

Nome: _____

Nº mecanográfico: _____

Este teste contém 6 questões e 3 páginas.

Responda às questões no espaço marcado no enunciado, indicando sempre um tipo para a função definida. Se nada for dito explicitamente em contrário, 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) `[[1,2]]++ [[]]++ [[3,4],[5]]` = _____

(b) `([1,2]:[]:[3,4]:[[5]]) !! 3` = _____

(c) `length ([]:[]:[])` = _____

(d) `drop 4 [0,4..32]` = _____

(e) `[(x+y,x*y) | x<-[1..4], y<-[x+1..4]]` = _____

(f) `[y | y<-ys, y `mod` 2 == 0] | ys <- [[3,5,2,8],[4,6,7,1,3],[9,5,11]]]` = _____

(g) Sem usar explicitamente a lista dada, defina a seguinte lista em compreensão.

`[(0,6),(1,5),(2,4),(3,3),(4,2),(5,1),(6,0)]` = _____

(h) Considere a seguinte definição em Haskell:

`h [] = 1`

`h [x] = x`

`h (x:y:xs) = x*y + h (y:xs)`

A avaliação da expressão `h [1,3,1,5,0,4]` tem como resultado: _____

(i) Indique um tipo admissível para `[('1',"a"),('2',"b")]` :

(j) Indique o tipo mais geral da função: `f x xs = sum xs < x`:

(k) Indique um tipo admissível para a função `ig` definida da seguinte forma:

`ig [] = True`

`ig [x] = True`

`ig (x1:x2:xs) = x1 == x2 && ig (x2:xs)`

(l) Indique o tipo mais geral da função `fix` definida como `fix f x = f x == x`:

2. (20%) Três números inteiros positivos a , b , e c , formam um terno pitagórico se $a^2 + b^2 = c^2$. Nesse caso, os valores a e b correspondem às medidas dos catetos e c à medida da hipotenusa do triângulo retângulo formado por a , b , e c .

- a) Defina uma função `pitagoricos`, que dados três inteiros (em qualquer ordem), determina se formam ou não um terno pitagórico.
- b) Defina uma função `hipotenusa`, que dada a medida dos catetos a e b (representados por valores do tipo `Float`), determina a medida da hipotenusa do triângulo retângulo de catetos a e b .

3. (20%) Defina a função `diferentes` que retorna todos os valores de uma lista que são diferentes do valor seguinte. Por exemplo `diferentes [1,2,2,1,1,3,3,3]` e `diferentes [1,2,1,3]` ambos retornam `[1,2,1]`, e `diferentes "aaa"` retorna `[]`.

- a) Defina `diferentes` recursivamente.
- b) Defina `diferentes` usando listas em compreensão. **Sugestão:** utilize a função `zip`.

4. (10%) Considere a função `zip3`, com a mesma funcionalidade da função `zip`, mas recebendo como argumento três listas de tamanhos possivelmente diferentes. Por exemplo, `zip3 [3,5,7] [1,2,4,6,9] "belo" = [(3,1,'b'),(5,2,'e'),(7,4,'l')]`. Defina a função `zip3` usando listas em compreensão e a função `zip`.

5. (10%) Defina recursivamente uma função `partir` que dado um valor `x` e uma lista `xs`, retorne duas listas contendo os elementos que aparecem antes da primeira ocorrência de `x` e os restantes elementos, respectivamente. Por exemplo `partir 5 [7,2,7,5,3,5,6,8]` retorna `([7,2,7], [5,3,5,6,8])` e `partir 'd' "banana"` retorna `("banana", "")`.

6. (10%) Consideremos uma partição de uma lista `xs` como uma lista de sublistas não vazias `[xs1, ..., xsn]` tais que `xs == xs1 ++ ... ++ xsn`. Por exemplo, as listas `[[1],[2],[3]]` e `[[1],[2,3]]` são ambas partições da lista `[1,2,3]`. Defina uma função `parts` que dada uma lista `xs` devolve a lista de todas as partições de `xs`. Por exemplo, `parts "abc"` devolve `[["a", "b", "c"], ["a", "bc"], ["ab", "c"], ["abc"]]`.
Nota: A ordem das permutações no resultado não é importante.