

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 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) `"abc":[[]] ++ "dce":[]` = `["abc", [], "dce"]`
- (b) `tail ([1]:[2]:[]:[3]:[4]:[])` = `[[2], [], [3], [4]]`
- (c) `[[1,2,3,4],[5,6,7,8],[9,10,11,12]] !! 2 !! 1` = 10
- (d) `map (\x -> x*2) [1,2,3,4]` = `[2,4,6,8]`
- (e) `zipWith (-) [1,3..10] [0,3..]` = `[1,0,-1,-2,-3]`
- (f) `dropWhile (<6) [2..10]` = `[6,7,8,9,10]`
- (g) `[x | x <- [1..10], x <= 6, x >= 4]` = `[4,5,6]`
- (h) Defina a seguinte lista em compreensão:
`[-2,4,-8,16,-32,64,-128,256,...]` = `[(-1)^x * 2^x | x <- [1..]]`
- (i) Considere a seguinte definição em Haskell:
`h = let f [x] = 1`
`f (y:z) = y * f z`
`in f [1,2,3,4,5]`

A avaliação da expressão `h` tem como resultado: 24

- (j) Indique um tipo admissível para a função `f` definida como `f xs = head (tail xs)`:

`[a] -> a`

- (k) Indique o tipo mais geral de `([False,True], ['0','1'])`: `([Bool],String)`

- (l) Considere as seguintes definições em Haskell:

```
data E a = V a | Op (E a) (E a)
aval (V v) f g = f v
aval (Op e1 e2) f g = g (aval e1 f g) (aval e2 f g)
```

Indique um tipo admissível para a função `aval`, se o tipo da função `f` for `f :: a -> Bool`:

`aval :: E a -> (a -> Bool) -> (Bool -> Bool -> Bool) -> Bool`

- (m) Indique o tipo mais geral de `foldr (++) []`: `[[a]] -> [a]`

2. (15%) Na cadeira de Programação Ficcional, o regente acha por bem que apenas alunos com (≥ 15) valores devem ser aprovados. *Nota: pode utilizar funções do prelúdio-padrão e/ou listas em compreensão mas não deve usar directamente **recursão**.*

- (a) Defina uma função `aprova`, que dada uma lista de notas, devolva a lista de quem foi aprovado ('A') ou reprovado ('R'). Por exemplo, `avalua [2,8,15,9] = ['R','R','A','R']` e `avalua [12,10,17,8,19] = ['R','R','A','R','A']`.
- (b) Defina uma função `injust` que determina quantos alunos foram injustiçados pela exigência na avaliação. Ou seja, os alunos que deveriam ter passado, mas não conseguiram. Por exemplo, `injust [2,8,15,9] = 0` e `injust [12,10,17,8,19] = 2`.

```
avalua :: [Int] -> String
avalua xs = [ if x >= 15 then 'A' else 'R' | x <- xs ]

injust :: [Int] -> Int
injust xs = length (filter (\x -> x >= 10 && x < 15) xs)
```

3. (10%) Defina recursivamente uma função `repete` que recebe um elemento `a` e produz a seguinte lista infinita: `[[], [a], [a,a], [a,a,a], ...]`.

```
repete :: a -> [[a]]
repete x = [] : map (x:) (repete x)

ou

repete x = [] : [x:1 | 1 <- repete x]
```

4. (5%) Escreva uma função `maximo` que lê da entrada padrão uma sequência de naturais terminada em 0 (um por linha) e escreve o valor máximo. **Sugestão:** pode usar a função `maximum` que devolve o valor máximo numa lista.

```
maximo :: IO ()
maximo = maximo_aux 0
maximo_aux :: Int -> IO ()
maximo_aux m = do x <- getLine
                  let n = read x in
                  if n == 0 then print m
                  else calc (max n m)

ou

maximo = do xs <- maximo_aux
           putStrLn (show (maximum xs))
maximo_aux :: IO [Int]
maximo_aux = do x <- getLine
               let n = read x in
               if n == 0 then
                 return []
               else do xs <- maximo_aux
                      return (n:xs)
```

5. (15%) Considere uma função `compL` que implementa composição de uma lista de funções. Por exemplo, a chamada `compL [f,g,h] v` retorna o valor `f (g (h v))`.

