

Dica: Resolva todos os exercícios no papel e, somente quando estiverem prontos, implemente-os no computador.

1. Considere as duas listas a seguir:

ls1 = [3 ,1 ,4 ,1 ,5]

ls2 = [1 ,6 ,1 ,8 ,0]

Mostre a saída das seguintes expressões, quando corretamente executadas. Explique o que acontece quando houver erro.

- a) ls1 > (tail ls2)
- b) (tail ls1) ++ ls2
- c) head (tail (tail ls1))
- d) fst (head ls1 , tail ls2)
- e) 3 ++ ls1
- f) snd (head ls1 , tail ls2)
- g) "1,2,3" ++ ls1
- h) (head (tail ls2), tail ls1)
- i) head ls1 ++ ls1
- j) ls2 ++ [1..7]
- k) [9] ++ ls1 ++ ls2

2. Usando as funções head e tail, defina a função terceiro que devolve o terceiro elemento de uma lista de inteiros.

3. Considere a função reverse do prelude-padrão:

```
> reverse [1 ,2 ,3]
```

```
[3 ,2 ,1]
```

Utilizando essa função (além das funções head e tail, crie as seguintes funções:

(a) Função ultimo, que devolve o ultimo elemento de uma string. Exemplo:

```
> ultimo " haskell "
```

```
'l'
```

(b) Função inicio, que devolve todos os elementos da string, exceto o ultimo. Exemplo:

```
> inicio " haskell "
```

```
" haskel "
```

4. Implemente uma função que receba o primeiro e o último nome de alguém e retorne suas iniciais em uma tupla.

Por exemplo:

```
> iniciais " Haskell " " Curry "
```

```
('H','C ')
```

5. O operador :, chamado construtor, permite construir uma lista a partir de um conjunto de elementos. Execute as seguintes expressões e tente entender o que está sendo feito.

```
> 1:[2 ,3 ,4]
```

```
> 1:2:3:4:[]
```

```
> [1 ,2 ,3]:[4..7]
```

```
> 1:[ 'a','b ']
```

```
> "a": " bCc "
```

[illegible][illegible]