Início	quinta, 27 de outubro de 2022 às 17:37
Estado	Prova submetida
Data de submissão:	quinta, 27 de outubro de 2022 às 19:07
Tempo gasto	1 hora 30 minutos
Nota	**9,34** de um máximo de 14,00 (**67**%)

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 20 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 (name, count) = if(count <= 100) then True else False
```

Pergunta 2

Respondida Pontuou 0,188 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.

```
updateSpecies :: Species -> Int -> Species
updateSpecies (name, oldCount) newCount = (name, newCount)
```

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 [] _ = []
filterSpecies (animal:animals) function = if(function animal) then ([animal] ++ next) else next
   where next = filterSpecies animals function
```

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 animals = sum(map (\(name,count) -> count) animals)
```

Pergunta 5

Respondida Pontuou 1,200 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 start end = take (fromIntegral end+1) (drop (fromIntegral start) xs)
```

Informação

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

Pergunta 6

Respondida Pontuou 1,500 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).

```
hasSubstr :: String -> String -> Bool
hasSubstr xs str = hasSubstrHelper xs str str

hasSubstrHelper :: String -> String -> String -> Bool
hasSubstrHelper _ [] _ = True
```

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)])
```

```
getAnimalName :: Species -> String
getAnimalName (animal, count) = animal

sortSpeciesWithSubstr :: Zoo -> String -> (Zoo, Zoo)
sortSpeciesWithSubstr animals str = (animalHasSubstr animals str, notAnimalHasSubstr animals str)
```

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 0,000 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 …]
```

```
rabbits :: (Integral a) => [a]
rabbits = rabbitHelper [2,3]

rabbitHelper :: (Integral a) => [a] -> [a]
rabbitHelper (xs) = rabbitHelper (xs ++ [(xs !! (length(xs) - 1)) + (xs !! (length(xs)-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
```

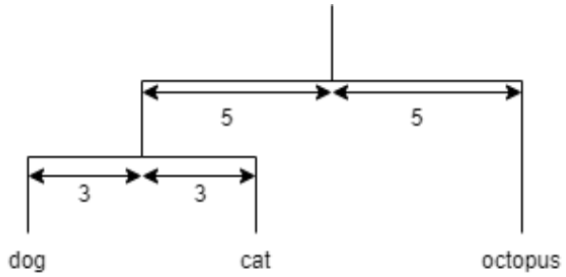
```
rabbitYears :: (Integral a) => a -> Int
rabbitYears year = length ([y | y<-(take (fromIntegral year) rabbits), y<year])
```

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

Respondida Pontuou 0,200 de 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
```

```
dendroWidth :: Dendrogram -> Int
dendroWidth (Leaf str) = 0
dendroWidth (Node left x right) = x*2 + div (max (dendroWidth left) (dendroWidth right)) 2
```

Pergunta 11

Respondida Pontuou 0,000 de 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"]
```

```
dendroInBounds :: Dendrogram -> Int -> [String]
dendroInBounds _ 1 = []
dendroInBounds (Leaf str) _ = [str]
dendroInBounds (Node left x right) limit = (dendroInBounds left (limit-1)) ++ (dendroInBounds right (limit-1))
```

◄ MT1 (27/10/2022) (Part 1)

Ir para...

MT1 - Grades ►