

Dica: Resolva todos os exercícios sem utilizar o computador. Uma vez que os exercícios estejam prontos, utilize o GHCI para conferir suas respostas.

1. Mostre os resultados das seguintes execuções, ou explique porque elas não podem ser executadas.

- - primeira versão

separa :: [Int] -> [Int]

separa (p:s:r) = (s:r)

> separa [1 ,2 ,3 ,4 ,5]

> separa [1 ,2 ,3]

> separa [1 ,2]

> separa [1]

- - segunda versão

separab :: [Int] -> [Int]

separab (p:s:r) = (r:s:p)

> separab [1 ,2 ,3 ,4 ,5]

> separab [1 ,2 ,3]

> separab [1 ,2]

> separab [1]

- - terceira versão

separac :: [Int] -> [Int]

separac (p:r) = r

> separac [1 ,2 ,3 ,4 ,5]

> separac [1 ,2 ,3]

> separac [1 ,2]

> separac [1]

> separac []

2. Implemente as seguintes funções em Haskell. Mostre a execução passo a passo dessas funções para os exemplos fornecidos.

(a) Determinar o comprimento de uma lista, ex:

> comprimento [3 ,14 ,1 ,5 ,9]

5

(b) Determinar o somatorio dos elementos de uma lista, ex:

> somatorio [3 ,14 ,1 ,5 ,9]

32

(c) Determinar o somatorio dos elementos ímpares de uma lista, ex:

> somatorio_impares [3 ,14 ,1 ,5 ,9]

18

Obs.: Não utilize a função odd

(d) Determinar a soma dos quadrados dos elementos de uma lista, ex:

> soma_quadrados [3 ,14 ,1 ,5 ,9]

312

(e) Determinar a soma dos elementos de uma lista que são múltiplos de 3, ex:

> soma_mult_3 [3 ,14 ,1 ,5 ,9]

12

(f) Determinar o produtorio dos elementos de uma lista, ex: ´

> produtorio [3 ,14 ,1 ,5 ,9]

1890

(g) Determinar o n-esimo elemento de uma lista, ex:

```
> n_esimo 3 [3 ,14 ,1 ,5 ,9]
```

5

Obs.: Considere que o primeiro elemento da lista esta na posição

(h) Determinar o ultimo elemento de uma lista, ex:

```
> ultimo [3 ,14 ,1 ,5 ,9]
```

9

Obs.: Nao utilize a função last.

(i) “Duplicar” os elementos de uma lista, ex:

```
> duplica [3 ,14 ,1 ,5 ,9]
```

```
[3 ,3 ,14 ,14 ,1 ,1 ,5 ,5 ,9 ,9]
```

(j) Reverter uma lista, ex:

```
> reverso [3 ,14 ,1 ,5 ,9]
```

```
[9 ,5 ,1 ,14 ,3]
```

(k) Substituir todas as ocorrencias de um elemento x em uma lista por outro elemento y, ex:

```
> substituir_todos 1 2 [3 ,14 ,1 ,5 ,1]
```

```
[3 ,14 ,2 ,5 ,2]
```

(l) Substituir a primeira ocorrencia de um elemento x em uma lista por outro elemento y, ex:

```
> substituir_primeiro 1 2 [3 ,14 ,1 ,5 ,1]
```

```
[3 ,14 ,2 ,5 ,1]
```

(m) Determinar o produto interno de dois vetores representados por listas, ex:

```
> produto_interno [3 ,14 ,1] [5 ,9 ,26]
```

167

(n) Determinar o maior elemento de uma lista, ex:

```
> maior [3 ,1 ,4 ,1 ,5 ,9]
```

14

(o) “Desduplicar” os elementos de uma lista, ex:

```
> desduplicar [3 ,3 ,14 ,14 ,1 ,1 ,5 ,5 ,9 ,9]
```

```
[3 ,14 ,1 ,5 ,9]
```

(p) Escreva uma func,ao que retorna verdadeiro se todos os elementos de uma lista de inteiros forem ímpares, ou falso, caso contrario, ex:

```
> impares [3 ,1 ,5 ,9]
```

True

(q) Inserir um elemento ordenadamente em uma lista ja ordenada, ex:

```
> insere 4 [1 ,3 ,5]
```

```
[1 ,3 ,4 ,5]
```

Se o elemento ja existir, pode inseri-lo depois de sua ocorrência.

