Universidade Federal do Espírito Santo

Departamento de Computação e Eletrônica - CEUNES

PROGRAMAÇÃO FUNCIONAL

Prof. Oberlan Romão

Lista de exercícios 3

1. Faça uma função recursiva que:

   a. Imprima todos os números palíndromos de 4 dígitos;
   b. Receba um número n e imprima os números de 1 a n;
   c. Receba dois número a e b e imprima os números naturais entre a e b;
   d. Receba dois inteiros a e b e retorna o valor a^b, sem usar o operador $**$, ou seja, $a**b$ ou a função `pow(a,b)`;
   e. Receba um número inteiro n e retorna a quantidade de divisores de n;
   f. Receba um número inteiro n e retorna `True` se n for primo e `False`, caso contrário;
   g. Receba um número inteiro n e retorna a soma dos dígitos de n. Por exemplo, se $n = 12345$, sua função deve retornar 15.

2. A *sequência de Fibonacci* é uma sequência de termos que tem como os 2 primeiros termos, respectivamente, os números 1 e 1 e os número subsequente é a soma dos dois anteriores. A série de Fibonacci pode ser vista a seguir: $1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, \ldots$

   Em termos matemáticos, a sequência é definida recursivamente pela fórmula abaixo,

$$F_n = \begin{cases} 1, & \text{se } n = 1 \text{ ou } n = 2 \\ F_{n-1} + F_{n-2}, & \text{se } n > 2 \end{cases}$$

   Faça uma função que imprima os n primeiros números da série de Fibonacci. Sua função deve, primeiramente, verificar se n é um número natural. Caso não seja, sua função deve exibir uma mensagem de erro e retornar `None`

3. Um *número perfeito* é aquele cuja soma de seus divisores, exceto ele próprio, é igual ao número. Por exemplo, 6 é perfeito porque 1 + 2 + 3 = 6. Escreva uma função, `ehPerfeito`, que receba um valor inteiro e retorne `True` se o número for perfeito e `False`, caso contrário. Em seguida, faça outra função, `perfeitos`, que imprima todos os números perfeitos de 1 a n. A função `perfeitos` deve utilizar a função `ehPerfeito`. Em ambas as funções deve-se verificar se o valor passado como argumento é um número natural. Faça uma função `main` que solicite ao usuário um número inteiro e que chama a função `perfeitos`.

4. Faça o rastreio (teste de mesa) dos programas abaixo. O que cada um imprime?

   a.
```
def funcao1(arg):
  if arg < 10:
    funcao1(arg + 1)
  else:
    print(arg)

funcao1(0)
```

b.
```python
def funcao2(arg):
  print(arg)
  if arg < 10:
    funcao2(arg + 1)

funcao2(0)
```

c.
```python
def funcao3(arg):
  if arg < 10:
    funcao3(arg + 1)
  print(arg)

funcao3(0)
```

5. (Extraído so site NOIC) Sabemos que $x^a \cdot x^b = x^{a+b}$. Em particular, sabemos que $x^{2b} = x^b \cdot x^b$. Logo, se o expoente n de x^n for par, podemos dizer que $x^n = x^{\frac{n}{2}} \cdot x^{\frac{n}{2}}$. No entanto, se n for ímpar, temos algo similar: podemos afirmar que $x^n = x^{\frac{n-1}{2}} \cdot x^{\frac{n-1}{2}} \cdot x$. Dessa forma, podemos montar a seguinte recorrência:

$$x^n = \begin{cases} 1 & \text{se } n = 0 \\ \left(x^{\frac{n}{2}}\right)^2 & \text{se } n \text{ par} \\ \left(x^{\frac{n-1}{2}}\right)^2 \cdot x & \text{se } n \text{ impar} \end{cases}$$

Implemente uma função recursiva que retorna o valor de x^n usando a recorrência anterior.

6. Os números que compõem o triângulo de Pascal[1] são chamados de números binomiais ou coeficientes binomiais. Um número binomial é representado por $\binom{n}{k}$. Com n e k números naturais e $n \geq k$. Faça uma função recursiva para calcular $\binom{n}{k}$, sabendo que:

$$\binom{n}{k} = \binom{n-1}{k-1} + \binom{n-1}{k}$$

e

$$\binom{n}{0} = \binom{n}{n} = 1 \text{ e } \binom{n}{1} = n$$

7. Faça um programa que, usando a função do exercício anterior, imprima as n primeiras linhas do triângulo de Pascal. Por exemplo, se $n = 10$, a saída seria algo assim:

```
1
1    1
1    2    1
1    3    3    1
1    4    6    4    1
1    5   10   10    5    1
1    6   15   20   15    6    1
1    7   21   35   35   21    7    1
1    8   28   56   70   56   28    8    1
1    9   36   84  126  126   84   36    9    1
```

[1] https://pt.wikipedia.org/wiki/Tri%C3%A2ngulo_de_Pascal

Dica: Implemente uma função recursiva para imprimir os valores de uma linha. Em seguida, usando a função anterior, defina uma função recursiva para imprimir os valores de cada linha.

Desafio: Imprimir o triângulo de Pascal "centralizado":

```
                  1
                1   1
              1   2   1
            1   3   3   1
          1   4   6   4   1
        1   5  10  10   5   1
      1   6  15  20  15   6   1
    1   7  21  35  35  21   7   1
  1   8  28  56  70  56  28   8   1
1   9  36  84 126 126  84  36   9   1
```

8. O número de Euler e pode ser definido por: $e = \sum_{i=0}^{\infty} \frac{1}{n!}$.

   Faça uma função recursiva (chamada `euler`) para calcular o seu valor aproximado através dessa série. A função `euler` deve receber o número de termos a serem usados no cálculo, fazer o cálculo e retornar o resultado. Por exemplo, para 5 termos, o resultado aproximado é o valor de

   $$e = \sum_{i=0}^{4} \frac{1}{n!} = \frac{1}{0!} + \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} = 2,70833$$

   Em seguida, faça uma função `main` que leia o número de termos a serem usados no cálculo e escreva o resultado. Veja um exemplo:

```
Numero de termos: 5
Resultado: 2,70833
```