UNIVERSIDADE FEDERAL DE ALFENAS
Curso de Ciência da Computação - Instituto de Ciências Exatas
Disciplina: Programação Funcional - Prova 02

Nome:	Correção 100 pt	Data 18/08/2021	Nota
Nome:	Correção 100 pt	Data 18/08/2021	Nota

Instruções para a realização desta prova

a) Esta prova deverá ser realizada em dupla
b) O conteúdo entregue deverá ser um arquivo de *script* para Haskell
c) O arquivo entregue deverá conter código haskell e comentários
d) O arquivo entregue deverá ser nomeado da seguinte forma:
   d.1) progFunc-P2_<número matrícula 1>_<número matrícula 2>.hs
   d.2) em que <número matrícula x> é o número de matrícula do estudante x
   d.3) Exemplo: **progFunc-P2_2029.1.04.049_2029.1.04.015_.hs**
e) Renomeie o arquivo progFunc-P2_<mat01>_<mat02>_.hs e edite o cabeçalho e seu *script*
f) Os *scripts* fora do formato e que não compilarem no Hugs não serão avaliados
g) Outras instruções de formatação do *script* estão no arquivo progFunc-P2_<mat01>_<mat02>_.hs
Boa Prova

Valor: 50% dos pontos da disciplina
Presenças: 50% das presenças do período

1. A Tabela 1 representa o quadro relativo a brindes (viagens) que uma empresa tem para oferece a seus funcionários quando são promovidos. Este quadro é gerado uma vez por mês e está atualizado. Para esta tabela, faça o que se pede.

código	cidade	número de passagens	número de hospedagens
01	Natal	21	34
02	Bertioga	17	65
03	Rio de Janeiro	9	10
04	Curitiba	3	54
05	Petrolina	2	09
06	Salvador	0	01
07	Teresina	21	56

Tabela 1: promoção

Considere o tipo declarado em Haskell como:

```
type Brinde = (String, Int, Int)
```

(a) Escreva uma função chamada *mapa* que, dado um inteiro referente ao código de um brinde, ela retorne a tupla do tipo *Brinde* com as informações referentes.

(b) Complete as funções declaradas abaixo para que, quando uma tupla da função mapa for fornecida, cada uma das funções abaixo possa retornar o valor referente presente na tupla

```
cidade::Brinde -> String

nPassagens::Brinde -> Int

nHospedagens::Brinde -> Int
```

(c) Faça uma função que, usando as declaradas anteriormente, forneça o número total de passagens que a empresa tem a oferecer.

(d) Faça uma função que, usando as declaradas anteriormente, forneça o número total de hospedagens que a empresa tem a oferecer.

(e) Faça uma função que, usando as declarações anteriores, receba como entrada o nome da cidade e forneça como saída uma tupla contendo o código, o número de passagens e o número de hospedagens oferecido para a cidade. Caso a função não encontre a cidade que foi digitada, ela deverá retornar (0, 0, 0).

(f) Faça uma função que receba uma tupla do tipo (String,Int,Int) fornecida pelo gerente do sistema. Nesta tupla, estão os valores do nome da cidade, o número de passagens e o número de hospedagens desejados por um cliente que acaba de receber uma promoção.

Caso a empresa tenha a cidade e possa oferecer a quantidade de passagens e de hospedagens desejados, a função deve retornar *True*. Caso contrário, deverá retornar *False*.

2. Sejam as funções abaixo escritas em Haskell sem qualquer erro de sintaxe ou lógica:

```
proximo::[(Char,Int)]->(Char,Int)
proximo [t] = t
proximo ((a,b):(c,d):x)
  |b<d = proximo ((a,b):x)
  |otherwise = proximo ((c,d):x)

realizado::(Char,Int)->[(Char,Int)]->[(Char,Int)]
realizado _ [] = []
realizado (k,m) ((a,b):x)
  |m == b = x
  |otherwise = (a,b):(realizado (k,m) x)

f01::[(Char,Int)]->String
f01 [] =  []
f01 x = (fst (proximo x)):(f01 (realizado (proximo x) x))
```

   (a) O que a função *proximo* faz? Dê um exemplo

   (b) O que a função *realizado* faz? Dê um exemplo

   (c) O que a função $f01$ faz? Dê um exemplo

   (d) De acordo com o código, a função *proximo* é chamada recursivamente e também externamente pela função $f01$. O que garante que esta função, de acordo com o código, não precisa de ter a lista vazia na base para funcionar corretamente?

   (e) Em *proximo*, quais os tipos de t, a, b, c, e x?

