TÉCNICO LISBOA

Fundamentos de Programação - 2019/2020 Aula Prática 07 (30 minutos) Turno 2ª feira 09:00-10:30
Nome:
Número:
Data:
Curso:

Capítulo 6.1 - Funções recursivas

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

```
>>> conta_pares((4, 5, 6))
2
>>> conta_pares([3, 5, 7])
0
>>> conta_pares((3, ))
0
```

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

Solução:

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

TÉCNICO LISBOA

Fundamentos de Programação - 2019/2020 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_impares` que recebe um número inteiro positivo `n` e devolve a soma de todos os seus algarismos impares. Por exemplo,

```
>>> soma_digit_impares(123456789)
25
>>> soma_digit_impares(246)
0
```

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

Solução:

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

TÉCNICO LISBOA

Fundamentos de Programação - 2019/2020 Aula Prática 07 (30 minutos) Turno 5ª feira 10:30-12:00
Nome:
Número:
Data:
Curso:

Capítulo 6.1 - Funções recursivas

Escreva a função recursiva `subst_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,

```
>>> subst_occ_lista([(2, 3), 'a', 3, True, 5], 'a', 2)
[(2, 3), 2, 3, True, 5]

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

>>> subst_occ_lista([], False, 4)
[]
```

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

Solução:

```
def subst_occ_lista(lst, a, b):
    if not lst:
        return lst
    elif lst[0] == a:
        return [b] + subst_occ_lista(lst[1:], a, b)
    else:
        return [lst[0]] + subst_occ_lista(lst[1:], a, b)
```

TÉCNICO LISBOA

Fundamentos de Programação - 2019/2020 Aula Prática 07 (30 minutos) Turno 4ª feira 08:00-09:30
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_impares((2, 5, 6, 7, 9, 1, 8, 8))
(5, 7, 9, 1)
>>> filtra_impares(())
()
```

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

Solução:

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

TÉCNICO LISBOA

Fundamentos de Programação - 2019/2020 Aula Prática 07 (30 minutos) Turno 6ª feira 12:30-14:00
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 - 2019/2020 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 `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 - 2019/2020 Aula Prática 07 (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` que recebe um tuplo/lista de inteiros e um número `m`, e devolve o número de elementos do tuplo/lista que são múltiplos de `m`. Por exemplo,

```
>>> conta_multiplos((4, 5, 6), 2)
2
>>> conta_multiplos([3, 4, 5], 2)
1
```

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

Solução:

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