Teoria de Grafos Trabalho 1

Exercícios

Todos os exercícios devem ser feitos em Haskell. Coloque cada solução em um arquivo próprio. Por exemplo, o exercício 1 deve estar no arquivo `ex1.hs`, o exercício 2 em `ex2.hs` e assim por diante.

Não use biblioteca alguma que implemente diretamente as funções pedidas.

1. Escreva uma função de nome `bissexto` que receba um ano e devolva `True` se ele for bissexto ou `False` em caso contrário.

```
> bissexto 2021
False

> bissexto 2000
True

> bissexto 2004
True
```

2. Escreva uma função de nome `acronimo` que receba uma sentença na forma de uma string e devolva outra string contendo o acrônimo da sentença dada.

```
> acronimo "National Aeronautics and Space Administration"
"NASA"

> acronimo "Cadastro Nacional de Estabelecimentos de Saúde"
"CNES"

> acronimo "Organização do Tratado do Atlântico Norte"
"OTAN"
```

3. Escreva uma função chamada `romano` que receba um número natural positivo e devolva uma string representando o número recebido em numeração romana.

```
> romano 21
"XXI"

> romano 800
"DCCC"

> romano 2021
"MMXXI"
```

4. Escreva uma função chamada `enésimoPrimo` que receba um número natural positivo n e devolva o enésimo número primo.

```
> enésimoPrimo 21
73

> enésimoPrimo 167
991

> enésimoPrimo 901
7001
```

5. Escreva uma função chamada `fatoresPrimos` que receba um número natural positivo e devolva uma lista contendo seus fatores primos.

```
> fatoresPrimos 1
[]

> fatoresPrimos 2
[2]

> fatoresPrimos 9
[3, 3]

> fatoresPrimos 12
[2, 2, 3]

> fatoresPrimos 901255
[5, 17, 23, 461]
```

6. Escreva uma função de nome `mapMat` que receba uma função f e uma matriz m com elementos de um tipo genérico e devolva outra matriz com o resultado de aplicar f em cada elemento de m.

```
> import Data.Array

> mat = array ((1,1),(2,3)) [((1,1),4),((1,2),0),((1,3),8),
                             ((2,1),7),((2,2),1),((2,3),7)]

> do print mat
array ((1,1),(2,3)) [((1,1),4),((1,2),0),((1,3),8),
                     ((2,1),7),((2,2),1),((2,3),7)]

> quadrado x = x * x

> mat2 = mapMat quadrado mat

> do print mat2
array ((1,1),(2,3)) [((1,1),16),((1,2),0),((1,3),64),
                     ((2,1),49),((2,2),1),((2,3),49)]
```

Nota: Talvez seja de ajuda consultar os materiais disponíveis em

- http://www.facom.ufu.br/~madriana/PF/Haskell7.pdf

- http://www.facom.ufu.br/~madriana/PF/TP6-Vetores.pdf

7. Escreva uma função de nome `matMult` que receba duas matrizes $m1$ e $m2$ devolva outra matriz com o resultado da multiplicação de $m1$ por $m2$. Use a biblioteca `Data.Array`, como no exercício anterior.

   Para construir as matrizes de teste, você pode usar a função `matriz` que recebe uma lista de listas e a transforma em uma matrix , como mostrado no exemplo a seguir.

```
import Data.List
import Data.Array

matriz mat = array ((1,1), (n,m)) (aplaina 1 mat)
  where
    n = length mat
    m = length (mat !! 0)
    linha i xs = [((i,j), xs !! (j - 1) ) | j <- [1 .. m]]
    aplaina i [] = []
    aplaina i (l:ls) = linha i l ++ aplaina (i+1) ls
```

```
> mA = matriz [ [ 3, 1],
                [-1, 2],
                [-4, 4] ]

> mB = matriz [ [5,  6, -3,  7],
                [1, -1,  0, -2] ]

> matMult mA mB
array ((1,1),(3,4)) [((1,1), 16), ((1,2), 17), ((1,3),-9), ((1,4), 19),
                     ((2,1), -3), ((2,2), -8), ((2,3), 3), ((2,4),-11),
                     ((3,1),-16), ((3,2),-28), ((3,3),12), ((3,4),-36)]

> matMult mB mA
array *** Exception: dimensões incompatíveis
```

8. Escreva uma função de nome `combinações` que receba uma lista xs com elementos de um tipo genérico e um número natural positivo k e devolva uma lista contendo todas as combinações dos elementos de xs tomados k a k.

```
> import Data.List

> combinações [1, 2, 3, 4] 2
[ [1,2], [1,3], [2,3], [1,4], [2,4], [3,4] ]

> combinações "abcdef" 4
[ "abcd", "abce", "abde", "acde", "bcde", "abcf", "abdf", "acdf",
  "bcdf", "abef", "acef", "bcef", "adef", "bdef", "cdef" ]

> combinações [1..6] 3
[ [1,2,3], [1,2,4], [1,3,4], [2,3,4], [1,2,5], [1,3,5], [2,3,5],
  [1,4,5], [2,4,5], [3,4,5], [1,2,6], [1,3,6], [2,3,6], [1,4,6],
  [2,4,6], [3,4,6], [1,5,6], [2,5,6], [3,5,6], [4,5,6] ]
```

9. Escreva uma função de nome `sorteia` que receba um número natural n e dois números inteiros `início` e `fim`. A função deve sortear n pares de números inteiros entre `início` e `fim`, inclusive, e devolvê-los em uma lista.

```
> sorteia 3 1 10
[ (9,2), (6,10), (2,1) ]

> sorteia 3 1 10
[ (5,4), (7,10), (4,8) ]

> sorteia 6 0 50
[ (45,42), (15,28), (1,45), (9,48), (17,49), (47,0) ]
```

10. Crie uma função chamada **representação b n** que receba um número natural positivo `b`, representando uma base, e um número natural `n`, representando um natural decimal, e devolva uma lista com a representação de `n` na base `b`. Para representar dígitos acima de 9, use a representação decimal deles. Assim, os dígitos da base 12 serão $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$. Alguns exemplos:

```
> representação 2 12     -- 12 na base binária
[1,1,0,0]

> representação 3 12     -- 12 na base 3
[1,1,0]

> representação 10 537   -- 537 na base 10
[5,3,7]

> representação 19 211   -- 211 na base 19
[11,2]

> representação 12 12345 -- 12345 na base 12
[7,1,8,9]
```

11. Crie uma função de nome **completa xs n** que receba uma lista `xs` de números inteiros e se `xs` possuir menos que `n` elementos, completa o início da lista com 0s. Caso contrário, devolve `xs` sem alterações. Exemplo:

```
>  completa [1,1,1] 5
[0,0,1,1,1]

 > completa [1,1,1] 2
[1,1,1]
```

12. Crie uma função chamada **conta b larg min max** que receba uma natural positivo `b`, representando uma base, e dois números naturais positivos: `larg`, `min` e `max`. A função devolve uma lista com todos os números entre `min` e `max`, inclusive, representados na base `b` e com largura dada por `larg`. Por exemplo

```
>  conta 2 3 1 7   -- conta de 1 a 7 em binário usando números com 3 dígitos de largura
[ [0,0,1], [0,1,0], [0,1,1], [1,0,0], [1,0,1], [1,1,0], [1,1,1] ]
```