Programação Funcional

Exame 618r

25 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 6, indique sempre o tipo da função definida.

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

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

(b) `tail(1:(5:(4:(3:[]))))` = _____

(c) `(sum . map length) [[1,2,3],[4,5,6],[]]` = _____

(d) `[(x,y )| x <- [0..4], y <- [x..4], odd (x+y)]` = _____

(e) `takeWhile (<100) (map (\x->x*x) [1,3..])` = _____

(f) `foldr (-) 10 [1,3..7]` = _____

(g) Defina a seguinte lista em compreensão:

`[(-2,1),(4,-3),(-6,7),(8,-15),(-10,31),...]` = _____

(h) Considere a seguinte definição em Haskell:

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

A avaliação da expressão `f [7,8,3,4]` tem como resultado: _____

(i) Indique um tipo admissível para a função `f` definida como `(f xs = reverse xs == xs)`:

_____

(j) Indique o tipo mais geral de `[(3==),(<1)]`: _____

(k) Considere as seguintes definições:

```
data Arv = ???
f:: Arv -> [Int]
f Folha = []
f (No l v r) = f l ++ [v] ++ f r
```

Complete a definição do tipo `Arv`, para que a função `f` esteja bem definida:

_____

(l) Considerando que `until :: (a -> Bool) -> (a -> a) -> a -> a`, qual o tipo principal de `map (until (>=100) (3*))`?

_____

2. (15%) Considere uma base de dados de estatísticas de futebol representada pelo tipo `[(String,Int,Int,Int)]` e que armazena, para cada equipa, o seu nome, número de vitórias, empates e derrotas. Responda às seguintes questões utilizando funções de ordem superior (isto é, sem usar recursividade explícita nem listas por compreensão).

(a) Escreva uma função `pontuação` que dada uma base de dados no formato acima, devolve uma lista das equipas e respectivos pontos acumulados (recorde que uma vitória corresponde a 3 pontos, um empate a 1 ponto e uma derrota a 0 pontos).

(b) Escreva a função `njogos` que dado um inteiro positivo `n` e uma base de dados, verifica que todas as equipas efectuaram o mesmo número `n` de jogos.

3. (5%) Escreva uma função `crescente` que lê números do teclado, um por linha e enquanto esses números formarem uma sequência crescente. No final deverá escrever no monitor o comprimento da sequência crescente que foi lida.

4. (10%) Defina recursivamente uma função `maisvezes` que produza a seguinte lista infinita: `[1,3,6,8,16,18,36,38,76,...]`.

5. (15%) Representamos as arestas de um grafo dirigido como uma lista de pares da forma `(a,b)`.

(a) Um caminho pode ser definido como uma sequência de arestas $(v_1, v_2), (v_2, v_3), \ldots, (v_{n-1}, v_n)$. Defina recursivamente uma função `caminho`, que dada uma lista `as` de arestas, determina se esta representa um caminho válido.

(b) Usando ordem-superior defina uma função `caminhoG`, que dado um caminho `ps` e uma lista `as` de arestas representando um grafo dirigido, verifica se `ps` é um caminho no grafo `as`.

6. (15%) Considere a seguinte definição de tipo de dados `ArvC a` para árvores binárias:

```
data ArvC a = Vazia | No a Int (ArvC a) (ArvC a)
```

(a) Defina a função `nelem`, que calcula o número total de nós internos de uma dada árvore do tipo `ArvC a`.

(b) Defina uma função `update`, que dada uma árvore qualquer do tipo `ArvC a`, retorna uma árvore do mesmo tipo onde o valor adicional do tipo inteiro indica o número total de nós da subárvore com raiz nesse nó. A função não deve percorrer cada sub-árvore mais do que uma vez, nem usar a função `nelem`. *Sugestão: defina a função* `eElem` *que, dada uma árvore com esta propriedade, calcula o número de nós internos, sem percorrer recursivamente a árvore.*

7. (10%) Responda **(apenas)** a uma das seguintes alíneas, usando indução matemática.

(a) Considerando as funções definidas na questão anterior, mostre que, para qualquer árvore `t` do tipo `ArvC a`, `nelem t = nelem (update t)`.

(b) Considerando as definições das funções `map` e `foldr` dadas nas aulas, mostre que para quaisquer `f` e `xs`: `map f xs = foldr (\x xs -> f x:xs) []`.