

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

1. (6,0 pts) 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:

| | | | | | | | |
|----|----|----|----|----|----|----|----|
| 38 | 35 | 62 | 25 | 60 | 23 | 10 | 7 |
| 63 | 26 | 37 | 34 | 11 | 8 | 59 | 22 |
| 36 | 39 | 28 | 61 | 24 | 57 | 6 | 9 |
| 27 | 64 | 33 | 40 | 5 | 12 | 21 | 58 |
| 50 | 29 | 4 | 13 | 48 | 41 | 56 | 19 |
| 1 | 14 | 49 | 32 | 53 | 20 | 47 | 44 |
| 30 | 51 | 16 | 3 | 42 | 45 | 18 | 55 |
| 15 | 2 | 31 | 52 | 17 | 54 | 43 | 46 |

Figura 1: Um passeio do cavalo.

Implemente uma função `passeioCavalo :: (Int, Int) -> Int -> [(Int, Int)]` que, dados uma posição inicial válida no tabuleiro e um inteiro N tal que $N > 5$ e $N \bmod 2 == 0$, 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.

```
1 fff F _ _ = []
2 fff x m n = n (***) (m show x) []
3
4 ggg _ F = F
5 ggg i (R a e d) = R (i a) (ggg i e) (ggg i d)
6
7 hhh j F l = l
8 hhh j (R a e d) l
9 | (R a e d) == (R " " F F) = "" ++ (hhh j e l) 'j' (hhh j d l)
10 | otherwise = a 'j' (hhh j e l) 'j' (hhh j d l)
11
12 (***) a b = a ++ " " ++ b
13
14 result = fff (R 1 (R 2 F F) (R 3 F F)) ggg hhh
```

(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 | 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 `l` (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 String)`, logo, `fff :: (T a) -> ((a -> String) -> (T a) -> (T String)) -> ((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 -> String)`

`(d -> e) ~ (a -> b) -> [a] -> [b]`

`d ~ (a -> b)`

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

`(c -> d) ~ (T t) -> ((t -> String) -> (T t) -> (T String)) -> ((String -> String -> String) -> (T String) -> String -> String)`

`c ~ (T t)`

`d ~ ((t -> String) -> (T t) -> (T String)) -> ((String -> String -> String) -> (T String) -> String -> String)`

comparando d

$(a \rightarrow b) \sim ((t \rightarrow \text{String}) \rightarrow (T\ t) \rightarrow (T\ \text{String})) \rightarrow ((\text{String} \rightarrow \text{String} \rightarrow \text{String}) \rightarrow (T\ \text{String}) \rightarrow \text{String} \rightarrow \text{String})$

$a \sim ((t \rightarrow \text{String}) \rightarrow (T\ t) \rightarrow (T\ \text{String}))$

$b \sim ((\text{String} \rightarrow \text{String} \rightarrow \text{String}) \rightarrow (T\ \text{String}) \rightarrow \text{String} \rightarrow \text{String})$

$c \rightarrow e \sim (T\ t) \rightarrow [((t \rightarrow \text{String}) \rightarrow (T\ t) \rightarrow (T\ \text{String}))] \rightarrow [((\text{String} \rightarrow \text{String} \rightarrow \text{String}) \rightarrow (T\ \text{String}) \rightarrow \text{String} \rightarrow \text{String})]$

$F :: (T\ t)$

$(\text{map.fff})\ F :: [((t \rightarrow \text{String}) \rightarrow (T\ t) \rightarrow (T\ \text{String}))] \rightarrow [((\text{String} \rightarrow \text{String} \rightarrow \text{String}) \rightarrow (T\ \text{String}) \rightarrow \text{String} \rightarrow \text{String})]$