

## Capítulo 6

# Funções revisitadas

### 6.1 Funções recursivas

Nota: Nos exercícios desta aula não pode utilizar a atribuição nem os ciclos `while` e `for`.

1. Escreva uma função recursiva, chamada `soma_digitos_pares`, que recebe um número inteiro não negativo  $n$ , e devolve a soma dos dígitos pares de  $n$ . Não pode usar cadeias de caracteres. Por exemplo,

```
>>> soma_digitos_pares(3456)
10
```

2. Escreva uma função recursiva, chamada `apenas_digitos_impares`, que recebe um número inteiro não negativo  $n$ , e devolve um inteiro composto apenas pelos dígitos ímpares de  $n$ . Se  $n$  não tiver dígitos ímpares, a função deve devolver zero. Não pode usar cadeias de caracteres. Por exemplo,

```
>>> apenas_digitos_impares(468)
0
>>> apenas_digitos_impares(12426374856)
1375
```

3. Escreva uma função recursiva, chamada `sublistas`, que recebe uma lista, e tem como valor o número total de sublistas que esta contém. Por exemplo,

```
>>> sublistas([[1], 2, [3]])
2
>>> sublistas([[[[1]]]])
4
>>>sublistas(['a', [2, 3, [[[1]]], 6, 7], 'b'])
4
```

4. Escreva a função recursiva que recebe três argumentos,  $a$ ,  $b$  e  $n$ , e que devolve o valor de somar  $n$  vezes  $a$  a  $b$ , isto é?,  $b + a + a + \dots + a$ ,  $n$  vezes. A sua função não pode usar a operação  $n \cdot a$ . Por exemplo,

```
>>> soma_n_vezes(3, 2, 5)
17
```

5. Escreva a função recursiva `troca_occ_lista` que recebe uma lista e dois valores,  $a$  e  $b$ , e devolve uma nova lista, obtida a partir da original substituindo todas as ocorrências de  $a$  por  $b$ . Por exemplo,

```
>>> troca_occ_lista([(2, 3), 'a', 3, True, 5], 'a', 2)
[(2, 3), 2, 3, True, 5]
>>> troca_occ_lista([(2, 3), 'a', 3, True, 5], False, 4)
[(2, 3), 'a', 3, True, 5]
>>> troca_occ_lista([], False, 4)
[]
```

6. Escreva a função recursiva `inverte` que recebe uma lista e devolve a lista invertida. Por exemplo,

```
>>> inverte([3, 4, 7, 9])
[9, 7, 4, 3]
```

7. Suponha que a operação `in` não existia em Python. Escreva a função recursiva `pertence` que recebe uma lista e um elemento e devolve `True` se o elemento pertence à lista e `False` em caso contrário. Por exemplo,

```
>>> pertence([3, 4, 5], 2)
False
>>> pertence([3, 4, 5], 5)
True
```

8. Escreva a função recursiva `subtrai` que recebe duas listas e devolve a lista que corresponde a remover da primeira lista todos os elementos que pertencem à segunda lista. Por exemplo,

```
subtrai([2, 3, 4, 5], [2, 3])
[4, 5]
subtrai([2, 3, 4, 5], [6, 7])
[2, 3, 4, 5]
```

9. Escreva a função recursiva `parte` que recebe uma lista de números e um número e que devolve uma lista de duas listas, a primeira lista contém os elementos da lista original menores que o número dado (pela mesma ordem) e a segunda lista contém os elementos da lista original maiores ou iguais que o número dado (pela mesma ordem). Não é necessário verificar a corecção dos dados de entrada. **Sugestão:** Use uma função auxiliar. Por exemplo,

```
>>> parte([3, 5, 1, 4, 5, 8, 9], 4)
[[3, 1], [5, 4, 5, 8, 9]]
```

10. Escreva a função recursiva `maior` que recebe uma lista de números e devolve o maior número da lista. **Sugestão:** Use uma função auxiliar. Por exemplo,

```
>>> maior([5, 3, 8, 1, 9, 2])
9
```