Programação I

Recursividade (ficha 11)

v1.0

1. À semelhança do comportamento de funções como `min()`, implemente a função `soma_todos()`, que aceita um número variável de argumentos inteiros (0 ou mais) e devolve a respetiva soma. Teste a função.

2. Implemente a versão iterativa e a versão recursiva da função `factorial(n)`.

3. A função `prefixo_comum(s1, s2)` devolve a parte comum no início de duas strings; se não tiverem um prefixo comum, deve devolver a string vazia.

    1. Implemente a versão iterativa da função
    2. Implemente a versão recursiva da função

```
print prefixo_comum("maria","manuel")  # imprime ma
print prefixo_comum("maria","emanuel") # imprime uma string vazia
print prefixo_comum("maria","mariana") # imprime maria
```

4. A função `letras(str)` devolve uma lista com os caracteres em `str`, sem repetição.

```
print letras("banana")    # imprime ['b', 'a', 'n']
```

    1. Implemente a versão iterativa da função
    2. Implemente a versão recursiva da função

5. Um palíndromo é uma palavra que é escrita da mesma forma da esquerda para a direita ou da direita para a esquerda (por exemplo: aia, ala, ama, Ana, arara, matam, radar, reger, rever, reviver, rotor). De um modo recursivo podemos dizer que uma palavra é um palíndromo se a primeira e a última letra são iguais e o "meio da palvra"é um palíndromo. Implemente a função `palindromo(str)` (recursiva) que recebe uma string e devolve `True` se for um palíndromo e `False` de outro modo.

6. Construa a função Ackermann, `A(m,n)` de forma recursiva. `A(m,n)` é definida como:

$$A(m,n) = \begin{cases} n+1 & \text{se } m = 0 \\ A(m-1,1) & \text{se } m > 0 \text{ e } n = 0 \\ A(m-1, A(m,n-1)) & \text{se } m > 0 \text{ e } n > 0 \end{cases}$$

```
>>> A(3,4)
125
```