Início	quinta, 27 de outubro de 2022 às 17:44
Estado	Prova submetida
Data de submissão:	quinta, 27 de outubro de 2022 às 19:13
Tempo gasto	1 hora 29 minutos
Nota	**10,30** de um máximo de 14,00 (**74**%)

Informação

Information and Rules:

- In this part of the test, you can consult any materials available on the local computer, but not printed materials.
- **The presence of electronic and/or communication devices is forbidden.**
- Any attempt of fraud will be punished with the annulment of the test to all intervenients.
- The maximum duration of the test is as stated below.
- The score of each question is stated below, totaling 14 marks.
- You should answer all questions in the respective answer boxes.
- **You may use GHCI to test your code.**
- **Functions requested in previous questions can be used in the following ones, even if you have not implemented them. Do not copy the code from earlier questions to the following ones!**
- **If you implement additional auxiliary functions, you must include them in the answers to all questions where you use them.**
- **Do not copy code from the problem statements (e.g., the "type" and "data" declarations) to the answer boxes!**
- **If you want to write a comment, do it in Haskell: add "--" before the text or use "{-" and "-}."**
- **Document your code and use intuitive variable names to clarify the program's interpretation.**
- **You cannot use library functions or import modules under any circumstances.**

Informação

The Protecting Friends for Life (PFL) Zoo houses many species of animals, some of which were considered extinct in the past!

Consider the Species type synonym, which represents an animal species as a tuple composed of the species name followed by its population in the zoo.

```
type Species = (String, Int)
```

The Zoo type synonym contains a list of species:

```
type Zoo = [Species]
```

Pergunta 1

Respondida Pontuou 0,750 de 0,750

Implement *isEndangered :: Species -> Bool*, which receives a species and determines if it is endangered. A species is considered endangered if there are 100 or less individuals in the zoo.

```
isEndangered :: Species -> Bool
isEndangered (n,x) | x <= 100 = True
                   | otherwise = False
```

Pergunta 2

Respondida Pontuou 0,750 de 0,750

Implement *updateSpecies :: Species -> Int -> Species*, which, given a Species and an amount of newborn babies, returns a new instance of Species with the updated population.

```
-- se nasceram os que estavam la continuam, se não for o caso é so substituir o x por a
updateSpecies :: Species -> Int -> Species
updateSpecies (n,x) a = (n,x+a)
```

Informação

Note: In the following three exercises, there are constraints over the programming techniques. Solutions that do not follow the constraints will receive at most 25% of the question score.

Pergunta 3

Respondida Pontuou 1,500 de 1,500

Implement *filterSpecies :: Zoo -> (Species -> Bool) -> Zoo*, which, given the list species of a zoo and a predicate (i.e. a function that performs a test on each species), returns the sublist of species that satisfy the predicate. The order of the species in the result must be the same as in the input.

Constraint: You must solve this exercise using recursion. List comprehensions and higher-order functions (such as map and filter) are prohibited.

```
filterSpecies :: Zoo -> (Species -> Bool) -> Zoo
filterSpecies [] func = []
filterSpecies (x:xs) func | func x == True =  x : filterSpecies xs func
                          | otherwise = filterSpecies xs func
```

Pergunta 4

Respondida Pontuou 1,500 de 1,500

Implement *countAnimals :: Zoo -> Int*, which, given the list of species of a zoo, counts the total population of the zoo.

Constraint: You must solve this exercise using higher-order functions. Recursion and list comprehensions are prohibited.

```
countAnimals :: Zoo -> Int
countAnimals xs = foldl (\x y -> x + (snd y)) 0 xs
```

Pergunta 5

Respondida Pontuou 1,500 de 1,500

Implement *substring :: (Integral a) => String -> a -> a -> String*, which returns the substring of a given string between an initial and final index (the character on the final index should also be included in the result; both indices are within bounds). Consider that the indices start at 0.

Constraint: You must solve this exercise using a list comprehension. Recursion and higher-order functions are prohibited.

Use case example:

```
ghci> substring "arctic fox" 0 5
"arctic"
```

```
substring :: (Integral a) => String -> a -> a -> String
substring xs s e = [x | (x,y) <- zip (xs) [0..t], fromIntegral(y) >= s && fromIntegral(y) <= e]
            where t = (length xs - 1)
```

Informação

Note: In the following exercises, you are free to use the programming techniques you prefer.

Pergunta 6

Respondida Pontuou 0,800 de 1,500

Implement *hasSubstr :: String -> String -> Bool*, which determines if the first string argument contains the second string argument (i.e. the second argument is a substring of the first one).

```
tmp :: String -> String -> Bool
tmp [] ys = False
tmp xs [] = True
tmp (x:xs) (y:ys) | x == y = tmp xs ys
            | otherwise = False
```

Pergunta 7

Respondida Pontuou 1,500 de 1,500

Implement *sortSpeciesWithSubstr :: Zoo -> String -> (Zoo, Zoo)*, which divides the species of the Zoo into a pair of sublists. The first sublist stores all the species whose name contains the string argument, while the second list has the remaining species. The order of the species in each list of the resulting pair must match the input list.

