Programação Funcional

Roteiro de atividades práticas 6

Funções de Alta Ordem e Algoritmos de Ordenação

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 a monitora Domitila Crispim, conforme explicado em sala.

Data máxima de envio: até 01/10/2020 (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"
```

10) Implemente as funções de ordenação `bubblesort, selectionsort, insertionsort e quicksort`, conforme as definições apresentadas no material de aula (ordenação crescente). Posteriormente,teste cada função com os seguintes exemplos de listas e observe a diferença de desempenho (tempo de processamento, dependendo da lista que é fornecida como entrada.

```
lst1 = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
lst2 = [20,19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1]
lst3 = [11,12,13,14,15,16,17,18,19,20,1,2,3,4,5,6,7,8,9,10]
lst4 = [10,9,8,7,6,5,4,3,2,1,20,19,18,17,16,15,14,13,12,11]
lst5 = [11,12,13,14,15,5,4,3,2,1,16,17,18,19,20,10,9,8,7,6]
lst6 = [1,12,3,14,5,15,4,13,2,11,6,17,8,19,20,10,9,18,7,16]
lst7 = [1..1000]
lst8 = [1000,999..1]
lst9 = lst1++[0]
lst10 = [0]++lst3
lst11 = lst1++[0]++lst3
```

```
lst12 = lst3++[0]++lst1
```

Nesse exercício, o aluno deve enviar apenas o código de implementação.

11) Altere cada um dos algoritmos de ordenação do exercício 10 para também contabilizar e retornar o número de comparações (do tipo `x <= y`) feitas pelo algoritmo em sua execução. Teste com as listas dadas. Exemplo:

```
> quicksort2 lst6
([1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20],168)
```

Nesse exercício, o aluno deve enviar o código de implementação e o resultado da aplicação de cada algoritmo sobre cada uma das 12 listas (o número de comparações realizado)

12) Altere cada um dos algoritmos de ordenação do exercício 10 para que façam ordenação decrescente e também retornem o número de comparações. Teste com as listas dadas. Exemplo:

```
> quicksort3 lst6
([20,19,18,17,16,15,14,13,12,11,10,9,8,7,6,5,4,3,2,1],168)
```

Nesse exercício, o aluno deve enviar o código de implementação e o resultado da aplicação de cada algoritmo sobre cada uma das 12 listas (o número de comparações realizado)