TÉCNICO LISBOA

Fundamentos de Programação - 2018/2019 Aula Prática 06 (30 minutos) Turno 2ª feira 09:00-10:30
Nome:
Número:
Data:
Curso:

Capítulo 6.1 - Funções recursivas

Escreva a função **recursiva** `produto_digitos` que recebe um número inteiro positivo e devolve o produto de todos os seus algarismos. Não pode usar cadeias de caracteres. Por exemplo,

```
>>> produto_digitos(345)
60
>>> produto_digitos(3045)
0
```

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

Solução:

```
def produto_digitos(n):
    if n < 10:
        return n
    else:
        return (n % 10) * produto_digitos(n // 10)
```

TÉCNICO LISBOA

Fundamentos de Programação - 2018/2019 Aula Prática 06 (30 minutos) Turno 2ª feira 10:30-12:00
Nome:
Número:
Data:
Curso:

Capítulo 6.1 - Funções recursivas

Escreva a função **recursiva** `soma_digitos_pares` que recebe um número inteiro positivo e devolve a soma de todos os seus algarismos pares. Não pode usar cadeias de caracteres. Por exemplo,

```
>>> soma_digitos_pares(345)
4
>>> soma_digitos_pares(357)
0
```

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

Solução:

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

TÉCNICO LISBOA

Fundamentos de Programação - 2018/2019 Aula Prática 06 (30 minutos) Turno 5ª feira 10:30-12:00
Nome:
Número:
Data:
Curso:

Capítulo 6.1 - Funções recursivas

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)
[]
```

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

Solução:

```
def troca_occ_lista(lst, de, para):
    if lst == []:
        return lst
    elif lst[0] == de:
        return [para] + troca_occ_lista(lst[1:], de, para)
    else:
        return [lst[0]] + troca_occ_lista(lst[1:], de, para)
```

TÉCNICO LISBOA

Fundamentos de Programação - 2018/2019 Aula Prática 06 (30 minutos) Turno 4ª feira 08:00-09:30
Nome:
Número:
Data:
Curso:

Capítulo 6.1 - Funções recursivas

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

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

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

Solução:

```
def filtra_pares(t):
    if 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 - 2018/2019 Aula Prática 06 (30 minutos) Turno 6ª feira 12:30-14:00
Nome:
Número:
Data:
Curso:

Capítulo 6.1 - Funções recursivas

Escreva uma função **recursiva**, chamada `numero_occ_lista`, que recebe uma lista de números e um número, e devolve o número de vezes que o número ocorre na lista. Por exemplo,

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

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

Solução:

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

TÉCNICO LISBOA

Fundamentos de Programação - 2018/2019 Aula Prática 06 (30 minutos) Turno 6ª feira 14:00-15:30
Nome:
Número:
Data:
Curso:

Capítulo 6.1 - Funções recursivas

Usando **recursão**, escreva a função `conta_pares_tuplo` que recebe um tuplo de inteiros e devolve o número de elementos pares no tuplo. Por exemplo,

```
>>> conta_pares_tuplo((4, 5, 6))
2
>>> conta_pares_tuplo((3, 5, 7))
0
>>> conta_pares_tuplo((3, ))
0
```

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

Solução:

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

TÉCNICO LISBOA

Fundamentos de Programação - 2018/2019 Aula Prática 06 (30 minutos) Turno 6ª feira 15:30-17:00
Nome:
Número:
Data:
Curso:

Capítulo 6.1 - Funções recursivas

Usando **recursão**, escreva a função `conta_multiplos_tuplo` que recebe um tuplo de inteiros e um número `m`, e devolve o número de elementos que são múltiplos de `m` no tuplo. Por exemplo,

```
>>> conta_multiplos_tuplo((4, 5, 6), 2)
2
>>> conta_multiplos_tuplo((3, 5, 7), 2)
0
```

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

Solução:

```
def conta_multiplos_tuplo(t, n):
    if not t:
        return 0
    else:
        if t[0] % n == 0:
            return 1 + conta_multiplos_tuplo(t[1:], n)
        else:
            return conta_multiplos_tuplo(t[1:], n)
```