3. Valores do tipo [[Int]], juntamente a uma String *padrão* P são alvo de uma computação a ser realizada. Cada lista [Int] deve ser transformada em uma String com as letras do alfabeto definidas pelas posições indicadas pelos Int, que iniciam o mapeamento de $0 \rightarrow' a'$ até $25 \rightarrow' z'$. Para cada nova String N gerada, caso algum caracter de N esteja presente em P, a String N fará parte do resultado da computação. Caso contrário, N deverá ser eliminada da solução. Para qualquer dúvida, suponha a computação com os dados de entrada abaixo:
   entrada: [[0,1,0,2,0,19,4],[21,23,24],[],[22,22,22,22,0,0,25,25,2]] "bw"

   saída: ["abacate","wwwwaazzc"]

4. Usando *list comprehension*, implemente uma função que receba [String] e retorne [(Int,String)], em que o inteiro é o tamanho da string e a string é a mesma dada na lista inicial.

5. Usando a função *map* definida no *Prelude do Kaskell*, faça uma função que receba uma lista de duplas de lista de inteiros e um *booleano* [([Int],Bool)]. Caso o valor lógico de cada dupla seja verdadeiro, a lista de inteiros é colocada na solução de sua função sem nenhuma alteração. Caso contrário, a lista de inteiros deverá perder todos os elementos em posição ímpar da lista antes de fazer parte da solução da função. Veja o exemplo: (Observação: o primeiro elemento da lista está na posição zero)

   suaFuncao [([2,1,0,4],True),([9,0,1,1,2,7], False)] retorna [[2,1,0,4],[9,1,2]]

```
map f [] = []
map f (a:b) = (f a):map f b
```

6. Faça uma função que receba uma lista de inteiros e retorne a lista sem todos os números pares que são precedidos por pelo menos um número ímpar. Veja o exemplo:

   suaFuncao [2,4,3,5,8,4,5,7,6,9,2,4,6] retorna [2,4,3,5,4,5,7,9,4,6]

7. Faça uma função que receba como entrada uma lista de inteiros e outra lista de inteiros resultado da computação da função da questão anterior. Como saída, esta função retornará uma dupla com a lista do segundo parâmetro e uma lista de posições que foram excluídas na lista original. Veja o exemplo:

   suaFuncao [2,4,3,5,8,4,5,7,6,9,2,4,6] [2,4,3,5,4,5,7,9,4,6] retorna ([2,4,3,5,4,5,7,9,4,6], [4,8,10])

8. Uma empresa matriz gerencia as várias filiais. Parte desta gerência consiste em: saber a produção anual por semana de cada filial, fechar a filial fora dos critérios de produtividade; e promover outras candidatas a filiais caso os critérios para promoção sejam alcançados. Veja os critérios que a matriz usa para gerir todos esses processos:

   - Cada filial ou candidata a filial (chamadas de empresa) tem um código String que a identifica. A matriz, além disso, armazena em outro código para distinguir entre filial registrada ou candidata a filial;
   - As empresas devem enviar, todo fim de ano, o valor de produção semanal à matriz. Cada semana do ano gera uma informação do tipo inteiro;
   - A matriz tem uma lista de expectativa mínia de produção para cada semana. Esta lista base é usada para medir a produção anual de cada empresa;
   - Uma filial é fechada quando ela não alcança a meta em seis semanas consecutivas no ano ou quando ela teve produção igual a zero em pelo menos seis semanas do ano. As candidatas

a filiais são promovidas se alcançarem as mesmas metas que mantêm as empresas filiais ativas.

- Finalmente, a matriz gerencia toda a informação em uma estrutura de dados da seguinte forma: [(String,Bool,[Int])], em que o primeiro campo é o código da empresa, o segundo é verdadeiro quando a empresa é filial e falso se for candidata e, por último, a lista de inteiros armazena um inteiro para cada semana do ano.

Fazendo uso da estrutura de dados que a matriz tem, faça em Haskell:

(a) Faça uma função que retorne **todas as filiais** da matriz. Faça uma função que retorne **todas as candidatas** a filiais;

(b) Faça uma função que permita saber se uma empresa alcançou ou não a meta anual, definida anteriormente;

(c) Faça uma função que informe, para **todas** as empresas, se cada uma alcançou ou não a meta;

(d) Faça uma função que retorne as empresas que passarão de candidatas a filiais para filiais.

(e) Defina um tipo diferente de [(String,Bool,[Int])] que permite gerenciar as informações das empresas sem repetir o segundo campo. Sua solução deve ser suficiente para realizar as mesmas computações que sua implementação computa. (Ou seja, todo que a matriz necessita saber não pode ser omitido em sua nova estrutura de dados proposta.

9. Faça uma solução em Haskell que receba uma String e retorne a quantidade de dígitos numéricos presentes nela. Como exemplo, se a entrada for a string "ab3ukfe5" a resposta será 2.

