

## Primeiro Exercício Escolar de PLC (IF686)



Data: 22 de maio de 2014. Tempo disponível: 2h00m. Prof.: Fernando Castor

1. (6,0 ptos) Um Passeio do Cavalo (ou Cavaleiro) é uma sequência de movimentos de um cavalo em um tabuleiro de xadrez tal que esse cavalo visita cada casa exatamente uma vez. Um passeio do cavalo sempre é possível em um tabuleiro quadrado cujo lado tem um número N>5 de casas, onde N é par. A figura abaixo mostra a sequência de movimentos de um cavalo realizando um passeio em um tabuleiro 8x8:

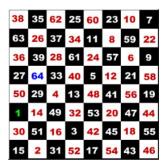


Figura 1: Um passeio do cavalo.

Implemente uma função passeio<br/>Cavalo :: (Int, Int) -> Int -> [(Int, Int)] que, dados uma posição inicial válida no tabuleiro e um inteiro N tal que N > 5 e N 'mod' 2 == 0, devolve uma lista de casas que corresponde a um passeio do cavalo em um tabuleiro quadrado cujo lado tem N casas. Você não precisa ter nenhuma preocupação com eficiência para resolver essa questão.

2. (5,0 ptos) Estude cuidadosamente o trecho de código abaixo e faça o que é pedido.

- (a) (3,0 ptos.) Determine assinaturas das funções f, g, h e (\*\*\*) no trecho de código acima de modo que o código compile (incluindo o valor de result). Considere que F e R são construtores para valores de um mesmo tipo T (que você terá que definir de modo que o código compile!) e que os casamentos de padrões realizados pelas funções g e h são exaustivos para argumentos do tipo T.
- (b) (2,0 ptos.) Determine o tipo da expressão(map.f) F

a)

(\*\*\*) Concatena Strings, logo: (\*\*\*) :: String -> String -> String

Como em h se tem uma árvore de String e em result uma de Int, temos: data T t = F I R t (T t) (T t)

De acordo com result: fff :: (T a) -> tipo de ggg -> tipo de hhh (sabendo que o retorno é uma lista)

g aplica i nos nós da árvore, logo: ggg :: (a -> b) -> (T a) -> (T b)

hhh :: tipo de j -> (T a) -> String -> String, pois retorna uma String ou I (que é o terceiro parâmetro). E como j concatena strings, incluindo da árvore, logo hhh :: (String -> String -> String) -> (T String) -> String -> String

Voltando para f, como temos que h recebe strings, então g seria do tipo ggg :: (a -> String) -> (T a) -> (T Stringo), logo, fff :: (T a) -> ((a -> String) -> (T a) -> ((String -> String -> String) -> (T String) -> String -> String)

Sendo assim, fff recebe uma árvore de inteiros, transforma em árvore de strings com g e finalmente transforma em uma única string com h, semelhante a uma função show.

```
b)
```

```
map.fff map:: (a -> b) -> [a] -> [b]
.:: (d -> e) -> (c -> d) -> c -> e
fff :: (T t) -> ((t -> String) -> (T t) -> (T String)) -> ((String -> String -> String) -> (T String) -> String)

<math display="block">(d -> e) \sim (a -> b) -> [a] -> [b]
d \sim (a -> b)
e \sim [a] -> [b]
(c -> d) \sim (T t) -> ((t -> String) -> (T t) -> (T String)) -> ((String -> String -> String) -> (T String) -> String -> String)
<math display="block">c \sim (T t)
d \sim ((t -> String) -> (T t) -> (T String)) -> ((String -> String) -> (T String) -> String -> String)
```

```
 \begin{array}{l} \text{comparando d} \\ \text{(a -> b)} \sim ((\text{t -> String}) \ -> (\text{T t}) \ -> (\text{T String})) \ -> ((\text{String -> String}) \ -> (\text{T String}) \ -> (\text{T String}) \ -> (\text{T String}) \ -> (\text{T String})) \ -> ((\text{String -> String}) \ -> (\text{T String})) \ -> ((\text{String -> String}) \ -> (\text{T String})) \ -> (\text{T String})) \ -> (\text{T String}) \ -> (\text{T String})) \ -> (\text{T String}) \ -> (\text{T String}) \ -> (\text{T String}) \ -> (\text{T String})) \ -> (\text{T String}) \
```