w*.

**Exemplo:** *lposicoes*([1,2,3,4,12,2,4,3,2],2) = [1, 5, 8]

28. Defina a função *pos\_max* que recebe como argumento uma lista de números inteiro e devolve a lista das posições onde ocorre o máximo da lista.

**Exemplo:** *pos\_max*([1,2,3,1,2,3]) = [2, 5]

29. Defina a função *ind\_pares* que recebe como argumento uma lista de listas de números inteiros  $w = \{w1, \dots, wn\}$  e devolve a lista  $r = \{r1, \dots, rn\}$  em que  $ri$  é composta pelas posições dos números pares em  $wi$ .

**Exemplo:** *pos\_pares*([[1,2,3],[4,5,6],[7,8,9,10]]) = [[1], [0, 2], [1, 3]]

30. Defina a função *fibonacci* que recebe como argumento um número natural *n* e devolve o *n*-ésimo número da sucessão de Fibonacci. Recorde que a sucessão dos números de Fibonacci é definida recursivamente como se segue: *fibonacci*(1)=1; *fibonacci*(2)=1; *fibonacci*(n+2)=*fibonacci*(n+1)+*fibonacci*(n)

**Exemplo:** *fibonacci*(1) = 1

*fibonacci*(2) = 1

*fibonacci*(3) = 2