

Nome: _____

Nº mecanográfico: _____

- Este exame contém 7 questões em 4 páginas e tem uma duração de 2h(+30m).
- 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) `[]:[]:[] ++ []:[]:[]` = _____

(b) `length ([]:[]:[] ++ []:[]:[])` = _____

(c) `map ('div' 4) [1..10]` = _____

(d) `takeWhile (<100) (iterate (3*) 1)` = _____

(e) `(filter even.(!!2)) [[3,5,8],[4],[7,4,1,8],[9,11]]` = _____

(f) `take 10 [2*x*y | x<-[1..], y <-[1..]]` = _____

(g) Defina a lista infinita `[0,1,4,7,8,31,12,127,16,511,...]` em compreensão:

(h) Considere a seguinte definição em Haskell:

```
f p [] = True
f p (x:xs) = p x && f p xs
```

A avaliação da expressão `f (>3) [3,4,5]` tem como resultado: _____

(i) Indique um tipo admissível para `[even,(==2),(>1)]`:

(j) Indique o tipo mais geral de `length.(filter (>'a'))`:

(k) Considere as seguintes definições em Haskell:

```
data ???

f :: Arv -> Int
f (No esq a dir) | dir == Vazia = a
                  | otherwise = f dir
```

O que falta em ??? para que a função `f` esteja bem definida?

(l) Indique o tipo mais geral de `flip f x y = f y x`:

2. (15%) Considere a função `imparDiv3`, que dada uma lista de inteiros, verifica que cada inteiro na lista que é divisível por 3 é ímpar. Por exemplo `imparDiv3 [1,15,153,83,64,9] = True` e `imparDiv3 [1,12,153,83,9] = False`.

- (a) Defina a função `imparDiv3` usando listas em compreensão e funções do prelúdio, mas não **recursão**.
- (b) Defina a função `imparDiv3` usando funções de ordem-superior (`map`, `filter`, `foldr`), mas não **recursão** ou **listas em compreensão**.

3. (10%) Considere o seguinte tipo `Rel a`, definido como `type Rel a = [(a,[a])]`, para representar relações binárias, como uma lista de adjacências (*Nota: a ordem dos elementos na lista de nós adjacentes não é importante, mas não devem aparecer elementos repetidos na lista de nós adjacentes, nem devem existir dois pares na relação com o mesmo primeiro elemento*). Por exemplo a relação $R = \{(1, 1), (1, 3), (2, 2), (3, 2), (3, 3)\}$ é representada por `r = [(1,[1,3]), (2,[2]), (3,[2,3])]`. Defina uma função `composta`, que dadas duas relações R_1 e R_2 , representadas da forma descrita acima, calcule a relação composta das duas. (*Nota: $R_1 \cdot R_2 = \{(x, z) \mid (x, y) \in R_1 \wedge (y, z) \in R_2\}$* .) Por exemplo `composta r r = [(1,[1,3,2]), (2,[2]), (3,[2,3])]`. **Sugestão:** poderá utilizar a função `nub` do módulo `Data.List` para remover elementos repetidos.

4. (5%) Considere a seguinte série definida por recorrência da seguinte forma: $a_1 = 1$, $a_n = 2 * a_{n-1} + n + 1$, $n > 1$. Defina em Haskell a lista infinita `[1,5,14,33,72,151,310,629,...]`, contendo todos os termos da série. *Nota: serão valorizadas soluções mais eficientes.*

5. (15%) Considere listas em que cada valor aparece sequencialmente duplicado. Por exemplo `[1,1,2,2,3,3]` ou `[True,True,False,False]`.

- (a) Defina recursivamente a função `duplicada`, que dada uma lista `xs`, verifica que a lista satisfaz a condição acima. Por exemplo `duplicada [1,1,2,2,3,3] = True` e `duplicada [True,False,False] = False`.
- (b) Usando ordem superior e/ou listas em compreensão, defina a função `duplica`, que dada uma lista `xs`, constrói uma nova lista de valores duplicados. Por exemplo, `duplica [1,2,3,4] = [1,1,2,2,3,3,4,4]` e `duplica [True,False] = [True,True,False,False]`.

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

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

- (a) Defina a função `emOrdem`, que dada uma árvore do tipo `Arv a`, retorna a lista de elementos na árvore, seguindo a ordem da esquerda para a direita.
- (b) Recorde a função de ordem-superior `any :: (a->Bool) -> [a] -> Bool` definida para listas. Defina uma função `anyArv`, que se comporte como a função `any`, mas opere sobre árvores do tipo `Arv a`.

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

Nota: pode utilizar qualquer propriedade que tenha sido demonstrada nas aulas. Pode ainda usar sem demonstrar as seguintes propriedades sobre listas: $\text{any } (xs ++ ys) = \text{any } xs \mid \mid \text{any } ys$ e $\text{foldr } f \ v \ (xs ++ ys) = \text{foldr } f \ v \ (\text{foldr } f \ v \ ys) \ xs$.

- (a) Considerando as funções `emOrdem` e `anyArv` definidas na questão anterior e a função `any` do prelúdio, mostre que para qualquer árvore `t` e predicado `p`, `anyArv p t = any p (emOrdem t)`.
- (b) Considerando as definições das funções `foldr` e `concat` dadas nas aulas, assim como a função `flip` do prelúdio, definida como `flip f x y = f y x`, mostre que para quaisquer `f` e `v` e `xs`: `foldr f v (concat xs) = foldr (flip (foldr f)) v xs`.