UNIVERSIDADE FEDERAL DE SERGIPE
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
DEPARTAMENTO DE COMPUTAÇÃO
PRIMEIRA LISTA DE PROGRAMAÇÃO-FUNCIONAL – PARTE 2

Profa. Leila Maciel de Almeida e Silva Data de entrega:27/06/2019

Aulas: Projeto e Escrita de Programas e Construção de Algoritmos por Indução Fraca e Forte

As questões a seguir devem ser feitas usando **funções recursivas**. A ideia é que refaçam as funções da questão 3 **sem o uso** de compreensões, mas adotando uma formulação indutiva. O problema é exatamente o mesmo da questão 3, listarei a seguir as funções que precisam ser reformuladas.

4.1 Escreva as funções a seguir para manipular o cardápio do restaurante armazenado no sistema.

(a) Adiciona um item no menu. Se o código do item já existir no menu deve retornar uma mensagem de erro sinalizado que existe um item já cadastrado para aquele código.

```
adicionaItemMenu :: Menu -> ItemRest -> Menu
```

(b) Remove um item no menu, informando seu código. Se o código do item não existir no menu deve retornar uma mensagem de erro sinalizando que não existe um item no menu para aquele código.

```
removeItemMenu :: Menu -> Codigo -> Menu
```

(c) Coleta um item no menu, informando seu código. Para simplificar, considere que esta operação só tem o caso de sucesso, ou seja, o item consultado sempre vai existir no menu.

```
coletaItemMenu :: Menu -> Codigo -> ItemRest
```

4.2 Escreva funções que permitam realizar ações que normalmente ocorrem no restaurante.

(a) Adiciona um pedido de uma mesa na lista de pedidos do restaurante. Se a lista de pedidos da mesa não for vazia, a função precisa checar se o código do item agora solicitado já existe na lista de pedidos da mesa. Em caso afirmativo, a quantidade a ele associada vai ser incrementada com a nova solicitação. Caso contrário, o novo item vai apenas ser adicionado à lista já existente. Esta função assume, por simplicidade, que o código do item a ser incluído existe no menu do restaurante.

```
adicionaPedido :: Mesa -> ItemCliente -> PedidosMesas -> PedidosMesas

adicionaPedido 1 (15,2) [[(15,1),(40,1),(52,2)], [], [(15,1), (150,1)]]
```
retornará `[[(15,3),(40,1),(52,2)], [], [(15,1), (150,1)]]`

```
adicionaPedido 1 (150,2) [[(15,1),(40,1),(52,2)], [], [(15,1), (150,1)]]
```
retornará `[[(150,2),(15,1),(40,1),(52,2)], [], [(15,1), (150,1)]]`

(b) Cancela um pedido na lista de pedidos do restaurante, para uma dada mesa. Por simplicidade, suponha que o pedido a ser cancelado existe na lista de pedidos da mesa. No ato do cancelamento,

se a quantidade a ser cancelada for igual ou superior à quantidade já solicitada, o item deve ser removido da lista de pedidos da mesa. Se o cancelamento for parcial, o item permanecerá na lista, mas com a quantidade decrementada de acordo com a quantidade cancelada.

```
cancelaPedido :: Mesa -> ItemCliente -> PedidosMesas -> PedidosMesas

cancelaPedido 1 (40,1) [[(15,1),(40,1),(52,2)], [], [(15,1), (150,1)]]
```
retornará `[[(15,1),(52,2)], [], [(15,1), (150,1)]]`

```
cancelaPedido 1 (52,1) [[(15,1),(40,1),(52,2)], [], [(15,1), (150,1)]]
```
retornará `[[(15,1),(40,1),(52,1)], [], [(15,1), (150,1)]]`

(c) Gera lista completa do pedido, que será usada quando a conta for finalizada. Os nomes e os preços de cada item são coletados do menu, usando o código do item para fazer a coleta. O preço gerado na lista de saída já é o preço do item totalizado, ou seja, o preço unitário multiplicado pela quantidade.

```
pedidoCompletoMesa :: Mesa -> PedidosMesas -> Menu ->
                                        [(Quant, Nome, Preco)]

pedidoCompletoMesa 1 [[(15,1),(40,1),(52,2)], [], [(15,1), (150,1)]] [(150, "Pastel", 1000), (15, "Agua", 400),
            (2, "Cerveja", 800), (40, "Picanha", 8850),
            (52, "Pudim", 1275)]
```

retornará `[(1,"Agua",400),(1,"Picanha",8800),(2,"Pudim",2550)]`

(d) Gera o total da conta a partir da lista de pedidos de uma mesa, após aplicar a função do item 3.2 (c).

```
totalMesa :: [(Quant, Nome, Preco)] -> Preco

totalMesa [(1,"Agua",400),(1,"Picanha",8800),(2,"Pudim",2550)]
```
retornará `11750`

Você pode usar as funções de formatação já feitas e efetuar os mesmos testes com as novas funções

```
adicionaPedido 2 (150, 2) pedidosRest
adicionaPedido 2 (2,2) it
adicionaPedido 1 (150, 2) it
cancelaPedido 1 (150,1) it
geraConta 1 it cardápio
liberaMesa 1 pedidosRest
```