TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 07 (30 minutos) Turno 2ª feira 09:00-10:30
Nome:
Número:
Data:
Curso:

Capítulo 6.1 - Funções recursivas

Escreva a função recursiva `prod_digit_multiplo` que recebe um número inteiro positivo `n` e um elemento `elem`, e devolve o produto de todos os algarismos de `n` que sejam multiplos de `elem`. Por exemplo,

```
>>> prod_digit_multiplo(123456789, 3)
162
>>> prod_digit_multiplo(123456789, 5)
5
```

Nota: Não pode utilizar cadeias de caracteres, atribuição, nem os ciclos `while` e `for`.

Solução:

```
def prod_digit_multiplo(n, elem):
    if n == 0:
        return 1
    elif (n % 10) % elem == 0:
        return (n % 10) * prod_digit_multiplo(n // 10, elem)
    else:
        return prod_digit_multiplo(n // 10, elem)
```

TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 07 (30 minutos) Turno 2ª feira 10:30-12:00
Nome:
Número:
Data:
Curso:

Capítulo 6.1 - Funções recursivas

Escreva a função recursiva `soma_digit_pares` que recebe um número inteiro positivo `n` e devolve a soma de todos os seus algarismos impares. Por exemplo,

```
>>> soma_digit_pares(123456789)
20
>>> soma_digit_pares(135)
0
```

Nota: Não pode utilizar cadeias de caracteres, atribuição, nem os ciclos `while` e `for`.

Solução:

```
def soma_digit_pares(n):
    if n == 0:
        return 0
    elif n % 2 == 0:
        return (n % 10) + soma_digit_pares(n // 10)
    else:
        return soma_digit_pares(n // 10)
```

TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 07 (30 minutos) Turno 5ª feira 10:30-12:00
Nome:
Número:
Data:
Curso:

Capítulo 6.1 - Funções recursivas

Escreva uma função recursiva, chamada `conta_occ`, que recebe uma lista/tuplo de números e um número `n`, e devolve o número de vezes que o número `n` ocorre na lista/tuplo. Por exemplo,

```
>>> conta_occ([1, 2, 3, 4, 3], 3)
2
>>> conta_occ((1, 2, 3, 4, 3), 1)
1
```

Nota: Não pode utilizar a atribuição, nem os ciclos `while` e `for`.

Solução:

```
def conta_occ(lst, n):
    if not lst:
        return 0
    elif lst[0] == n:
        return 1 + conta_occ(lst[1:], n)
    else:
        return conta_occ(lst[1:], n)
```

TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 07 (30 minutos) Turno 5ª feira 14:00-15:30
Nome:
Número:
Data:
Curso:

Capítulo 6.1 - Funções recursivas

Escreva a função recursiva `filtra_impares` que recebe um tuplo contendo inteiros e devolve o tuplo contendo apenas os inteiros impares. Por exemplo,

```
>>> filtra_pares((2, 5, 6, 7, 9, 1, 8, 8))
(2, 6, 8, 8)
>>> filtrapares(())
()
```

Nota: Não pode utilizar a atribuição, nem os ciclos `while` e `for`.

Solução:

```
def filtra_pares(t):
    if not t:
        return t
    elif t[0] % 2 == 0:
        return (t[0], ) + filtra_pares(t[1:])
    else:
        return filtra_pares(t[1:])
```

TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 07 (30 minutos) Turno 6ª feira 10:00-11:30
Nome:
Número:
Data:
Curso:

Capítulo 6.1 - Funções recursivas

Escreva uma função **recursiva** com o nome `junta_ordenadas` que recebe como argumentos duas listas ordenadas por ordem crescente e devolve uma lista também ordenada com os elementos das duas listas. Não é necessário validar os argumentos da sua função. Por exemplo,

```
>>> junta_ordenadas([2, 5, 90], [3, 5, 6, 12])
[2, 3, 5, 5, 6, 12, 90]
```

Nota: Não pode utilizar a atribuição nem os ciclos `while` e `for`.

Solução 1:

```
def junta_ordenadas(l1, l2):
    if not l1 or not l2:
        return l1 + l2
    elif l1[0] < l2[0]:
        return [l1[0]] + junta_ordenadas(l1[1:], l2)
    else:
        return [l2[0]] + junta_ordenadas(l1, l2[1:])
```

TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 07 (30 minutos) Turno 6ª feira 14:00-15:30
Nome:
Número:
Data:
Curso:

Capítulo 6.1 - Funções recursivas

Escreva a função **recursiva** `intersecta_listas` que recebe duas listas e devolve a lista contendo apenas os elementos comuns às duas listas. Por exemplo:

```
>>> intersecta_listas([2, 4, 5, 6, 7, 9], [4, 6, 8, 9])
[4, 6, 9]
```

Nota: Não pode utilizar a atribuição nem os ciclos `while` e `for`. Pode utilizar o operador `in`.

Solução 1:

```
def intersecta_listas(l1, l2):
    if not l1 or not l2:
        return []
    else:
        if l1[0] in l2:
            return [l1[0]] + intersecta_listas(l1[1:], l2)
        else:
            return intersecta_listas(l1[1:], l2)
```

TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 07 (30 minutos) Turno 6ª feira 15:30-17:00
Nome:
Número:
Data:
Curso:

Capítulo 6.1 - Funções recursivas

Escreva uma função **recursiva**, chamada `numero_occ_lista`, que recebe uma lista e um número, e devolve o número de vezes que o número ocorre na lista e nas suas sublistas, se existirem.

```
>>> num_occ_lista([1, 2, 3, 4, 3], 3)
2
>>> num_occ_lista([1, 2, 3, 4, 3], 1)
1
>>> num_occ_lista([1, [[[1]], 2], [[[2]]], 2], 2)
3
```

Nota: Não pode utilizar a atribuição nem os ciclos `while` e `for`.

Solução 1:

```
def num_occ_lista(lst, n):
    if not lst:
        return 0
    elif isinstance(lst[0], list):
        return num_occ_lista(lst[0], n) + num_occ_lista(lst[1:],
n)
    elif lst[0] == n:
        return 1 + num_occ_lista(lst[1:], n)
    else:
        return num_occ_lista(lst[1:], n)
```