(r) Calcular o quadrado de cada elemento da lista, ex:

```
> quadrado [3 ,14 ,1 ,5 ,9]
```

```
[9 ,196 ,1 ,25 ,81]
```

(s) Verificar se um elemento pertence a lista, ex: `

```
> pertence 5 [3 ,14 ,1 ,5 ,9]
```

True

(t) Remover todos os elementos k de uma lista, ex:

```
> remover_todos 1 [3 ,14 ,1 ,5 ,9 ,1]
```

```
[3 ,14 ,5 ,9]
```

(u) Dada uma lista de tuplas-2, criar uma nova lista com apenas os primeiros elementos de cada tupla, ex:

```
> primeiros [(3 ,14) ,(1 ,5) ,(9 ,1)]
```

```
[3,1,9]
```

(v) Dada uma lista de listas, concatenar todas as sub-listas em uma unica lista, ex:

```
> concatenar [[3,14,1,5],[9,1,2]]
```

```
[3,14,1,5,9,1,2]
```

(w) Dadas duas listas de mesmo tamanho, obter uma terceira lista, representando a diferença absoluta entre as listas dadas, ex:

```
> diferenca [1, 3, 2, 8] [2, 5, 6, 8]
```

```
[1, 2, 4, 0]
```

(x) Dadas duas listas, verificar se ambas sao exatamente iguais, ou seja, possuem o mesmo tamanho e os mesmos elementos, ex:

```
> iguais [1, 3, 2, 8] [1, 3, 2, 8]
```

```
True
```

(y) Dada uma lista de numeros reais, calcular a média aritmética dos elementos, ex:

```
> media [3.0, 1.4, 1.0]
```

```
1.8
```

(z) Dado um inteiro n, devolva a lista de todos os numeros inteiros ímpares entre 1 e n, ex:

```
> lista_impares 9
```

```
[1,3,5,7,9]
```

3. Implemente, usando recursão, as seguintes operações sobre conjuntos, lembrando que um conjunto e uma sequência ordenada de elementos não-repetidos:

(a) pertence: define se um dado elemento pertence a um conjunto, ex:

```
> pertence 2 [1,3,5]
```

```
False
```

(b) uniao: dados dois conjuntos, fornece a uniao deles, ex:

```
> uniao [1,3,5] [2,3,4]
```

```
[1,2,3,4]
```

(c) inter: dados dois conjuntos, fornece a intersecção deles, ex:

```
> inter [1,3,5] [1,2,3,4]
```

```
[1,3]
```

(d) diff: dados dois conjuntos, fornece a diferenca deles, ex:

```
> diff [1,3,5] [1,2,3,4]
```

```
[2,4,5]
```

(e) subc: dados dois conjuntos, diz se o primeiro e subconjunto do segundo, ex:

```
> sub_conjunto [2,4,6] [1,2,3,4,5,6,7,8]
```

```
True
```

(f) interc: intercalar dois conjuntos de mesmo tamanho em um terceiro conjunto, ex:

```
> intercala [1,5,7,9,10] [2,3,4,5,8]
```

```
[1,2,3,4,5,6,7,8,9,10]
```

Nao se esqueça de declarar corretamente os tipos de cada função.

4. No modulo Char encontramos a função toUpper que converte uma letra minuscula na sua correspondente maiuscula.

(a) Crie uma função recursiva maius que converte todas as letras de uma palavra em maiusculas;

(b) Usando a função isAlpha, tambem do módulo Char, refaça a função maius para descartar símbolos e numeros;

(c) Faça uma nova função que recebe uma palavra e devolve em tupla a palavra original e a sua correspondente escrita em maiuscula.

OBS: Para importar o modulo Char usa-se import Data.Char, no seu arquivo fonte

5. Considere uma lista de numeros inteiros como entrada de uma função. A função retorna verdadeiro se a lista e alternante ou falso, caso contrário. Uma lista e alternante se seus dígitos se alternam entre par e impar.

Exemplos:

[1,2,3,4,1,2] é alternante

[2,1,4,1,6,3,8,1] é alternante

[1,3,2,1] nao é alternante

[2,1,4,3,5,4] nao é alternante

6. A representação binária de um número consiste em realizar sucessivas divisoes deste numero por 2 e imprimir do ultimo para o primeiro, todos os restos das divisões.

Por exemplo:

$6 / 2 = 3$ (resto 0) ! $3 / 2 = 1$ (resto 1) ! $1 / 2 = 0$ (resto 1)

6 em binario = 110

Implemente a função onverte :: Int -> [Int], ex:

> converte 23

[1 ,0 ,1 ,1 ,1]

7. Escreva uma função que dada uma lista com 0s e 1s, representando um número binário, calcule seu correspondente na forma decimal.

> converte [1 ,0 ,1 ,1 ,1]

23

8. Implemente uma função que tem como entrada um número inteiro e que retorna uma lista com cada dígito do numero separadamente. Dica: parte inteira e resto da divisão por 10.

Exemplo:

> digitos 1234

[1, 2, 3, 4]