- (a) Defina a função `compL` recursivamente.
- (b) Defina a função `compL` usando ordem superior.

```
compL :: [a -> a] -> a -> a
```

recursivamente:

```
compL [] v = v
compL (f:fs) v = f (compL fs v)
```

usando ordem superior:

```
compL fs v = foldr f v fs
              where f g x = g x
```

ou

```
compL fs v = (foldr (.) id fs) v
```

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

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

- (a) Defina uma função `soma` que dada uma árvore, calcule a soma dos valores nos nós da árvore.
- (b) Recorde a função `foldr` definida para listas. Defina uma função `foldtree`, que se comporte como a função `foldr`, mas opere sobre árvores. **Nota:** A função `foldtree` deverá ter como parâmetro uma função do tipo `a -> b -> b -> b`. Por exemplo, podemos definir a função `soma t` da alínea anterior, como `foldtree soma3 0 t`, onde `soma3 x y z = x + y + z`.

```
soma :: Num a => Arv a -> a
soma Vazia = 0
soma (No x esq dir) = x + (soma esq) (soma dir)
```

```
foldtree :: (a -> b -> b -> b) -> b -> Arv a -> b
foldtree f v Vazia = v
foldtree f v (No x esq dir) = f x (foldtree f v esq) (foldtree f v dir)
```

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

Nota: pode utilizar qualquer propriedade que tenha sido demonstrada nas aulas, ou demonstrar qualquer resultado adicional que facilite a prova.

- (a) Considerando as funções definidas na questão anterior, mostre que para qualquer árvore numérica t , $\text{soma } t = \text{foldtree soma3 } 0 \ t$, onde $\text{soma3 } x \ y \ z = x + y + z$.
- (b) Considerando as definições das funções $++$ e foldr dadas nas aulas, mostre que para quaisquer f, v e xs : $\text{foldr } f \ v \ (xs++ys) = \text{foldr } f \ (\text{foldr } f \ v \ ys) \ xs$.

(a) Por indução sobre t .

- Caso base: $t = \text{Vazia}$

$\text{foldtree soma3 } 0 \ t = t = \text{soma Vazia}$

- Caso indutivo: $t = \text{No } x \ \text{esq } \text{dir}$

Hipótese de indução: $\text{soma esq} = \text{foldtree soma3 } 0 \ \text{esq}$
 $\text{soma dir} = \text{foldtree soma3 } 0 \ \text{dir}$

Tese: $\text{soma } t = \text{foldtree soma3 } 0 \ t$

$\text{foldtree soma3 } 0 \ t$
 $= \text{foldtree soma3 } 0 \ (\text{No } x \ \text{esq } \text{dir})$
 $= \text{soma3 } x \ (\text{foldtree soma3 } 0 \ \text{esq}) \ (\text{foldtree soma3 } 0 \ \text{dir})$
 $= \text{soma3 } x \ (\text{soma esq}) \ (\text{soma dir})$
 $= x + (\text{soma esq}) + (\text{soma dir})$
 $= \text{soma } (\text{No } x \ \text{esq } \text{dir}) = \text{soma } t \quad (\text{c.q.d.})$

(b) Por indução sobre xs .

- Caso base: $xs = []$

$\text{foldr } f \ v \ ([]++ys) = \text{foldr } f \ v \ ys = \text{foldr } f \ (\text{foldr } f \ v \ ys) \ []$

- Caso indutivo: $xs = (z:zs)$

Hipótese de indução: $\text{foldr } f \ v \ (zs++ys) = \text{foldr } f \ (\text{foldr } f \ v \ ys) \ zs$

Tese: $\text{foldr } f \ v \ ((z:zs)++ys) = \text{foldr } f \ (\text{foldr } f \ v \ ys) \ (z:zs)$

$\text{foldr } f \ v \ ((z:zs)++ys)$
 $= \text{foldr } f \ v \ (z:(zs++ys))$
 $= f \ z \ (\text{foldr } f \ v \ (zs++ys))$
 $= f \ z \ (\text{foldr } f \ (\text{foldr } f \ v \ ys) \ zs)$
 $= f \ z \ (\text{foldr } f \ (\text{foldr } f \ v \ ys) \ zs)$
 $= \text{foldr } f \ (\text{foldr } f \ v \ ys) \ (z:zs) \quad (\text{c.q.d.})$