Lista de Exercícios

1. **soma**: recebe uma dupla de inteiros e retorna a sua soma.
Ex.: soma (1,2) ==> 3

2. **menorDeDois**: recebe dois valores inteiros e retorna o menor.

3. **menorDeTres**: recebe três valores inteiros e retorna o menor.

4. **areaCircunf**: recebe um float representando o raio de uma circunferência e retorna sua área.

5. **andTres**: recebe três booleanos e retorna a operação AND aplicada aos três.

6. Defina o operador # que concatena duas strings caso elas sejam diferentes. Se forem iguais retorna uma das duas.

7. **ePrimo**: recebe um número natural e verifica se ele é primo.

8. **fatorial**: recebe um número natural e retorna o seu fatorial.

9. **fibonacci**: recebe um número inteiro positivo e retorna o n-ésimo elemento da seqüência de Fibonacci.

10. **elemento**: recebe uma lista qualquer e um número inteiro positivo para retornar o n-ésimo elemento da lista.
Ex.: elemento [3,5,7] 1 ==> 3

11. **pertence**: recebe um elemento e uma lista e verifica se o elemento pertence à lista.
Ex.: pertence 1 [3,7,4,2] ==> False

12. **nroElementos**: recebe uma lista qualquer e retorna o número de elementos na lista.
Obs.: não usar a função length

13. **maior**: recebe uma lista de inteiros e retorna o maior.
Obs.: não usar a função max

14. **contaOcorrencias**: recebe um elemento e uma lista e retorna o número de ocorrências do elemento na lista.

15. **unicaOcorrencia**: recebe um elemento e uma lista e verifica se existe uma única ocorrência do elemento na lista.
Ex.:
unicaOcorrencia 2 [1,2,3,2] ==> False
unicaOcorrencia 2 [3,1] ==> False
unicaOcorrencia 2 [2] ==> True

16. **maioresQue**: recebe um número inteiro e uma lista de inteiros e retorna uma lista com os números que são maiores que o fornecido.
Ex.:

maioresQue 10 [2,12,4,7,8,30,15] ==> [12,30,15]

17. **concatena**: recebe duas listas quaisquer e retorna uma terceira lista com os elementos da primeira no início e os elementos da segunda no fim.
Ex.:
concatena [] [] ==> []
concatena [1,2] [3,4] ==> [1,2,3,4]

18. **remover**: recebe um elemento e uma lista e retorna a lista sem a primeira ocorrência do elemento.

19. **removerUltimo**: recebe uma lista e remove o último elemento da lista.

20. **removerRepetidos**: recebe uma lista qualquer e retorna outra lista sem repetição de elementos.
Ex.:
removerRepetidos [7,4,3,5,7,4,4,6,4,1,2] ==> [7,4,3,5,6,1,2]

21. **maiores**: recebe um numero natural n e uma lista de inteiros e retorna uma lista com os n maiores números sem alterar a ordem entre os elementos.
Ex.:
maiores 4 [9,3,5,7,8,4,4,7] ==> [9,7,8,7]

22. **geraSequencia**: recebe um número inteiro n positivo e retorna a lista [1, -1, 2, -2, 3, -3, ... n, -n].

23. **inverte**: recebe uma lista e retorna outra, que contém os mesmos elementos da primeira, em ordem invertida.

24. **divide**: recebe um número natural n e uma lista e retorna um par onde o primeiro elemento é uma lista com os n primeiros números da lista original e o segundo elemento é uma lista com o resto dos elementos da lista original.
Ex.:
divide 0 [1,2,3,4] ==> ([], [1,2,3,4])
divide 2 [1,2,3,4] ==> ([1,2], [3,4])

25. **intercala**: recebe duas listas e retorna outra lista com os elementos das listas originais intercalados.
Ex.:
intercala [1,2,3] [8,9] ==> [1,8,2,9,3]
intercala [] [1,2,6] ==> [1,2,6]

26. **uniao**: recebe duas listas quaisquer que não contenham elementos repetidos e retorna uma nova com todos os elementos das duas listas originais (sem repetição).
Ex.:
uniao [3,6,5,7] [2,9,7,5,1] ==> [3,6,5,7,2,9,1]

27. **interseccao**: recebe duas listas quaisquer sem elementos repetidos e retorna uma lista com os elementos que são comuns às duas.
Ex.:

interseccao [3,6,5,7] [9,7,5,1,3] ==> [3,5,7]

28. **sequencia**: recebe dois numeros naturais n e m, e retorna uma
lista com n elementos, onde o primeiro é m, o segundo é m+1, etc...
Ex.:
sequencia 0 2 ==> []
sequencia 3 4 ==> [4,5,6]

29. **insereOrdenado**: recebe uma lista de números em ordem crescente e um número qualquer, retorna uma lista de números em ordem crescente com os elementos da lista inicial mais o número passado.

30. **ordenado**: recebe uma lista de números inteiros e verifica se eles estão ordenados ou não.

