Ensino de Programação – MEINF

Exame

25 de Fevereiro de 2014 (10h00)

Dispõe de 2:00 horas para realizar este teste.

Questão 1 (7v = 1+2+2+2)

Responda às alíneas seguintes:

a) Defina uma função em Haskell que calcula a lista com todos os divisores de um dado número inteiro. Por exemplo:

```
divisores :: Int -> [Int]
> divisores 18
= [1,2,3,6,9,18]
```

b) Especifique uma função em Haskell que calcula o mínimo múltiplo comum de dois números inteiros. Por exemplo:

```
mmc :: Int -> Int -> Int
> mmc 16 24
= 48
```

c) Especifique uma função em Haskell que calcula a interceção de duas listas de inteiros. Por exemplo:

```
intercepcao :: [Int] -> [Int] - > [Int]
> intercepcao [1,2,3,6,9,18] [1,2,3,4,6,8,12,24]
= [1,2,3,6]
```

d) Especifique uma função que calcula o mínimo múltiplo comum de uma lista de inteiros. Exemplo:

```
mmclist :: [Int] -> Int
> mmclist [4,5,10]
= 20
```

Questão 2 (9v = 1+2+1+1+2+2)

Vamos representar horas por um par de números inteiros:

```
type Hora = (Int, Int)
```

Assim o par `(0,15)` significa *meia noite e um quarto* e o par `(13,45)` significa *duas menos um quarto*. Defina funções em Haskell para:

a) testar se um par de inteiros representa uma hora do dia válida;

```
horaValida :: Hora -> Bool
> horaValida (25,12)
= False
```

b) testar se uma hora é ou não maior que outra (comparação);

```
compHora :: Hora -> Hora -> Bool
> compHora (15,12) (10,43)
= True
```

c) converter um valor em horas (par de inteiros) para minutos (inteiro);

```
hora2Min :: Hora -> Int
> hora2Min (1,12)
```

```
= 72
```

d) converter um valor em minutos para horas;

```
min2Hora :: Int -> Hora
> min2Hora 72
= (1,12)
```

e) calcular a diferença entre duas horas (cujo resultado deve ser uma Hora);

```
difHoras :: Hora -> Hora -> Hora
> difHoras (15,10) (20,50)
= (5,40)
```

f) adicionar um determinado número de minutos a uma dada hora.

```
somaMin :: Hora -> Int -> Hora
> somaMin (15,10) 53
= (16,3)
```

Questão 3 (4v = 1+1+2)

Considere as seguintes definições de tipos de dados para os tipos numéricos fração e número misto. Uma fração é composta por um par de inteiros, o numerador e o denominador. Por sua vez, um número misto é constituído por um inteiro e uma fração própria (o numerador é menor do que o denominador):

```
data Frac = Fracao Int Int
deriving Show
data NumMisto = NMisto Int Frac
```

```
deriving Show
```

e especifique em Haskell as funções que se descrevem a seguir:

a) Especifique um função que recebe uma fração e a converte num número misto:

```
frac2Misto :: Frac -> NumMisto
> frac2Misto (Fracao 22 6)
= NMisto 3 (Fraccao 4 6)
```

b) Especifique a função inversa da anterior, ou seja, recebe um número misto e transforma-o numa fração:

```
misto2Frac :: NumMisto -> Frac
> misto2Frac (NMisto 3 (Fraccao 4 6))
= Fraccao 22 6
```

c) Especifique uma função que calcula a soma de dois número mistos:

```
somaMisto :: NumMisto -> NumMisto -> NumMisto
> somaMisto (NMisto 3 (Fraccao 4 6)) (NMisto 1 (Fraccao
2 5))
= NMisto 5 (Fraccao 1 15)
```