Teste de Programação Funcional

DCC/FCUP

17 de Abril de 2013

Versão A

Nome: _____

Nº mecanográfico: _____

- **Este teste contém 3 questões e 2 páginas.**
- **Responda à questão 1 no espaço marcado no enunciado.**
- **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) `head ([]++[1,2]++[3]) =` _____

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

(c) `length ([1,2]:[3]:[]) =` _____

(d) `length [(x,y)| x<-"abc", y<-[0,1,2]] =` _____

(e) `sum (map (*2) [0,1,2]) =` _____

(f) `length (drop 2 "ABCDEF") =` _____

(g) Indique um tipo admissível para `([0,1,2], ["abc","def"])`

_____

(h) Indique o tipo mais geral da função `(+)`

_____

(i) Indique o tipo mais geral da função `map`

_____

(j) Indique o tipo mais geral da função `drop`

_____

Responda às questões 2 e 3 numa folha de exame.

2. (40%) Considere uma função `count :: (a -> Bool) -> [a] -> Int` que conta o número de elementos de uma lista para os quais o predicado dado é `True`. Exemplos:

```
> count (>2) [0,1,2,3]
1
> count (/='a') "banana"
3
```

(a) Escreva uma definição recursiva da função `count`.

(b) Escreva uma definição não-recursiva de `count` usando `filter` e `length`.

(c) Usando `count` e a função `isLetter :: Char -> Bool` escreva uma definição da função `extras :: String -> Int` que conta o número de caracteres numa cadeia que *não* são letras.

3. (30%) Escreva uma definição recursiva da função

```
split :: Char -> String -> [String]
```

que separa uma cadeia em sub-cadeias delimitadas por um carater dado. Exemplos:

```
> split '@' "pbv@dcc.fc.up.pt"
["pbv","dcc.fc.up.pt"]
> split '/' "/usr/include/linux/"
["","usr","include","linux",""]
```

Sugestão: use as funções `takeWhile` e `dropWhile` do prelúdio-padrão.