Início	sexta-feira, 8 de novembro de 2024 às 18:00
Estado	Prova submetida
Data de submissão:	sexta-feira, 8 de novembro de 2024 às 18:15
Tempo gasto	15 minutos
Nota	**0,88** de um máximo de 5,00 (**17,5**%)

Informação

A wrong answer in a multiple-choice question (five options) implies a penalty of 25% of the question's value.

Pergunta 1

Pontuou -0,125 de 0,500

Consider the three following expressions:

```
a = [1 .. 100000] ++ (take 1000 [1 .. 100000])
```

```
b = (take 1000 [1 .. 100000]) ++ [1 .. 100000]
```

```
c = (takeWhile (>1000) [1 .. 100000]) ++ [1 .. 100000]
```

Order these expressions by the time needed to evaluate them, starting with the fastest one to compute:

a. c < b < a

b. c < a < b

[✓ c. c < a = b (a and b are computed in the same amount of time)] ✖

d. b < c < a

e. a = b = c (all three expressions take the same amount of time to be computed)

```
GHCi, version 8.8.4: https://www.haskell.org/ghc/  :? for help
Prelude> :set +s
Prelude> let a = [1 .. 100000] ++ (take 1000 [1 .. 100000])
(0.00 secs, 24,872 bytes)
Prelude> let b = (take 1000 [1 .. 100000]) ++ [1 .. 100000]
(0.00 secs, 23,120 bytes)
Prelude> let c = (takeWhile (>1000) [1 .. 100000]) ++ [1 .. 100000]
(0.01 secs, 23,120 bytes)
```

Resposta correta:
c < b < a

Pergunta 2

Pontuou -0,125 de 0,500

What is the result of this expression?

```
concat [[a,b] | (a,b) <- zip ['a' ..] "abcde"]
```

a. `["abcde,"abcde,"abcde","abcde,"abcde"]`

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

[✓ c. `["aa","bb","cc","dd","ee"]`] ✖

d. `"abcdeabcde"`

e. `"aabbccddee"`

Resposta correta:

```
"aabbccddee"
```

Pergunta 3

Pontuou 0,500 de 0,500

What is the result of this expression?

```
zipWith (*) [1 ..] [3,1 .. -5]
```

a. None of the other options.

[✓ b. `[3,2,-3,-12,-25]`] ✓

c. `[-3,-2,3,12,25]`

d. `[]`

e. An infinite list

Resposta correta:

```
[3,2,-3,-12,-25]
```

Pergunta 4

Pontuou -0,125 de 0,500

Which of these expressions has the following type?

```
(Eq a) => [a] -> Bool
```

a. `filter (==3)`

[✓ b. `map (<3)`] ✖

c. `(\l -> reverse l == l)`

d. `(\(_:t) -> tail t)`

e. `tail`

Prelude> :type (\l -> reverse l == l)
(\l -> reverse l == l) :: Eq a => [a] -> Bool

Resposta correta:

```
(\l -> reverse l == l)
```

Pergunta 5

Pontuação 0,500

What is the type of function myFun?

```
myFun x y = (x / y) < 1
```

a. `(Floating a, Ord a) => a -> b -> Bool`

b. `(Floating a) => a -> a -> Bool`

c. `(Num a, Ord a) => a -> a -> Bool`

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

e. None of the other options.

Resposta correta: None of the other options.

Prelude> myFun x y = (x / y) < 1
Prelude> :type myFun
myFun :: (Ord a, Fractional a) => a -> a -> Bool

Pergunta 6

Pontuou -0,125 de 0,500

Consider function f.

```
f [] = Just 1
f (h:t)
  | h > 0 = case f t of Just d -> Just (h*d)
                        _ -> Nothing
  | otherwise = Nothing
```

Se a lista estiver vazia a função retorna 1

Verifica se a head é maior que 0, avalia recursivamente f. Se a chamada recursiva retornar `Just d`, multiplica `h` por `d` e retorna o resultado dentro de `Just`. Caso contrário, se a chamada recursiva retornar `Nothing`, propaga o erro retornando `Nothing`. Se o elemento da cabeça `h` não for maior que 0 (ou seja, for 0 ou negativo), retorna `Nothing`.

Which of these sentences about function f is FALSE?

a. The evaluation of f [0 ..] leads to a result (without infinite recursion) of Just 0.

b. The elements of the input list can have the type Int, Integral and Float (among other valid types).

c. If a given input of f returns a Nothing, its reverse will also return a Nothing.

[✓ d. The result of f [1,2,3,4] is Just 24.] ✗

e. The type of f is f :: (Num a, Ord a) ⇒ [a] → Maybe a

*Main> :type f
f :: (Num a, Ord a) => [a] -> Maybe a

Resposta correta: The evaluation of f [0 ..] leads to a result (without infinite recursion) of Just 0.

Pergunta 7

Pontuou 0,500 de 0,500

Which of the following Prelude functions is equivalent to mysteryFunc?

```
mysteryFunc = foldr (++) []
```

foldr Aplica uma função de forma recursiva à lista, começando pelo lado direito.

a. `length`

b. `tail` Retorna a lista sem o primeiro elemento, não concatena listas.

c. `reverse`

d. `(:)` (adiciona um elemento à frente de uma lista), não concatena listas.

[✓ e. `concat`] ✓

Resposta correta:

```
concat
```

Pergunta 8

Pontuação 0,500

What is the type of function fun?

```
fun f g x y = f (g x y) (f x x)
```

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

b. None of the other options.

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

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

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

Resposta correta:

```
(a -> a -> a) -> (a -> b -> a) -> a -> b -> a
```

*Main> fun f g x y = f (g x y) (f x x)
*Main> :type fun
fun :: (t1 -> t1 -> t1) -> (t1 -> t2 -> t1) -> t1 -> t2 -> t1

Pergunta 9

Pontuou -0,125 de 0,500

What is the result of this expression?

```
foldl (-) 2 (map ((+1).(*2)) [3,2,1])
```

- [✓ a. None of the other options.] ✗
- b. -16
- c. 3
- d. 4
- e. -13

Resposta correta:
-13

Pergunta 10

Pontuou 0,500 de 0,500

What does function g do?

```
g = gAux []  A função principal `g` chama `gAux` com uma lista vazia como argumento inicial

gAux a = do  `gAux` é uma função auxiliar que recebe um acumulador `a`
  x <- getChar  Usa `getChar` para ler um único caractere da entrada do usuário.
  if (x == '\n')  Verifica se o caractere lido é uma nova linha ('\n').
    then return a
    else gAux (x:a) Caso contrário, adiciona o caractere `x` ao início da lista `a` e chama `gAux` recursivamente
```

- a. Reads several lines from the console and prints them in reverse order.
- [✓ b. Reads a whole line from the console and prints it in reverse order.] ✓
- c. Reads several lines from the console and prints them exactly as they were written.
- d. Reads a character from the console and prints it, after appending a '\n' to it.
- e. Reads a whole line from the console and prints it exactly as it was written.

Resposta correta:
Reads a whole line from the console and prints it in reverse order.