Programação Funcional

Roteiro de atividades práticas 7

Funções de Alta Ordem e uso de Contadores

Esse roteiro deve ser desenvolvido de forma assíncrona pelo aluno. Para que essas atividades sejam avaliadas e contabilizadas (nota e presença) o arquivo .hs referente às atividades abaixo deve ser enviado para o e-mail claudineyrt@gmail.com.

Data máxima de envio: até 06/05/2021 (Quinta) até 23H59

1) Usando a função `map`, escreva a função `paridade` a seguir que recebe uma lista de inteiros `l` e retorna uma lista contendo os valores booleanos que indicam a paridade dos elementos de `l`.

```
> paridade [1,2,3,4]
[False,True,False,True]
```

2) Usando a função `map`, escreva a função `prefixos` a seguir que recebe uma lista de strings `l` e retorna uma lista contendo os três primeiros caracteres de cada elemento de `l`.

```
> prefixos ["haskell", "curry"]
["has","cur"]
```

3) Usando a função `map`, escreva a função `saudacao` a seguir que recebe uma lista de nomes (strings) `l` e retorna uma lista contendo cada elemento de `l` concatenado com a saudação "Oi " na frente de cada nome

```
> saudacao ["Daniel", "Maria","Pedro"]
["Oi Daniel", "Oi Maria","Oi Pedro"]
```

4) Reescreva a definição da função **`filter`** `que já faz parte da` biblioteca padrão do Haskell, chamando-a de **`filtrar.`** Além disso, defina a função filtrar usando lista por compreensão.

5) Usando a função `filter`, escreva a função `pares` que recebe uma lista de inteiros `lst e` e retorna uma lista contendo os elementos pares de `lst`.

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

6) Usando a função `filter`, escreva a função `solucoes` a seguir que recebe uma lista de inteiros `l` e retorna uma lista contendo os valores que satisfazem a equação `(5*x + 6) < (x*x)`. Use uma expressão lambda (função anônima) para representar a função que realiza o teste do filtro.

```
>solucoes [3,4,5,6,7,8,9]
[7,8,9]
```

7) Usando a função `foldr1`, escreva a função `maior` a seguir que recebe uma lista e retorna seu maior elemento.

```
> maior [4,5,2,1]
5
```

8) Usando a função `foldr`, escreva a função `menor_min10` a seguir que recebe uma lista e retorna o menor elemento da lista, desde que este não acima de 10, Se o menor elemento for um valor acima de 10, retorna 10.

```
> menor_min10 [4,5,2,1]
1

> menor_min10 [14,21]
10
```

9) Usando a função `foldr`, escreva a função `junta_silabasplural` a seguir que recebe uma lista de sílabas (strings) e retorna uma palavra (string) formada pela concatenação das sílabas e incluindo um "s" no final .

```
> junta_silabas_plural ["cor","ti","na"]
"Cortinas"

> junta_silabas_plural ["ti","jo","lo"]
"tijolos"
```

Obs: nos exercícios 10 e 11, o uso de funções de alta ordem não é necessário.

10) Implemente a função **menores10** que recebe uma lista de inteiros e retorna duas informações: uma nova lista com todos os elementos menores que 10 da lista de entrada e quantos elementos são menores que 10. Essas informações devem ser retornadas em uma tupla. Ex:

```
> menores10 [1,34,6,3,21]
([1,6,3],3)
```

11) Implemente a função **busca**, que verifica se um dado elemento está presente em uma lista de entrada. A função retorna se o elemento está ou não presente na lista (**True ou False**) e o número de comparações feitas na busca. Essas informações devem ser retornadas em uma tupla. Ex:

```
> busca_elem 1 [5,4,1,6,3,2]
(True,3)

> busca_elem 7 [5,4,1,6,3,2]
(False,6)
```