Exercícios motivos de

04) Imp avançou + que funcional pq arq von Neumann é imperativo : F(M70) , V(FFO)

03) Resultado de "p": "rograma é "Programa" : F(6MA) ; V(7FM)

06) Resultado de take 4 $ filter (>4) [1..] é [4..] : F(70X) ; V(42W)

12) Stream é um motivador p/ indeix funções lambda em Java : F(2XD) ; V(FWP) ?

13) y ← f x , pode-se concluir que o tipo de f é a -> IO b e o tipo de y é b : F(PTE) ; V(RQW)

16) Aplicação parcial é quando parte da execução ainda não foi concluída : F(6FC) ; V(3FF) ~~Errei~~ (?)

17) A primeira linguagem de prog funcional criada para computadores foi SML : F(AF6) ; V(W8P)

18) Linguagem funcião + eficiente que C e pascal : F(M70) ; V(O8E)

19) zip [1..] ["apple", "orange", "cherry", "mango"] gera um erro : F(R1A) , V(FE8)

23) A primeira linguagem de programação funcional foi LISP : F(6T6) ; V(RQQ)

24) Para simular estados em linguagem funcional, deve-se utilizar passagem de parâmetros : F(2DE) ; V(038)

26) Tipo de h f x = f $ f x é (a->a) -> a -> a : F(2MX) ; V(BBQ)

28) Haskell f ($) x = f x é alternativa a () p/ precedência de função : F(E1Q) ; V(W38)

30) Toda função em Haskell possui um tipo bem definido, que pode ser genérico : F(ECT) ; V(W3C)

34) Funções puras permitem que o compilador otimize o código como necessário : F(6TA) ; V(7Q2)

35) map :: (a -> Bool) -> [a] -> [Bool] : F(ACX) ; V(FBB)

36) map :: (a->b) -> [a] -> [b] : F(2MT) ; V(FBF) ↗ V tem xta no a

37) find :: (a -> Bool) [a] -> Maybe a : F(AM2) ; V(MT9)

39) SML é linguagem derivada de Haskell : F(EC9) ; V(40A)

41) filter :: (a->b) -> [a] -> [b] : F(EA0) ; V(830)

42) O resultado de head $ [1..7] `interect` [5..10] é 5 : F(AF8) , V(7C7)

48) A avaliação lazy permite criar estruturas potencialmente infinitas : F(M6X) ; V(RQQ)

02) De forma simplificada, Lazy Evolution consiste em esperar p/ processar uma expressão (ou parte dela) até o exato momento em que o resultado é utilizado : F(6A4) ; V(3Q6)

08) O conceito de Lazyness é essencial na programação funcional : F(EF1) ; V(7QB)

14) O tipo de f em f x = (head x) : (f $ tail x) é [a] -> [a] : F(2X5) ; V(W32)

20) Varias linguagens imperativas incluem caract de funcional, como C : F(EC7) , V(79R) ↘ Errado

31) Uma função pode ter 1 ou + retornos : F(R3E) ; V(F8W) ?

48) Lazy permite criar estruturas potencialmente infinitas : F(M6X) ; V(RQQ)

I

31) Uma função pode ter 1 ou mais retorn
F(R3E) ; V(F8W

Digitalizado com CamScanner