Exame de Programação Funcional

DCC/FCUP

16 de Junho de 2014

Versão A

Nome: _____

Nº mecanográfico: _____

- **Este exame contém 5 questões em 4 páginas.**
- **Responda às questões 1 e 2 no espaço marcado no enunciado.**
- **Responda às questões 3, 4 e 5 numa folha de exame separada.**
- **Duração: 2h + 30m tolerância.**

1. (20%) Responda a cada uma das seguintes questões, indicando **apenas** o resultado de cada expressão.

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

(b) `reverse (take 4 [1,3..]) =` _____

(c) `[x | x<-[1..10], x*x<10] =` _____

(d) `length [x+y | x<-[1,2], y<-[3,4]] =` _____

(e) `filter (>'b') "abacadabra" =` _____

(f) `zip [2..] "abc" =` _____

(g) `foldr (\x y->2*x+y) 0 [1,2,3] =` _____

(h) Indique um tipo admissível para `[(1,1), (2,2), (3,3)]`

_____

(i) Indique um tipo admissível para `[reverse, tail, take 5]`

_____

(j) Considere a seguinte definição duma função `f`:

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

Indique o tipo mais geral desta função:

_____

2. (25%) A *linguagem dos Ps* é um jogo de palavras em que duplicamos cada vogal (letras 'a', 'e', 'i', 'o', 'u') e colocamos um 'p' entre elas; todos os outros carateres e letras ficam inalterados. Assim, por exemplo, a frase "*ola, mundo!*" fica "*opolapa, mupundopo!*".

Escreva uma definição da função `tranforma :: String -> String` que transforma uma frase para a linguagem dos Ps. Para simplificar, pode assumir que a frase não contém letras maísculas nem acentuadas.

Exemplos:

```
transforma "ola, mundo!"       = "opolapa, mupundopo!"
transforma "4 gatos e 3 ratos" = "4 gapatopos epe 3 rapatopos"
```

Responda às questões 3, 4 e 5 numa folha de exame separada.

3. (25%) Vamos representar as temperaturas médias de vários dias consecutivos por uma lista de valores *Float*.

Escreva uma definição da função `subidas :: [Float] -> Int` que calcula *quantas vezes a temperatura subiu* (isto é, a temperatura do dia anterior foi estritamente inferior à do dia atual). No caso de a lista de temperaturas ter menos de dois valores, o resultado deverá ser zero.

Exemplos:

```
subidas [19,20,21,22]           = 3
subidas [19,20,20,22]           = 2
subidas [20,19,18,19]           = 1
subidas [20,21,20,19,19,22,23]  = 3
subidas [20,20]                 = 0
subidas [20]                    = 0
subidas []                      = 0
```

4. (15%) Considere a seguinte definição de um tipo de dados para *árvores binárias* em que `N` é o construtor de nós e `F` é o construtor de folhas.

```
data Arv a = F | N a (Arv a) (Arv a)
```

(a) Escreva uma definição recursiva da função `alturas :: Arv a -> Arv Int` que transforma uma árvore binária noutra com a mesma estrutura mas em que o valor de cada nó é dado pela sua *altura* (isto é, o maior comprimento dum caminho desse nó até uma folha).

Exemplo (com uma árvore de cadeias de carateres):

```
   alturas (N "joão" (N "abel" F F) (N "pedro" F F))
=  N 2 (N 1 F F) (N 1 F F)
```

Para obter cotação total deverá **evitar re-calcular a altura de cada nó múltiplas vezes**; deverá ainda estruturar o seu programa usando funções auxiliares.

(b) Usando a função anterior, defina uma outra função `equilibrada :: Arv a -> Bool` que exprima a condição duma árvore ser *equilibrada*, isto é, as alturas das sub-árvores de cada nó diferem no máximo de 1 unidade.

Exemplo (com a mesma árvore):

```
   equilibrada (N "joão" (N "abel" F F) (N "pedro" F F))
=  True
```

5. (15%) Considere as seguintes definições das funções `take` e `drop` do prelúdio-padrão:

```
take 0 xs           = []                  -- take.0
take n []     | n>0 = []                  -- take.1
take n (x:xs) | n>0 = x : take (n-1) xs   -- take.2

drop 0 xs           = xs                  -- drop.0
drop n []     | n>0 = []                  -- drop.1
drop n (x:xs) | n>0 = drop (n-1) xs       -- drop.2
```

Usando indução sobre n e análise de casos sobre a lista xs mostre que

$$\texttt{take}\ m\ (\texttt{drop}\ n\ xs) = \texttt{drop}\ n\ (\texttt{take}\ (m+n)\ xs)$$

para todos inteiros não-negativos m, n e listas xs.