

Os exercícios devem ser entregues em um arquivo obrigatoriamente com o nome **Lista2.hs**. Os nomes das funções devem ser idênticos aos nomes apresentados nos exemplos.

1. **pertence:** Declare uma função que verifica se um elemento pertence a uma lista, recebendo um número e uma lista como parâmetro, e retornando True se o elemento estiver na lista e False caso contrário.
Exemplo: `pertence 3 [1, 4, 3, 2] => True`
2. **intercessao:** Declare uma função que retorne a intercessão entre duas listas.
Exemplo: `intercessao [1, 3, 5, 7, 9] [2, 5, 3, 6, 9] => [3, 5, 9]`
3. **inversoLista:** Declare uma função que retorne o inverso de uma lista, similar ao comportamento reverse de Prelude.
Exemplo: `inversoLista [1, 2, 3, 4] => [4, 3, 2, 1]`
4. **nUltimos:** Declare uma função que retorne os n últimos elementos de uma lista.
Exemplo: `nUltimos 3 [1, 2, 3, 4, 5, 6] => [4, 5, 6]`
5. **enesimo:** Declare uma função que receba um número n e uma lista e retorne o n-ésimo elemento da lista. Se n for maior que o tamanho da lista, retorne -1.
Exemplo1: `enesimo 3 [10, 20, 30, 40, 50] => 30`
Exemplo2: `enesimo 10 [10, 20, 30, 40, 50] => -1`
6. **repetir:** Declare uma função que receba dois números inteiros n e m, e retorne uma lista com n vezes o número m.
Exemplo: `repetir 4 10 => [10, 10, 10, 10]`
7. **intercalacao:** Declare uma função que receba duas listas previamente ordenadas e faça a intercalação (merge) dos elementos tendo como resultado a junção das duas listas em uma lista também ordenada.
Exemplo: `intercalacao [10, 15, 17, 20] [1, 2, 13, 15, 22] => [1, 2, 10, 13, 15, 15, 17, 20, 22]`
8. **menor:** Declare uma função que retorne o menor valor de uma lista.
Exemplo: `menor [10, 4, 5, 3, 12] => 3`
9. **removerElem:** Declare uma função que receba uma lista e um elemento e retorne a lista sem a primeira ocorrência desse elemento.
Exemplo: `removerElem 1 [2, 4, 1, 3, 2, 1] => [2, 4, 3, 2, 1]`
10. **ordenarLista:** Usando as funções anteriores, declare uma função que ordene os elementos de uma lista.
Exemplo: `ordenarLista [32, 10, 23, 10, 12, 4] => [4, 10, 10, 12, 23, 32]`
11. **insereElem:** Declare uma função que receba um elemento e uma lista ordenada insira este elemento na lista colocando-o na posição correta, ou seja, a lista resultante deve estar ordenada. Se o elemento já pertencer à lista, ele não deve ser incluído.
Exemplo: `insereElem 12 [6, 9, 10, 15, 20] => [6, 9, 10, 12, 15, 20]`
12. **primeirosDuplas:** Declare uma função que receba uma lista de duplas [(a,b)], e retorne uma lista com o primeiro elemento de cada dupla [a].

Exemplo: primeirosDuplas [("a", 34), ("b", 80), ("c", 180)] => ["a","b","c"]

13. **somaDuplas:** Declare uma função que receba uma lista de duplas e retorne uma lista com a soma dos elementos de cada dupla.
Exemplo: somaDuplas [(1,2), (3,4), (10, 23)] => [3, 7, 33]
14. **menoresDuplas:** Declare uma função que recebe uma lista de duplas e retorna uma lista contendo todas as duplas cujo primeiro elemento seja menor que o segundo.
Exemplo: menoresDuplas [(1, 3), (5, 3), (8, 10), (3, 3)] => [(1, 3), (8, 10)]
15. **separarDuplas:** Declare uma função que receba um valor v e uma lista, a função deve retornar uma dupla de listas, a primeira lista deve conter os elementos que são menores ou iguais a v e a segunda lista deve retornar os elementos maiores que v .
Exemplo: separarDuplas 9 [10, 3, 5, 17, 12, 4, 9] => ([3, 5, 4, 9], [10, 17, 12])
16. **mdc:** Implemente o Algoritmo de Euclides Estendido, tal como a função gcd disponível em Prelude. O Algoritmo de Euclides estendido permite calcular o máximo divisor comum (MDC) fornecendo como resultado os coeficientes.
Considere a seguinte regra:
a se $b = 0$,
 $\text{mdc}(b, a \bmod b)$ se $b > 0$,

Exemplo:
(1) $120/23 = 5$ resta 5
(2) $23/5 = 4$ resta 3
(3) $5/3 = 1$ resta 2
(4) $3/2 = 1$ resta 1
(5) $2/1 = 2$ resta 0
 $\text{MDC}(120,23) = 1$
17. **inversoDupla:** Declare uma função que receba uma lista de duplas $[(x,y)]$, e retorne uma lista com o inverso de cada dupla, ou seja $[(y,x)]$.
Exemplo: inversoDupla [(1,2), (6,1),(4,11)] => [(2,1),(1,6),(11,4)]
18. **simetrico:** Declare uma função que receba uma lista de duplas, e retorne lista indicando se os elementos são iguais ou não (True/False).
Exemplo: simetrico [(1,2), (4,4), (3,2)] => [False,True,False]
19. **pares:** Declare uma função que recebe um número inteiro, e retorna uma lista de duplas de inteiros distintos (x,y) tal que $1 \leq x, y \leq i$.
Exemplo: pares 3 = [(1, 2), (1, 3), (2, 1), (2, 3), (3, 1), (3, 2)]
Importante: Lembre-se do comportamento compreensão de listas em Haskell.
20. **inverteDNA:** Suponha que a sequência de um DNA é formado pelas letras A, T, C e G. Escreva uma função que recebe uma sequência de DNA, e retorne a sequência invertendo as letras, tal que: A será T e T será A, C será G e G será C. Ao final, toda a sequência deve também ser invertida.
Exemplo: inverteDNA "TAGCCGTGCA" = "TGACGGCTA"
21. **trocoCafe:** Desenvolva uma função em Haskell que permita calcular o troco em moedas para o café. Para isso, a função deve receber o valor do café (Int) e o valor em dinheiro pago pelo cliente (Int), e

retornará uma lista de tuplas $[(a, b)]$, tal que a é o valor da moeda, e b a quantidade de moedas deste valor.

São permitidas moedas de 5, 10, 20 e 50 centavos, e deve ser sempre retornado moedas de maior valor antes.

Exemplo: trocoCafe 65 110 = $[(20,2), (5,1)]$

22. **magica:** Desenvolva uma função em Haskell que recebe e retorna uma string, com o comportamento abaixo:

Input String: "AB" / Output String: "AABAA"

Input String: "ABC" / Output String: "AAABBCBBAAA"

Input String: "abC" / Output String: "aaabbCbbaaa"

Input String: "ABCD" / Output String: "AAAABBBCCDCCBBBAAAA"

Importante: essa função é resolvida em 2 partes:

1. dada a string "ABCD", deve ser gerada a lista resultante "AAAABBBCCD";
2. concatenar a lista resultante a segunda parte da lista invertida exceto o último caracter.

Lembre-se, uma string é uma lista de caracteres!