Início	quinta, 27 de outubro de 2022 às 17:14
Estado	Prova submetida
Data de submissão:	quinta, 27 de outubro de 2022 às 17:33
Tempo gasto	19 minutos 5 segundos
Nota	**4,75** de um máximo de 6,00 (**79**%)

Pergunta 1

Correta Pontuou 0,500 de 0,500

What is the type of the following function?

```
orderedPair (a, b)
  | a <= b = (a, b)
  | otherwise = (b, a)
```

a. `(Num a, Num b) => (a, b) -> (b, a)`

b. `(Ord a) => (a, a) -> (a, a)` ✓

c. `(Num a, Num b) => (a, b) -> (a, b)`

d. `(Ord a, Ord b) => (a, b) -> (b, a)`

e. `(Num a, Ord a, Num b, Ord b) => (a, b) -> (b, a)`

Pergunta 2

Correta Pontuou 0,500 de 0,500

What is the result of the following expression?

```
(length . (filter (> 0))) [1, 2, -3, 4, -5]
```

a. `0`

b. The evaluation of the expression produces an error.

c. `3` ✓

d. `1`

e. `2`

Pergunta 3

Correta Pontuou 0,500 de 0,500

What is the type of the following expression?

```
[(++) [], map (+1)]
```

a. `[[a] -> [a]]`

b. `(Num a) => [a]`

c. `(Num a, Num b) => [[a] -> [b]]`

d. `[[a] -> [b]]`

e. `(Num a) => [[a] -> [a]]` ✓

Pergunta 4

Correta Pontuou 0,500 de 0,500

What is the type of the following function?

```
fun (x, y, _) = (y, x, y)
fun (_, y, x) = (y, x, y)
```

a. `(a, a, a) -> (a, a, a)`

b. `(a, b, c) -> (a, b, c)`

c. `(a, b, a) -> (b, a, b)` ✓

d. `(a, b, c) -> (d, e, f)`

e. `(a, a, a) -> (b, b, b)`

Pergunta 5

Correta Pontuou 0,500 de 0,500

What is the result of the following expression?

```
[(a, b) | a <- "abc", b <- [1, 2], a <= 'd']
```

a. `[('a',1), ('b',1), ('c',1), ('a',2), ('b',2), ('c',2)]`

b. `[('a',1), ('a',2), ('b',1), ('b',2), ('c',1), ('c',2)]` ✓

c. The evaluation of the expression produces an error.

d. `[('a',1), ('b',1), ('c',1)]`

e. `[]`

Pergunta 6

Correta Pontuou 0,500 de 0,500

What is the result of the following expression?

```
foldl (/) 200 [1, 2, 4]
```

a. `50.0`

b. The evaluation of the expression produces an error.

c. `25.0` ✓

d. `100.0`

e. `1.0e-2`

Pergunta 7

Correta Pontuou 0,500 de 0,500

Which of the following Prelude functions does NOT necessarily return a list?

a. `(++)`

b. `(:)`

c. `zip`

d. `init`

e. `(!!)` ✓

Pergunta 8

Correta Pontuou 0,500 de 0,500

Consider the three following statements about the "type" and "data" keywords.

A - "type" does not allow the use of type variables, unlike "data".

B - "type" does not allow recursive type definitions.

C - It is possible to define an instance of Eq using "data".

Which statements are correct?

a. Only B and C. ✓

b. Only A and B.

c. Only A and C.

d. Only B.

e. A, B and C.

Pergunta 9

Incorreta Pontuou -0,125 de 0,500

Among the types Maybe, State and IO, which of them are monads?

a. Maybe, State and IO.

b. Only Maybe and IO. ✗

c. Only State and IO.

d. Only IO.

e. None of these types is a monad.

Pergunta 10

Correta Pontuou 0,500 de 0,500

What is the correct type of the following function?

```
howdy name = putStrLn ("howdy " ++ name ++ "!")
```

a. `String -> IO ()` ✓

b. `String -> String`

c. `IO ()`

d. `IO (String)`

e. `String -> IO (String)`

Pergunta 11

Incorreta Pontuou -0,125 de 0,500

Haskell has lazy evaluation, which allows for ...

a. the automatic inference of the functions' type. ✗

b. the optimization of the memory consumption of a program, in exchange for an increased execution time.

c. the definition of polymorphic functions.

d. certain computations with infinite data structures to be finite.

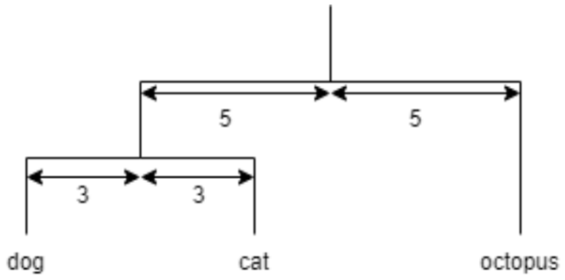
e. the definition of higher-order functions.

Pergunta 12

Correta Pontuou 0,500 de 0,500

Consider a dendrogram as a binary tree where each path leads to a string. Each non-leaf node of the dendrogram specifies the horizontal distance from the father node to each of the two child nodes. A father node is always at an equal horizontal distance from both its children.

Example of a dendrogram:



What is the most correct definition of the Dendrogram type?

a. `data Dendrogram = Leaf String | Node Int Int Dendrogram`

b. `(Integral a) => data Dendrogram = Leaf (String, a) | Node Dendrogram Dendrogram`

c. `data Dendrogram = Leaf (String, Int) | Node Dendrogram Dendrogram`

d. `(Integral a) => data Dendrogram = Leaf String | Node Dendrogram a a Dendrogram`

e. `data Dendrogram = Leaf String | Node Dendrogram Int Dendrogram` ✓