10. Faça uma solução em Haskell que receba uma lista L de valores do tipo Int e um Int x e informe se o valor do elemento de L de índice x são iguais. Os índices da lista começam pelo valor 0 e, caso x não seja um índice válido, como um valor negativo ou maior que a lista, a resposta deverá ser *False*. Como exemplo, para as entradas [2,5,4,3,7,8,3,9] e 3, a resposta é True.

11. Faça uma solução em Haskell que receba uma lista de duplas de inteiros e um valor inteiro e retorne uma lista de Bool informando, para cada dupla de inteiros, quais que somados ultrapassam o parâmetro fornecido como entrada. Como exemplo, se passados [(2,1),(3,5),(5,1),(1,9)] 7 tem como solução [False, True, False, True]

12. Faça uma solução em Haskell que receba dois inteiro x e y. A Função deverá retornar uma dupla de listas de Inteiros sendo que os elementos da primeira lista são divisíveis por x e os da segunda lista não são divisíveis por x no intervalo fechado entre x e y Como exemplo, se as entradas forem 3 e 13, a resposta será ([3,6,9,12],[4,5,7,8,10,11,13])

13. Faça uma solução em Haskell que receba uma String S e retorne outra que é resultado de S concatenada o número de vezes seu tamanho. Como exemplo, se a entrada for "Ela" a saída será "ElaElaEla"

14. Sobre Programação Funcional, responda:

    (a) O que é função de alta ordem?

    (b) Qual vantagem existe em usar funções de alta ordem?

    (c) O que é avaliação preguiçosa presente no Haskel? Cite um exemplo que quando pode ocorrer.

    Veja as definições em um módulo Haskel abaixo:

```
type Pessoa: [(Id, Nome, Telefone)]
type Conhece [(Id,[Id])]
```

15. Sobre estas definições, faça:

    (a) Uma solução que permita dizer, para um nome de pessoa fornecido, quantas pessoas ela conhece;

    (b) Uma solução que permita dizer, para um nome de pessoa fornecido, todos nomes de pessoas que a conhecem;

    (c) Uma solução que permita dizer, para um nome de uma pessoa fornecido, seu telefone;

    (d) Uma solução que permita dizer, para um nome de uma pessoa fornecido, todos nomes de pessoas que ela não conhece;

    (e) uma solução que permita dizer todas as pessoas que não são conhecidas por ninguém.

16. Faça, em Haskell, uma solução que permita receber uma lista l de duplas d de Int e String e retorne uma String contendo a concatenação dos caracteres de cada String de d que ocupam a posição indicada pelo Int de sua dupla.

    exemplo: f [(2,“abobora”),(0,“alicate”),(1,“ainda”)] retorna “oai”

17. Faça, em Haskell, uma solução que permita retornar a lista dos números Naturais.

18. Faça, em Haskell, a função filtraElimina de ordem superior que permita filtrar e excluir de uma lista [Int] os números selecionados por uma função passada como parâmetro. Seja, neste caso, $f::Int \rightarrow Bool$ um parâmetro para filtraElimina. Neste caso, filtraElimina exclui um inteiro da lista quando f retorna *True*. Porém, a computação de f foi encapsulada por outro programador em um módulo Haskell e não é conhecida por você.

19. Faça, em Haskell, uma solução para o seguinte problema: Receber uma lista l de inteiros e retornar uma tupla de listas de inteiros sendo a primeira os divisores de dois presentes em l, a segunda os divisores de três presentes em l e, finalmente, a terceira, os demais.

    exemplo: f [1,5,4,7,6,9] retorna ([4,6],[6,9],[1,5,7])

20. Uma lista de listas de inteiros possui valores importantes para seu proprietário. A todo mês, ele precisa de uma lista que seja formada pelos maiores números de cada uma das listas de inteiros que são elementos da lista maior. Para ficar mais claro, veja o exemplo:

    dada a lista: [[2,4,**31**,7],[6,5,**98**,30,1],[43,**67**,4,10,4],[**1**,1,1,1],[7,6,8,**56**,3,5]] o proprietário precisará dos valores em negrito em uma lista separada. Assim sendo, implemente uma solução para esse problema.

