Programação Funcional

Exame 618

5 de Junho de 2018

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

(a) `"abc":[[ ]] ++ "dce":[ ]` = _____

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

(c) `[[1,2,3,4],[5,6,7,8],[9,10,11,12]] !! 2 !! 1` = _____

(d) `map (\x -> x*2) [1,2,3,4]` = _____

(e) `zipWith (-) [1,3..10] [0,3..]` = _____

(f) `dropWhile (<6) [2..10]` = _____

(g) `[x | x <- [1..10], x <= 6, x >= 4]` = _____

(h) Defina a seguinte lista em compreensão:
`[-2,4,-8,16,-32,64,-128,256,...]` = _____

(i) Considere a seguinte definição em Haskell:

```
h = let f [x] = 1
        f (y:z) = y * f z
    in f [1,2,3,4,5]
```

A avaliação da expressão `h` tem como resultado: _____

(j) Indique um tipo admissível para a função `f` definida como `f xs = head (tail xs)`:

_____

(k) Indique o tipo mais geral de `([False,True], ['0','1'])`: _____

(l) Considere as seguintes definições em Haskell:

```
data E a = V a | Op (E a) (E a)
aval (V v) f g = f v
aval (Op e1 e2) f g = g (aval e1 f g) (aval e2 f g)
```

Indique um tipo admissível para a função `aval`, se o tipo da função `f` for `a -> Bool`:

_____

(m) Indique o tipo mais geral de `foldr (++) []`:

_____

2. (15%) Na cadeira de Programação Ficcional, o regente acha por bem que apenas alunos com (>= 15) valores devem ser aprovados. *Nota: pode utilizar funções do prelúdio-padrão e/ou listas em compreensão mas não deve usar directamente* ***recursão****.*

(a) Defina uma função `aprov`, que dada uma lista de notas, devolva a lista de quem foi aprovado (`'A'`) ou reprovado (`'R'`). Por exemplo, `avalia [2,8,15,9] = ['R','R','A','R']` e `avalia [12,10,17,8,19] = ['R','R','A','R','A']`.

(b) Defina uma função `injust` que determina quantos alunos foram injustiçados pela exigência na avaliação. Ou seja, os alunos que deveriam ter passado, mas não conseguiram. Por exemplo, `injust [2,8,15,9] = 0` e `injust [12,10,17,8,19] = 2`.

3. (10%) Defina recursivamente uma função `repete` que recebe um elemento `a` e produz a seguinte lista infinita: `[[],[a],[a,a],[a,a,a,],...]`.

4. (5%) Escreva uma função `maximo` que lê da entrada padrão uma sequência de naturais terminada em 0 (um por linha) e escreve o valor máximo. **Sugestão:** pode usar a função `maximum` que devolve o valor máximo numa lista.

5. (15%) Considere uma função `compL` que implementa composição de uma lista de funções. Por exemplo, a chamada `compL [f,g,h] v` retorna o valor `f (g (h v))`.

(a) Defina a função `compL` recursivamente.

(b) Defina a função `compL` usando ordem superior.

6. (15%) Considere a seguinte declaração de tipo para árvores binárias:

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

(a) Defina uma função `soma` que dada uma árvore, calcule a soma dos valores nos nós da árvore.

(b) Recorde a função `foldr` definida para listas. Defina uma função `foldtree`, que se comporte como a função `foldr`, mas opere sobre árvores. **Nota**: A função `foldtree` deverá ter como parâmetro uma função do tipo `a -> b -> b -> b`. Por exemplo, podemos definir a função `soma t` da alínea anterior, como `foldtree soma3 0 t`, onde `soma3 x y z = x + y + z`.

7. (10%) 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, mostre que para qualquer árvore númerica `t`, `soma t = foldtree soma3 0 t`, onde `soma3 x y z = x + y + z`.

(b) Considerando as definições das funções `++` e `foldr` dadas nas aulas, mostre que para quaisquer `f`, `v` e `xs`: `foldr f v (xs++ys) = foldr f (foldr f v ys) xs`.