TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 08 (30 minutos) Turno 2ª feira 09:00-10:30 e 10:30-12:00
Nome:
Número:
Data:
Curso:

Capítulo 7 - Recursão e Iteração

Usando recursão de cauda, escreva a função `conta_impares_tuplo` que recebe um tuplo de inteiros e devolve o número de elementos ímpares no tuplo. Por exemplo:

```
>>> conta_impares_tuplo((4, 5, 6))
1
>>> conta_impares_tuplo((3, 5, 7))
3
>>> conta_impares_tuplo((2, 4))
0
```

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

Solução:

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

TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 08 (30 minutos) Turno 5ª feira 10:30-12:00
Nome:
Número:
Data:
Curso:

Capítulo 7 - Recursão e Iteração

Escreva a função recursiva de cauda chamada `cria_lista_multiplos` que recebe dois números inteiros positivos `n` e `m`, e devolve uma lista com os `m` primeiros múltiplos desse número `n`. Considere que 0 é múltiplo de todos os números. Por exemplo:

```
>>> cria_lista_multiplos(6, 10)
[0, 6, 12, 18, 24, 30, 36, 42, 48, 54]
```

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

Solução 1:

```
def cria_lista_multiplos(n, m):
    def cria_aux(lst,max) :
        if len(lst) == max:
            return lst
        return cria_aux(lst + (lst[-1] + n,), max)
    return cria_aux((0,), m)
```

Solução 2:

```
def cria_lista_multiplos(n, m):
    def cria_aux(lst, mul) :
        if mul == m:
            return lst
        return cria_aux(lst + (n*mul,), mul+1)
    return cria_aux((), 0)
```

TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 08 (30 minutos) Turno 5ª feira 14:00-15:30
Nome:
Número:
Data:
Curso:

Capítulo 7 - Recursão e Iteração

Usando recursão de cauda, escreva a função `soma_divisores` que recebe um número inteiro positivo *n*, e devolve a soma de todos os divisores de *n*. No caso de *n* ser 0 deverá devolver 0. Por exemplo:

```
>>> soma_divisores(20)
42
>>> soma_divisores(13)
14
```

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

Solução:

```
def soma_divisores(num):
    def soma_aux(div, soma):
        if div == 0:
            return soma
        elif num%div == 0:
            return soma_aux(div - 1, soma + div)
        else:
            return soma_aux(div - 1, soma)
    return soma_aux(num, 0)
```

TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 08 (30 minutos) Turno 6ª feira 10:00-11:20
Nome:
Número:
Data:
Curso:

Capítulo 7 - Recursão e Iteração

Escreva a função recursiva de cauda chamada `tuplo_digit_multiplo` que recebe um número inteiro positivo `n` e um elemento `elem`, e devolve o tuplo formado por todos os algarismos de `n` que sejam múltiplos de `elem`. Por exemplo,

```
>>> tuplo_digit_multiplo(123456789, 3)
(3, 6, 9)
>>> tuplo_digit_multiplo(123456789, 5)
(5,)
```

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

Solução:

```
def tuplo_digit_multiplo(num, div):
    def tuplo_aux(num, res):
        if num == 0:
            return res
        elif (num%10)%div == 0:
            return tuplo_aux(num//10, (num%10,) + res)
        else:
            return tuplo_aux(num // 10, res)
    return tuplo_aux(num, ())
```

TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 08 (30 minutos) Turno 6ª feira 14:00-15:30
Nome:
Número:
Data:
Curso:

Capítulo 7 - Recursão e Iteração

Usando recursão de cauda, escreva a função `elimina_occ_tuplo` que recebe um tuplo e um valor `a` e devolve um novo tuplo, obtido a partir do original eliminado todas as ocorrências de `a`. Por exemplo:

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

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

Solução:

```
def elimina_occ_tuplo(tup, val):
    def elimina_aux(tuplo, res):
        if tuplo == ():
            return res
        elif tuplo[0] == val:
            return elimina_aux(tuplo[1:], res)
        else:
            return elimina_aux(tuplo[1:], res + (tuplo[0],))
    return elimina_aux(tup, ())
```

TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 08 (30 minutos) Turno 6ª feira 15:30-17:00
Nome:
Número:
Data:
Curso:

Capítulo 7 - Recursão e Iteração

Usando recursão de cauda, escreva a função `num_divisores` que recebe um número inteiro positivo *n*, e devolve o número de divisores de *n*. No caso de *n* ser 0 deverá devolver 0. Por exemplo:

```
>>> num_divisores(20)
6
>>> num_divisores(13)
2
```

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

Solução:

```
def num_divisores(n):
    def num_div_aux(i, res):
        if i == 0:
            return res
        elif n % i == 0:
            return num_div_aux(i - 1, res + 1)
        else:
            return num_div_aux(i - 1, res)
    return num_div_aux(n, 0)
```