Use case example:

```
ghci> sortSpeciesWithSubstr [("arctic fox",30), ("polar bear",5), ("arctic wolf",12)] "arctic"
([("arctic fox",30),("arctic wolf",12)], [("polar bear",5)])
```

```
primeira :: Zoo -> String -> Zoo
primeira xs str = [x | x <- xs, hasSubstr (fst x) str]

segunda :: Zoo -> String -> Zoo
segunda xs str = [x | x <- xs, hasSubstr (fst x) str == False]
```

Informação

The rabbit population of the zoo increases every year. In year 0, there were 2 rabbits, while in year 1 there were 3 rabbits. In the following years, the number of rabbits corresponds to the sum of the rabbit population of the two previous years.

Pergunta 8

Respondida Pontuou 1,000 de 1,000

Implement *rabbits :: (Integral a) => [a]*, which returns an infinite list with the rabbit population of each year (starting at year 0).

Use case example:

```
ghci> rabbits
[2,3,5,8 …]
```

```
aux :: (Integral a) => a -> a
aux 0 = 2
aux 1 = 3
aux n = aux(n-1) + aux(n-2)
```

Pergunta 9

Respondida Pontuou 1,000 de 1,000

Implement *rabbitYears :: (Integral a) => a -> Int*, which returns the number of years needed for the rabbit population to be greater or equal to the input integral value.

Use case examples:

```
ghci> rabbitYears 2
0
ghci> rabbitYears 6
3
```

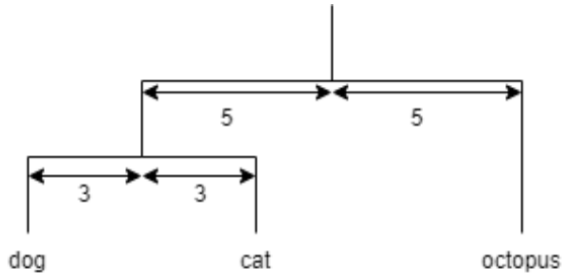
```
aux2 :: (Integral a) => [(a,a)]
aux2 = [(x,y) | (x,y) <- zip rabbits [0..]]

rabbitYears :: (Integral a) => a -> Int
rabbitYears n = head[fromIntegral(y) | (x,y) <- aux2, x >= n]
```

Informação

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.

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

For instance, consider the following figure and corresponding expression for a dendrogram.




```
myDendro :: Dendrogram
myDendro = Node (Node (Leaf "dog") 3 (Leaf "cat")) 5 (Leaf "octopus")
```

Pergunta 10

Não respondida Pontuação 1,500

Implement *dendroWidth :: Dendrogram -> Int*, which returns the width of a dendrogram (i.e., the horizontal distance between the leftmost and rightmost leaf nodes).

Use case example:

```
ghci> dendroWidth myDendro
13
```

Pergunta 11

Não respondida Pontuação 1,500

Implement *dendroInBounds :: Dendrogram -> Int -> [String]*, which returns the list of strings whose leaf nodes are up to a certain horizontal distance from the root of the dendrogram. Any order of the output list will be accepted.

Use case examples:

```
ghci> dendroInBounds myDendro 5
["cat","octopus"]
ghci> dendroInBounds myDendro 10
["dog","cat","octopus"]
```

◄ MT1 (27/10/2022) (Part 1)

Ir para...

MT1 - Grades ►

Início	quinta, 27 de outubro de 2022 às 17:17
Estado	Prova submetida
Data de submissão:	quinta, 27 de outubro de 2022 às 17:37
Tempo gasto	19 minutos 57 segundos
Nota	**4,88** de um máximo de 6,00 (**81**%)

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. `(Ord a) => (a, a) -> (a, a)` ✔

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

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

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

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. `3` ✔

b. The evaluation of the expression produces an error.

c. `2`

d. `0`

e. `1`

Pergunta 3

Correta Pontuou 0,500 de 0,500

What is the type of the following expression?

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

a. `(Num a) => [[a] -> [a]]` ✔

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

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

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

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

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, c) -> (d, e, f)`

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

e. `(a, b, a) -> (b, a, 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), ('a',2), ('b',1), ('b',2), ('c',1), ('c',2)]` ✔

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

c. `[]`

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

e. The evaluation of the expression produces an error.

Pergunta 6

Correta Pontuou 0,500 de 0,500

What is the result of the following expression?

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

a. `100.0`

b. `50.0`

c. `1.0e-2`

d. The evaluation of the expression produces an error.

e. `25.0` ✔

Pergunta 7

Correta Pontuou 0,500 de 0,500

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

a. `(++)`

b. `init`

c. `(!!)` ✔

d. `zip`

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 A and B.

b. A, B and C.

c. Only A and C.

d. Only B.

e. Only B and C. ✔

Pergunta 9

Não respondida Pontuação 0,500

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

a. Only IO.

b. Only Maybe and IO.

c. Maybe, State and IO.

d. Only State and 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. `IO ()`

b. `String -> String`

c. `String -> IO ()` ✔

d. `String -> IO (String)`

e. `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 higher-order functions. ✖

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

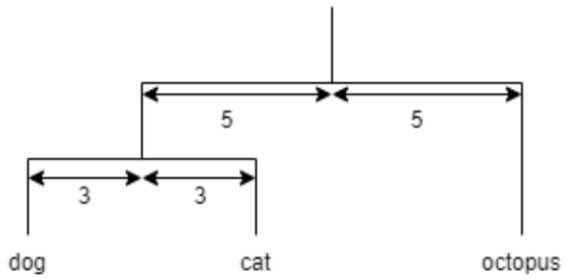
e. the definition of polymorphic functions.

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, Int) | Node Dendrogram Dendrogram`

b. `data Dendrogram = Leaf String | Node Dendrogram Int Dendrogram` ✔

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

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

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