Programação Funcional

Teste 418

14 de Abril de 2018

DCC/FCUP

Nome: _____

Nº mecanográfico: _____

Este teste contém 6 questões e 3 páginas.
Responda às questões no espaço marcado no enunciado, indicando sempre um tipo para a função definida. Se nada for dito explicitamente em contrário, pode usar funções auxiliares e/ou do prelúdio-padrão de Haskell.

1. (30%) Responda a cada uma das seguintes questões, indicando **apenas** o resultado de cada expressão.

(a) `[[1,2,3]] ++ [] ++ [[4],[5]]` = _____

(b) `length ([1]:[2]:[]:[3]:[4]:[])` = _____

(c) `take 7 [8,6..0]` = _____

(d) `fst ([(1,2),(3,4),(5,6),(9,7),(8,13)] !! 3)` = _____

(e) `[ (y,x) | x <- [1,2], y <- [x..4]]` = _____

(f) `[ 2^x | x <-[1..5], y <- [1..3], (x+y) 'mod' 3 == 0]` = _____

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

`[1,3,7,15,31,63,127,255,511,1023]` = _____

(h) Considere a seguinte definição em Haskell:

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

A avaliação da expressão `f [1..5]` tem como resultado: _____

(i) Indique um tipo admissível para `([False,True],['1','2'])`:

_____

(j) Indique o tipo mais geral da função: `p x y = (x,y)`:

_____

(k) Considere a seguinte função em Haskell:

```
h [] l = l
h [x] l = x:l
h (x:y:ys) l = if x == y then h ys (x:l) else h (y:ys) (x:l)
```

Indique um tipo admissível para a função `h`:

_____

(l) Indique o tipo mais geral da função `feql` definida como
`feql xs = head xs == head (reverse (tail xs))`:

_____

2. (20%) Representamos coordenadas de pontos no plano cartesiano como pares da forma (x, y).

a) Defina uma função `distancia` que dados dois pontos (x_1, y_1), (x_2, y_2), calcula a distância entre os dois pontos segundo a fórmula:

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

b) Três pontos $A = (x_1, y_1)$, $B = (x_2, y_2)$ e $C = (x_3, y_3)$ são colineares se o declive da recta AB[1] é igual ao declive da recta BC. Defina uma função `colineares`, que dados três pontos determina se são ou não colineares.

3. (20%) Considere a função `niguais` que, dado um inteiro `n` e um elemento `x`, devolve uma lista com `n` cópias de `x`. Por exemplo `niguais 5 'a' = ``aaaaa''` e `niguais 2 True = [True,True]`.

a) Defina `niguais` recursivamente.

b) Defina `niguais` usando listas em compreensão.

[1] O declive da recta formada pelos pontos (x_1, y_1), (x_2, y_2) é $d = \frac{y_1 - y_2}{x_1 - x_2}$.

4. (10%) Escreva uma definição recursiva da função `merge` para juntar duas listas ordenadas numa só mantendo a ordenação. Exemplo: `merge [3,5,7] [1,2,4,6] = [1,2,3,4,5,6,7]` e `merge ‘‘ceu’’ ‘‘belo’’ = ‘‘bceelou’’`.

5. (10%) Defina a função `length_zip` que dada uma lista `xs`, devolve uma lista de pares, em que a primeira componente é elemento que ocupa a mesma posição na lista original e o segundo é o comprimento da sublista que tem esse elemento à cabeça. Por exemplo: `length_zip ‘‘zip’’ = [(3, 'z'), (2, 'i'), (1, 'p')]`. **Sugestão:** utilize a função `zip`.

6. (10%) O *problema de decompor uma quantia em trocos* pode ser formalizado da seguinte maneira: dado um natural `n` e uma lista de naturais `xs`, encontrar decomposições de `n` como soma de valores em `xs` (eventualmente com repetições). Por exemplo, para `n=25` e `xs=[2,5,10]` uma decomposição possível é `[5,10,10]` (porque `25 = 5+10+10`). Escreva uma definição duma função `decompor` tal que `decompor n xs` encontra *todas* as alternativas para o problema dos trocos. O resultado deverá ser a lista vazia quando o problema não tem solução.