TÉCNICO LISBOA

Fundamentos de Programação - 2018/2019 Aula Prática 08 (30 minutos) Turno 2ª feira 09:00-10:30
Nome:
Número:
Data:
Curso:

Capítulo 7 - Recursão e Iteração

Um número primo é um número inteiro maior do que 1 que apenas é divisível por 1 e por si próprio. Um método simples, mas pouco eficiente, para determinar se um número, `n`, é primo consiste em testar se `n` é múltiplo de algum número entre `2` e $\sqrt{n}$. Usando este processo, escreva uma função recursiva de cauda chamada `primo`, que recebe um número inteiro e tem o valor `True` apenas se o seu argumento for primo. Por exemplo:

```
>>> primo(3)
True
>>> primo(8)
False
```

Solução:

```
def primo(n):
    from math import sqrt
    def primo_aux(i):
        if i < 2:
            return True
        elif n % i == 0:
            return False
        else:
            return primo_aux(i - 1)
    return primo_aux(int(sqrt(n)))
```

TÉCNICO LISBOA

Fundamentos de Programação - 2018/2019 Aula Prática 08 (30 minutos) Turno 2ª 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 um número inteiro positivo, e devolve uma lista com os dez primeiros múltiplos desse número. Considere que 0 é múltiplo de todos os números. Por exemplo:

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

Solução:

```
def cria_lista_multiplos(n):
    def cria_lista_multiplos_aux(lst):
        if len(lst) == 10:
            return lst
        else:
            return cria_lista_multiplos_aux(lst + [lst[-1] + n])
    return cria_lista_multiplos_aux([0])
```

TÉCNICO LISBOA

Fundamentos de Programação - 2018/2019 Aula Prática 08 (30 minutos) Turno 4ª feira 08:00-09:30
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
```

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)
```

TÉCNICO LISBOA

Fundamentos de Programação - 2018/2019 Aula Prática 08 (30 minutos) Turno 5ª feira 10:30-12:00
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
```

Solução:

```
def soma_divisores(n):
    def soma_div_aux(d, res):
        if d == 0:
            return res
        elif n % d == 0:
            return soma_div_aux(d - 1, res + d)
        else:
            return soma_div_aux(d - 1, res)
    return soma_div_aux(n, 0)
```

TÉCNICO LISBOA

Fundamentos de Programação - 2018/2019 Aula Prática 08 (30 minutos) Turno 6ª feira 12:30-14:00
Nome:
Número:
Data:
Curso:

Capítulo 7 - Recursão e Iteração

Usando recursão de cauda, escreva a função `troca_occ_tuplo` que recebe um tuplo e dois valores, `a` e `b`, e devolve um novo tuplo, obtido a partir do original substituindo todas as ocorrências de `a` por `b`. Por exemplo:

```
>>> troca_occ_tuplo(((2, 3), 'a', 3, True, 5), 'a', 2)
((2, 3), 2, 3, True, 5)
>>> troca_occ_tuplo(((2, 3), 'a', 3, True, 5), False, 4)
((2, 3), 'a', 3, True, 5)
>>> troca_occ_tuplo((), False, 4)
()
```

Solução:

```
def troca_occ_tuplo(t, de, para):
    def troca_aux(t, res):
        if not t:
            return res
        elif t[0] == de:
            return troca_aux(t[1:], res + (para,))
        else:
            return troca_aux(t[1:], res + (t[0],))
    return troca_aux(t, ())
```

TÉCNICO LISBOA

Fundamentos de Programação - 2018/2019 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 `soma_pares_lista` que recebe uma lista de inteiros e devolve a soma de todos os elementos pares da lista. Por exemplo:

```
>>> soma_pares_lista([1,2,3,4,5,6,7])
12
>>> soma_pares_lista([])
0
```

Solução:

```
def soma_pares_lista(lst):
    def soma_aux(lst, res):
        if not lst:
            return res
        elif lst[0] % 2 == 0:
            return soma_aux(lst[1:], res + lst[0])
        else:
            return soma_aux(lst[1:], res)
    return soma_aux(lst, 0)
```

TÉCNICO LISBOA

Fundamentos de Programação - 2018/2019 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 `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
```

Solução:

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