31. **ordena**: recebe uma lista com números e retorna outra lista com os números ordenados
Ex.:
ordena [7, 3, 5, 7, 8, 4, 4] ==> [3, 4, 4, 5, 7, 7, 8]

32. **rodarEsquerda**: recebe um número natural, uma lista e retorna uma nova lista onde a posição dos elementos mudou como se eles tivessem sido "rodados".
Ex.:
rodarEsquerda 0 [‘a’,‘s’,‘d’,‘f’,‘g’] ==> [‘a’,’s’,’d’,’f’,’g’]
rodarEsquerda 1 [‘a’,‘s’,‘d’,‘f’,‘g’] ==> [‘s’,’d’,’f’,’g’,’a’]
rodarEsquerda 3 [‘a’,’s’,’d’,’f’,’g’] ==> [‘f’,’g’,’a’,’s’,’d’]
rodaresquerda 4 [‘a’,‘s’,‘d’,‘f’,‘g’] ==> [‘g’,’a’,’s’,’d’,’f’]

33. **rodarDireita**: recebe um número natural, uma lista e retorna uma nova lista onde a posição dos elementos mudou como se eles tivessem sido "rodados".
Ex.:
rodarDireita 0 [‘a’,’s’,’d’,’f’,’g’] ==> [‘a’,’s’,’d’,’f’,’g’]
rodarDireita 1 [‘a’,’s’,’d’,’f’,’g’] ==> [‘g’,’a’,’s’,’d’,’f’]
rodarDireita 3 [‘a’,’s’,’d’,’f’,’g’] ==> [‘d’,’f’,’g’,’a’,’s’]
rodarDireita 4 [‘a’,’s’,’d’,’f’,’g’] ==> [‘s’,’d’,’f’,’g’,’a’]

34. **todasMaiusculas**: recebe uma string qualquer e retorna outra string onde todas as letras são maiúsculas. Pode ser útil saber os seguintes códigos ASCII: a=97, z=122, A=65, Z=90, 0=48, 9=57, espaço=32.
Ex.: todasMaiusculas "abc 123" = "ABC 123"

35. **primeirasMaiusculas**: recebe uma string qualquer e retorna outra string onde somente as iniciais são maiúsculas.
Ex.:
primeirasMaiusculas "FuLaNo bElTrAnO silva" ==> "Fulano Beltrano Silva"

36. **seleciona**: recebe uma lista qualquer e uma lista de posições, retorna uma lista com os elementos da primeira que estavam nas posições indicadas.
Ex.:
seleciona [‘a’,’b’,’c’,’d’,’e’,’f’] [0,3,2,3] ==> [‘a’,’d’,’c’,’d’]

37. **palindrome**: recebe uma string e verifica se ela é uma palíndrome ou nao
Ex.:
palindrome "ana" ==> True
palindrome "abbccbba" ==> True
palindrome "abbdbbaa" ==> False

38. **somaDigitos**: recebe um número natural e retorna a soma de seus dígitos.
Ex.:
somaDigitos 3284 ==> 17

39. **compactar**: recebe uma lista de inteiros e transforma todas as repetições em sub-listas de dois elementos: sendo o primeiro elemento o número de repetições encontradas e o segundo elemento é o número que repete na lista original. Os números que não repetem na lista original não devem ser alterados.
Ex.:
compactar [2,2,2,3,4,4,2,9,5,2,4,5,5,5] ==> [[3,2],[3],[2,4],[2],[9],[5],[2],[4],[3,5]]

40. **separaParImpar**: um programa que dada uma lista, retorne uma tupla listalista (de inteiros) onde a lista da esquerda contém os números
impares e a lista da direita os números pares
Ex.:
separaParImpar [1,2,3,4,5,6,7] => ([1,3,5,7],[2,4,6])

41. **calculaArea**: recebe uma forma geométrica (você deve definir este tipo) que pode ser um triângulo, quadrado, retângulo ou circunferência e retorna a sua área.

42. **paraLista**: recebe um tipo lista (você deve definir este tipo) e retorna uma lista em haskell.
Ex.:
paraLista (Elem 2(Elem 3(Elem 4(Vazia)))) => [2,3,4]

43. **somaArvore**: recebe uma árvore binária de inteiros (você deve definir este tipo) e retorna a soma de seus nós.

44. **nroOcorrencias**: recebe um número inteiro e uma árvore binária de inteiros (você deve definir este tipo) e retorna a quantidade de ocorrências desse número.

45. **arvoreEmLista**: recebe uma árvore binária de qualquer tipo (você deve definir este tipo) e retorna uma lista com todos elementos da árvore da esquerda para direita.

46. **somaPares**: recebe uma lista com par de inteiros e retorna outra lista que contém a soma de cada par.
Obs.: Resolver esta questão utilizando compreensão de lista.
Ex.:
somaPares [(2,3),(1,8)] => [5,9]

47. **soPrimos**: recebe uma lista de inteiros e retorna outra lista apenas com os números primos.
Obs.: Resolver esta questão utilizando compreensão de lista.

Ex.:
soPrimos[2,4,5,6,8] => [2,5]

48. **removeChar**: recebe um caractere e uma string, retornando a string sem as ocorrências do caractere informado.
Obs.: Resolver esta questão utilizando compreensão de lista.

49. **somenteDigitos**: recebe uma string e retorna apenas os dígitos.
Obs.: Resolver esta questão utilizando compreensão de lista.
Ex.:
somenteDigitos “ab32c3” => “323”

50. Um sorteio da loto pode ser representado por uma lista de cinco números. Um conjunto de cartões de apostas pode ser representado por uma lista de listas (cada lista representando um cartão). Assuma que o prêmio e os números em cada cartão estão ordenados. Defina as seguintes funções:

numAcertos :: Cartao -> Cartao -> Int: retorna o número de acertos em um cartão. (o primeiro cartão é o sorteado).
numQuinas :: Cartao -> [Cartao] -> Int: retorna o número de cartões premiados com a quina.
bilhetesPremiados :: Cartao -> [Cartao] -> [Int]: retorna uma lista de inteiros contendo o(s) indice(s) dos cartões premiados com a quina.
terQuadQuin :: Cartao -> [Cartao] -> (Int, Int, Int): retorna uma tupla de três inteiros contendo a quantidade de cartões premiados com ternos, quadras e quinas, respectivamente.
indTerQuadQuin :: Cartao -> [Cartao] -> ([Int], [Int], [Int]): retorna uma tupla de três listas de inteiros contendo o índice dos cartões premiados com ternos, quadras e quinas, respectivamente.