CRISTINÁRIO I - HASKELL
23 DE ABRIL DE 2025

1. O que a função tail retorna quando aplicada a [1,2,3]?
2. Qual o resultado de length [True, False, True]?
3. O que faz a função reverse quando aplicada a "haskell"?
4. O que acontece com null []?
5. Qual a saída de take 2 [10,20,30]?
6. Qual o resultado de [x*2 | x <- [1..5]]?
7. Como seria escrita a lista [2,4,6,8,10] usando compreensão de listas com even?
8. O que filter (>5) [1..10] retorna?
9. Qual o tipo da função not?
10. O que map length ["oi", "haskell", "x"] retorna?
11. Qual o resultado de sum [1..10]?
12. Qual o efeito da função dropWhile (<3) [1,2,3,4]?
13. O que elem 5 [1..10] retorna?
14. Qual o tipo da função ++?
15. Como representar [1,2,3] usando apenas o operador (:)?
16. O que a função zip [1,2] ["a", "b", "c"] retorna?
17. Qual o efeito de replicate 3 "ok"?
18. Qual o resultado de product [1,2,3,4]?
19. O que last [1,2,3,4] retorna?
20. O que init [1,2,3,4] faz?
21. Explique o comportamento da função foldl (+) 0 [1,2,3,4].
22. Qual o tipo mais genérico de map?
23. O que a expressão map (filter even) [[1,2],[3,4,5],[6]] retorna?
24. Implemente uma função somaPares :: [Int] -> Int que soma apenas os números pares de uma lista.
25. O que a expressão (\x y -> x + y) 3 4 retorna?
26. Escreva uma função que receba uma lista de números e retorne o maior número (sem usar maximum).
27. Qual o resultado de scanl (+) 0 [1,2,3]?
28. Dada a definição data Dia = Seg | Ter | Qua deriving Show, o que show Seg retorna?
29. Explique o que é uma função curried em Haskell, com exemplo.
30. O que faz a função >>= no contexto de monads? Dê um exemplo com o tipo Maybe.
31. Implemente uma função dobro que receba um número e retorne o dobro dele.
32. Crie uma função parOuImpar :: Int -> String que retorne "Par" se o número for par, e "Ímpar" caso contrário.
33. Implemente uma função somaLista :: [Int] -> Int que some todos os elementos de uma lista.

34. Escreva uma função maiorDeDois :: Int -> Int -> Int que retorne o maior entre dois números.

35. Crie uma função inverso :: [a] -> [a] que inverta a ordem dos elementos de uma lista.

36. Implemente filtraPares :: [Int] -> [Int], que retorne apenas os números pares da lista.

37. Faça uma função multiplicaTodos :: [Int] -> Int que retorne o produto de todos os elementos da lista.

38. Escreva uma função recursiva contaElementos :: [a] -> Int que conte quantos elementos há na lista.

39. Implemente ultimo :: [a] -> a, que retorna o último elemento de uma lista.

40. Crie segundo :: [a] -> a, que retorna o segundo elemento de uma lista (assuma que sempre haverá pelo menos dois).

41. Implemente uma função repete :: Int -> a -> [a] que cria uma lista com o elemento repetido N vezes.

42. Escreva uma função intercala :: [a] -> [a] -> [a] que intercale os elementos de duas listas.

43. Crie contaVogais :: String -> Int que conte quantas vogais há numa string.

44. Implemente ehPalindromo :: String -> Bool, que verifica se uma palavra é um palíndromo.

45. Faça uma função aplicaDuasVezes :: (a -> a) -> a -> a que aplica uma função duas vezes sobre um valor.

46. Crie potencia :: Int -> Int -> Int que eleva um número à potência de outro (sem usar (^)).

47. Implemente concatListas :: [[a]] -> [a] que concatena uma lista de listas.

48. Crie uma função removerElemento :: Eq a => a -> [a] -> [a] que remove todas as ocorrências de um elemento na lista.

49. Implemente frequencia :: Eq a => a -> [a] -> Int, que conta quantas vezes um elemento aparece em uma lista.

50. Escreva uma função intersecao :: Eq a => [a] -> [a] -> [a] que retorna os elementos em comum entre duas listas.