Programação Funcional

Exame 517

26 de Maio de 2017

DCC/FCUP

Nome: _____

Nº mecanográfico: _____

- **Duração: 2h + 30m tolerância.**
- **Este exame contém 7 questões e 4 páginas.**
- **Responda às questões no espaço marcado no enunciado.**
- **Pode usar funções auxiliares e/ou do prelúdio-padrão de Haskell.**
- **Nas questões 2 a 7, indique sempre o tipo da função definida.**

1. (6 valores) Responda a cada uma das seguintes questões, indicando **apenas** o resultado de cada expressão.

(a) `tail ([1,2]:[]:[[5,2]])` = _____

(b) `take 4 [1,4..]` = _____

(c) `(length.concat) [[1,2,3,4],[5,6,7,8],[2,3,4]]` = _____

(d) `filter (\x -> x‘mod‘3 ==0) [0..9]` = _____

(e) `foldr (:) [2] [1,3..7]` = _____

(f) `[(x,y) | x <- [1..4], y <- [x..4], x+y == 4]` = _____

(g) Sem usar explicitamente a lista dada, defina a seguinte lista:

`[1,4,7,10,13,16,19,22,25,28,31]` = _____

(h) Considere a seguinte definição em Haskell:

```
f (x:xs) = (2+x) * f xs
f [x] = 3+x
f [] = 1
```

A avaliação da expressão `f [3,2,1]` tem como resultado: _____

(i) Indique um tipo admissível para a função `f` definida como `f x = x!!3`: _____

(j) Indique o tipo mais geral de `[(!!2),length]`: _____

(k) Considere as seguintes definições:

```
data N = Z | S N

ntoI Z = 0
ntoI (S n) = (ntoI n) + 1
```

Indique um tipo admissível para a função `ntoI`: _____

(l) Indique o tipo mais geral de `(\x -> map (>x))` :

_____

2. (3 valores) Nas duas alíneas seguintes, pode utilizar funções do prelúdio-padrão e/ou listas em compreensão mas não deve usar directamente **recursão**.

(a) Defina uma função `maioresQ` que, dada uma lista de valores `l` e um valor `x`, retorna a lista de valores de `l`, que são maiores do que `x`.

(b) Defina a função `tamanhoS` que, dada uma lista de strings como argumento, retorna a lista dos seus comprimentos. Por exemplo, `tamanhoS ["Hoje", "é", "um", "lindo","dia"] = [4, 1, 2, 5, 3]`.

3. (3 valores) Considere uma representação de relações binárias como matrizes binárias, representadas como listas de listas. Note que uma relação binária R entre dois conjuntos A e B, pode representar-se como uma matriz (de 0's e 1's) M, tal que $M[i,j] = 1$ se e só se (i,j) pertence a R.

(a) Implemente uma função `timesMat` que dada uma matriz binária de `m` por `n` e uma matriz binária de `n` por `r`, determina a matriz binária de `m` por `r` correspondente ao produto (módulo 2) das duas matrizes.

(b) Implemente uma função `transitiva`, que dada a representação de uma relação como uma matriz quadrada, determina se a matriz é transitiva. Recorde que, uma relação binária R num conjunto A é transitiva sse $\forall a, b, c \in A. R(a,b) \wedge R(b,c) \Rightarrow R(a,c)$.

4. (1 valor) Defina uma função `tabuada` que leia do teclado um inteiro `n` e imprima no écran os primeiros dez múltiplos (positivos) de `n` (um por linha).
Nota: Pode utilizar a função `sequence :: [IO a] -> IO ()` *para executar uma lista de ações.*

5. (2 valores) Escreva uma definição em Haskell que produza a seguinte lista infinita de inteiros:

`inff = [0,1,3,6,10,15,21,28,36,45,...]`.

6. (3 valores) Considere a seguinte declaração de tipo para árvores binárias:

```
data Arv a = Vazia | No a (Arv a) (Arv a)
```

(a) Defina uma função `listar`, que dada uma árvore, devolva a lista dos elementos na árvore.

(b) Defina uma função `simetrica`, que dada uma árvore, devolva uma nova árvore em que os elementos aparecem em posições simétricas (em relação à raiz) à sua posição inicial.

7. (2 valores) Responda **(apenas)** a uma das seguintes alíneas, usando indução matemática. *Nota: pode utilizar qualquer propriedade que tenha sido demostrada nas aulas, ou demonstrar qualquer resultado adicional que facilite a prova.*

(a) Considerando as funções definidas na questão anterior e as definições das funções `reverse` e `++` dadas nas aulas, mostre que para qualquer árvore `t`,

```
listar t = reverse (listar (simetrica t)).
```

(b) Considerando as definições das funções `++` e `foldr` dadas nas aulas mostre que, para qualquer operador binário associativo `op` tal que `e` é o elemento neutro de `op`, e quaisquer listas `xs` e `ys` então: `foldr op e (xs++ys) = op (foldr op e xs) (foldr op e ys)`.