Exame de Programação Funcional

DCC/FCUP

15 de Junho de 2021

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.**

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

(a) `tail (reverse [1..5])` = _____

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

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

(d) `head (reverse (zip [1,2,3] "abc"))` = _____

(e) `foldr (-) 0 [1,2,3]` = _____

(f) Indique um tipo admissível para `[(1,[2]),(3,[4,5])]`:

_____

(g) Indique um tipo admissível para `filter (/=0)`:

_____

(h) Considere a seguinte definição da função `takeWhile` do prelúdio-padrão

```
takeWhile p []     = []
takeWhile p (x:xs) | p x       = x : takeWhile p xs
                   | otherwise = takeWhile p xs
```

Indique o tipo mais geral desta definição:

_____

2. (20%)

(a) Escreva uma definição da função *primo* :: *Integer* → *Bool* que testa se um número é primo procurando sucessivamente divisores.
Para obter a cotação total deverá implementar uma versão eficiente que teste candidatos a divisor apenas até à raiz quadrada do número dado.

(b) Um par de primos gémeos é da forma $(p, p+2)$ tal que p e $p+2$ são ambos primos. Usando a função *primo* anterior, escreva uma definição da função *gemeos* :: *Integer* → $(Integer, Integer)$ tal que *gemeos* n determina o primeiro par de primos gémeos maiores ou iguais do que n.
Sugestão: use uma lista em compreensão.

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

3. (20%) Vamos representar figuras no plano por listas de pontos (pares de coordenadas).

```
type Point = (Double,Double) -- coordenadas x, y
```

Pretende-se determinar o menor rectângulo ortogonal aos eixos e que envolve toda uma lista de pontos (ver a Figura 1).

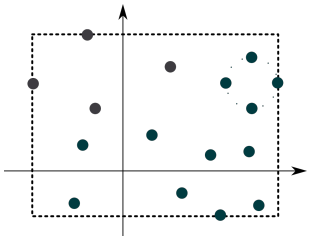

Figura 1: Menor rectângulo envolvente duma lista de pontos

Defina uma função

```
boundingBox :: [Point] -> (Point, Point)
```

que determina os dois cantos inferior esquerdo e superior direito do rectângulo envolvente de uma lista de pontos. Sugestão: pode exprimir as coordenadas dos cantos do rectângulo usando as funções *minimum*, *maximum* :: $Ord\ a \Rightarrow [a] \rightarrow a$ do preludio-padrão.

4. (20%) Considere a definição em Haskell dum tipo de dados para conjuntos finitos representados como árvores binárias de pesquisa (isto é, cada nó divide o conjunto nos sub-conjuntos de valores menores e maiores do que o valor do nó).

```
data Set a = Empty | Node a (Set a) (Set a)
```

(a) Escreva uma definição recursiva da função

```
insert :: Ord a => a -> Set a -> Set a
```

que insere um elemento num conjunto mantendo a propriedade das árvores de pesquisa. Tenha o cuidado de garantir que não insere no conjunto valores repetidos.

(b) Escreva uma definição recursiva da função

```
exists :: (a -> Bool) -> Set a -> Bool
```

que verifica se uma função boleana é verdadeira para algum elemento de um conjunto.

5. (20%) Considere as seguintes definições das funções *take* e *drop* do prelúdio-padrão.

$$
\begin{aligned}
&\mathit{take}\ 0\ xs = [] && \text{(take.1)}\\
&\mathit{take}\ n\ [\,] \mid n > 0 = [\,] && \text{(take.2)}\\
&\mathit{take}\ n\ (x : xs) \mid n > 0 = x : \mathit{take}\ (n-1)\ xs && \text{(take.3)}\\
&\mathit{drop}\ 0\ xs = xs && \text{(drop.1)}\\
&\mathit{drop}\ n\ [\,] \mid n > 0 = [\,] && \text{(drop.2)}\\
&\mathit{drop}\ n\ (x : xs) \mid n > 0 = \mathit{drop}\ (n-1)\ xs && \text{(drop.3)}
\end{aligned}
$$

Usando indução sobre n, mostre que

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

para todo $m, n \geq 0$ e listas xs.