UFU/FACOM/BSI
Disciplina: AARE/PF
Período: 2020/Etapa 1
Ref: Primeiro Trabalho de Programação

Objetivo:

O objetivo deste trabalho é ajudá-lo a aprender mais sobre como definir simples funções em HASKELL e introduzir a idéia de desenvolver definições por decompor o problema original em subproblemas de dificuldade inferior ao problema dado.

Tarefa: *O Dia da Semana*

Você deve escrever uma função que, dada uma determinada data do século XXI descrita no formato (dia,mês,ano), obtém o dia da semana correspondente. Por exemplo:

`Main> diaDaSemana (1,1,2001)` retorna `Segunda`
`Main> diaDaSemana (2,1,2001)` retorna `Terca`

Como tal problema deve ser abordado?

Existe uma fórmula complicada chamada *Congruência de Zeller* que resolve o problema. Presumo que você não saiba disto. Desta forma vamos tentar uma outra abordagem para resolver o problema.

Se nós soubéssemos o primeiro dia do século XXI e pudéssemos obter o número total de dias até a data que estamos interessados, isto poderia ajudar muito. Existem alguns aspectos a serem considerados; o problema de anos bissextos é um deles.

Para começar aqui estão algumas declarações de sinônimos que tornarão mais legível nosso programa. Uma declaração de tipo sinônimo como aparece a seguir é possível pela construção `type`.

```
type Dia = Int
type Mes = Int
type Ano = Int
type Data = (Dia,Mes,Ano)
```

Observe que após estas declarações se poderia definir qualquer função que use o tipo inteiro no cabeçalho escrevendo-se por exemplo `Dia` no lugar de `Int`.

Continuando com o procedimento para se resolver o problema Inicialmente você deve escrever uma função chamada `bissexto` que determina se um dado ano é bissexto. Veja a seguir alguns exemplos de chamada da função `bissexto`.

```
Main>bissexto 2096 => True
Main>bissexto 2097 => False
```

Desde o ajuste no calendário realizado pela papa Gregório em 1752 as seguintes regras são usadas para se verificar se um dado ano é bissexto (ano com 366 dias):
Um ano divisível por 4 é um ano bissexto,
mas se ele for divisível por 100 deixa de ser bissexto.
Contudo ele volta a ser bissexto se for divisível por 400.

Implemente a função `bissexto`. Antes de continuar certifique-se que ela funciona adequadamente. Após implementar uma função SEMPRE teste-a para certificar-se que de fato ela faz aquilo que você espera antes de usá-la em outra função.

Vamos tratar agora do problema de encontrar o número de dias em cada mês. Você deverá usar as seguintes funções:
1 - A função `numDeDiasEmCadaMesDeUmAno:` dado um ano devolve o número de dias em cada mês daquele ano:
Por exemplo:
```
Main>numDeDiasEmCadaMesDeUmAno 2097
```
devolve
```
[31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31]
```

Definição:
```
numDeDiasEmCadaMesDeUmAno :: Ano -> [Int]
numDeDiasEmCadaMesDeUmAno ano =
 [31,feb,31,30,31,30,31,31,30,31,30,31]
 where feb
        |bissexto ano = 29
        |otherwise    = 28
```

2 - a função `numDeDias:` fornecida uma data, fornece o número de dias desde 31 de Dezembro de 2000.

Definição:
```
numDeDias :: Data -> Int
numDeDias (dia,mes,ano) =
     dia -- dias deste mes
     + sum (take (mes-1) (numDeDiasEmCadaMesDeUmAno ano))
     + (ano-2001)*365 + (ano-2001)`div`4
```

Esta função funciona da seguinte forma: considere a seguinte expressão a ser avaliada:

```
Main>numDeDias (18,7,2097)
```

Considerando-se esta chamada tem-se que a última linha que define `numDeDias` especificada anteriormente obtém o número de dias de 2001 a 2096 admitindo que cada ano tem 365 dias e o soma com os dias resultantes de anos bissextos. A segunda linha usa `numDeDiasEmCadaMesDeUmAno` para obter número de dias do corrente ano. Finalmente a primeira linha adiciona 18 ao número de dias até então. Por exemplo, avaliando

```
Main>numDeDias (18,7,2097) devolve 35263
```

Observe que a função `numDeDias` usa a função `numDeDiasEmCadaMesDeUmAno` e a função `bissexto`. Quando você escreve programas, esta é a forma que você deve trabalhar, ou seja, criar funções mais complicadas pela combinação de funções mais simples. O resto do desenvolvimento do programa segue esta abordagem usando a função `numDeDias` definida anteriormente. O que necessitamos agora é definir uma função, digamos

```
nomeDoDia :: Int -> String
```

que dado um número na faixa de 0 a 6 retorne o nome do dia. Por exemplo, avaliando:

`Main>nomeDoDia 0` devolve `"Domingo"`
`Main>nomeDoDia 1` devolve `"Segunda"`

Sugiro que você use padrões dados por constantes na definição de `nomeDoDia`. Uma vez que você tenha esta função funcionando adequadamente, você deve ser capaz de usar `numDeDias`, `nomeDoDia`, `rem` e o fato de que 31 de Dezembro de 2000 caiu num Domingo para construir a função

```
diaDaSemana :: Data -> String
```

Eis algumas dicas e sugestões:
Inicialmente, tente escrever uma função que funcione adequadamente para datas entre (1,1,2001) e (6,1,2001). Então modifique-a de tal forma que ele funcione genericamente. Lembre-se que sua função não tem que funcionar para datas anteriores a (1,1,2001) e posteriores a (31,12,2100). Você pode usar o comando `cal` do Unix/Linux para verificar as respostas dadas pela sua função.