Universidade Federal de Uberlândia - FACOM - BSI
Programação Funcional - Primeira Avaliação
4 Maio 2009

Nome: _____

- A prova terá duração de 1h40min.
- Não será permitida consulta à documentos durante a prova.
- Certifique-se de ter respondido à todas as questões de forma clara.

1. (6 pontos) Sejam as funções Haskell definidas nos itens (a) e (b):

(a)
```
func [] = []
func ((x,y):ps) = x : func ps
```

1. Descreva o objetivo da função.
2. Defina uma declaração de tipos, de modo que as entradas e saídas de dados contemplem apenas valores inteiros.
3. A partir da nova declaração de tipos, forneça o resultado da chamada da função, e mostre o traço da execução (passo a passo).

```
Main> func [(1,3),(5,6),(3,3),(7,2)]
```

(b)
```
fg::Int->[Int]->Bool
fg x [] = False
fg x (y:z) | x == y = True
           | otherwise = fg x z
```

1. Descreva o objetivo da função.
2. Especifique passo a passo e forneça o resultado da seguinte execução:

```
Main> fg 5 [1,5,3,5,9,5]
```

Solução:

Parte A

(1) A função retorna uma lista contendo as primeiras posições das tuplas contidas na lista de entrada.

(2) A declaração seguinte completa o código do programa:

```
func::[(Int,Int)]->[Int]
```

Programa completo:

```
func::[(Int,Int)]->[Int]
func [] = []
func ((x,y):ps) = x : func ps
```

(3)
```
func [(1,3),(5,6),(3,3),(7,2)]
(1: func [(5,6),(3,3),(7,2)])
(1: (5: (func [(3,3),(7,2)])
(1: (5: (3 : func [(7,2)])))
```

```
(1: (5: (3 : (7: func [])))
(1: (5: (3: (7 : []))))
[1,5,3,7]
```

Parte B

(1) A função verifica se um elemento pertence à uma lista.

(2)
```
fg 5 [1,5,3,5,9,5]
fg 5 [5,3,5,9,5]
True
```

2. (6 pontos) Escreva uma função Haskell com a seguinte declaração de tipos:

ehDecrescente :: [Int] -> Bool

A função deve retornar *True* caso a lista de entrada (contendo números inteiros) esteja em ordem descrescente, e *False* caso contrário. Os casos especiais *lista vazia* e *lista de um elemento* são considerados como estando em ordem decrescente.

Exemplo de interpretação da função:

```
Main> ehDecrescente [317,67,12,-3,-5,-10]
True
```

Solução:

```
ehDecrescente::[Int]->Bool
ehDecrescente [] = True
ehDecrescente [x] = True
ehDecrescente (x:y:xz) | x > y = ehDecrescente(y:xz)
                       | otherwise = False
```

3. (6 pontos) Numa conferência científica, os coordenadores do evento calculam o valor de uma inscrição baseado no número de mini-cursos escolhidos pelo participante e se o participante é aluno, professor ou profissional.

- Para alunos, deve ser cobrado R$90,00 por mini-curso. Porém, se o aluno se inscreve em mais de 3 mini-cursos ele terá desconto de 30%, e, neste caso cada um custará R$63,00.
- Para professores, o preço de cada mini-curso é R$300,00.
- Para profissionais, cada mini-curso custa R$450,00.

Para simplificação, considere valores inteiros para representar alunos, professores e profissionais:

1 Alunos
2 Professores
3 Profissionais

Escreva uma função Haskell compatível com a declaração:

valorInscr :: Int -> Int -> Int

Por exemplo, sua função deve retornar:

```
Main> valorInscr 1 4
252
Main> valorInscr 2 5
1500
```

Solução:

```
valorInscr::Int->Int->Int
valorInscr x y | x == 1 && y <= 3 = y*90
               | x == 1 && y > 3 = y*63
               | x == 2 = y*300
               | x == 3 = y*450
               | otherwise = 0
```

4. (6 pontos) Seja a função Haskell abaixo que verifica se um número é *par*:

```
par::Int->Bool
par x = if mod x 2 == 0 then True else False
```

Crie uma nova função Haskell que recebe uma lista de números inteiros, e separa os elementos pares da lista, retornando-os numa nova lista. Por exemplo:

```
Main>separaPares [4,3,5,2,6,8,1]
[4,2,6,8]
```

Solução:

```
separaPares::[Int]->[Int]
separaPares[] = []
separaPares (a:t) | par a = a:separaPares t
                  | otherwise = separaPares t
```

5. (6 pontos) Seja uma relação de produtos a serem vendidos em uma banca de revistas:

123	Epoca	6,90
322	Veja	8,90
452	Info	5,60
113	Exame	7,20

A declaração de tipos abaixo pode ser utilizada para a definição de um programa Haskell que manipula tais dados:

```
type Item = (Int, [Char], Float) -- Numero, Descricao do item, Preço de venda
type ListaProdutos = [Item]
```

Utilizando a declaração de tipos acima:

1. Crie uma lista de produtos contendo 4 tuplas-3, que representam as revistas e suas informações respectivas.
2. Faça uma função que retorne o número do item de maior preço na lista.

Solução:

```
type Item = (Int,[Char],Float)
type ListaProdutos = [Item]

novaLista::ListaProdutos
novaLista = [(123,"Epoca",6.90),(322,"Veja",8.90),(452,"Info",5.69),
             (113,"Exame",7.20)]

numMaior::ListaProdutos->Int
numMaior[(a,b,c)] = a
numMaior((a1,b1,c1): ((a2,b2,c2):xs)) | c1>c2 = numMaior((a1,b1,c1):xs)
                                      | otherwise = numMaior((a2,b2,c2):xs)
```

Questão	Pontos	Nota
1	6	
2	6	
3	6	
4	6	
5	6	
Total:	30	