21. Faça em Haskell uma função que, dados um Char e uma lista de Char, retorne uma lista de Bool que indica, a cada posição, se o Char se encontra (True) ou não (False) na lista Exemplo: testa 'a' ['a','u','y','a','a','l'] retorna [True, False, False, True, True, False]

22. Faça uma função, em Haskell, utilizando list comprehension, que faça a seguinte computação: Receber uma lista de inteiros e um inteiro. A função deverá retornar uma lista de duplas de inteiros onde cada dupla é formada pelo elemento da lista com o inteiro passado por parâmetro desde que o elemento da lista seja maior que o inteiro. Ex: geraListaDupla [9,4,3,7] 5 deverá retornar [(9,5),(7,5)]

23. Defina uma solução em Haskell que, dada uma lista x, retorna uma lista com os elementos repetidos em x. Ex: repetidos [1,2,1,1,1,3,4,3,1] = [1,3] É importante que a lista resultante da sua solução NÃO tenha, em hipótese alguma, elementos repetidos.

24. Utilizando *list comprehension*, defina a função em Haskell chamada termina em ::Int->[Int] que recebe um número n e devolve a lista com os números entre 0 e 100 que terminam com o númer n. Veja o exemplo:

    termina em 3 = [3,13,23,33,43,53,63,73,83,93]

    termina em 7 = [7,17,27,37,47,57,67,77,87,97]

25. Faça a sequência de questões:

    (a) Implemente, em Haskell, o operador &&&, infixo (infix) com precedência 7, que realize a seguinte tarefa: Dados um inteiro x e uma dupla de inteiros k, retornar o inteiro de k que mais se aproxima de x. Veja o exemplo: 7 &&& (9,0) terá como resultado 9 pois a distância de 9 a 7 é 2 enquanto a distância de 0 a 7 é 7. (&&&) 7 (6,15) terá como resultado 6.

    Obs: Faça quantas funções se fizerem necessárias. Outra informação importante é que, se a distância dos números da dupla ao inteiro x for a mesma, você poderá retornar qualquer um dos números da dupla. Veja o exemplo (&&&) 7 (6,8) pode retornar 6 ou 8.

    (b) Faça uma função de ordem superior em Haskell que receba como parâmetros: uma função f do tipo Int->(Int,Int)->Int, um Inteiro x e uma lista de duplas de inteiros listaDuplas. Esta função deverá aplicar todas as duplas da lista listaDuplas à função recebida como parâmetro f e o inteiro x e retornar uma lista de inteiro resultantes de cada aplicação realizada.

(c) faça a chamada da função do item b) passando o operador do item a), um inteiro de sua preferência e uma lista com, pelo menos, três duplas de sua preferência. Rastreie a execução e deixe claro o resultado.

26. Faça, em Haskell, a função mescla que receba duas listas a e b de caracteres. A função deverá retornar uma lista de duplas com cada caracter da lista a e o número de vezes que esses caracteres aparecem na lista b. Não deverá repetir um caracter caso ele apareça mais de uma vez na lista a. Veja o exemplo: mescla [a, y, a, u, p, y, u] "estou fazendo a prova final de haskell e vou passar"deverá retornar [(a, 7), (y, 0), (u, 2), (p, 2)]

27. Implemente, em Haskell, o operador -*-, infixo (infix) com precedência 3, que realize a seguinte tarefa: Dadas duas duplas de inteiro x e k, retornar uma dupla de inteiros com o maior e o menor entre todos, nesta ordem. Veja o exemplo: (7,8) -*- (9,0) retornará (9,0). (-*-) (7,0) (6,15) terá como resultado (15,0).

    Para a resolução desta função você deverá fazer chamadas às funções maior e menor definidas com o seguinte cabeçalho: Int->Int->Int. É necessário implementar maior e menor também.

28. Resolva a sequência de funções que se seguem:

    (a) Faça a função *ocorrencia* em Haskell que receba um inteiro x e uma lista l do tipo [Int]. Essa função deverá retornar uma dupla compreendida pelo inteiro x e o número de vezes que este aparece na lista l.

    (b) Faça a função *aplica* de ordem superior em Haskell que receba como parâmetros: uma função f::Int->[Int]->(Int,Int) e uma lista de inteiros K. A função *aplica* deverá computar, a cada passo, a aplicação da cabeça da lista k e a própria lista k à função f e retornar uma lista de duplas de inteiros resultantes de cada computação realizada até que todos elementos de k sejam computados por f.

    (c) Mostre como utilizar a função ocorrencia, item b), como parâmetro da função aplica, item a), para que a função aplica retorne uma lista de duplas com os elementos da lista de inteiros que ela receber e suas respectivas ocorrências.

29. Escreva uma definição equivalente à exibida abaixo, mas usando apenas uma única cláusula simples.

```
funny x y z
  | x > z = True
  | y >= x = False
  | otherwise = True
```

30. Defina um operador em Haskell que, dado um inteiro e uma lista, o operador retorne uma lista retirando uma ocorrência do inteiro na lista. Exemplo: se o operador receber 4 e [5,4,7,4,8] este retornará [5,7,4,8]. Observe que é necessário excluir apenas a primeira ocorrência.