Peso: 0,5

EXERCÍCIO 08

Instruções:

- Implementar os algoritmos conforme enunciados;
- Compactar o (s) arquivo (s) no formato .zip ou .rar;
- Postar no AVA

1. **Contagem de Dígitos em um Número**: Crie uma função recursiva que receba um número inteiro positivo e retorne o número de dígitos desse número.

python Copiar código

```python
def contar_digitos(n):
    # Caso base: se o número é menor que 10, ele tem apenas 1 dígito
    if n < 10:
        return 1
    else:
        # Recursão: remove o último dígito (n // 10) e conta os dígitos restantes
        return 1 + contar_digitos(n // 10)

# Exemplo de uso
print(contar_digitos(12345))  # Saída: 5
```

2. **Somar Elementos de uma lista**: Escreva uma função recursiva que receba uma lista de números inteiros e retorne a soma de todos os seus elementos.

python Copiar código

```python
def soma_lista(lista):
    # Caso base: se a lista estiver vazia, a soma é 0
    if not lista:
        return 0
    else:
        # Recursão: soma o primeiro elemento com o resto da lista
        return lista[0] + soma_lista(lista[1:])

# Exemplo de uso
print(soma_lista([1, 2, 3, 4, 5]))  # Saída: 15
```

3. **Verificar Palíndromo**: Desenvolva uma função recursiva que verifique se uma palavra é um palíndromo (se a palavra lida de trás para frente é a mesma).

```python
def verificar_palindromo(palavra):
    # Caso base: se a palavra tiver 0 ou 1 caracteres, é um palíndromo
    if len(palavra) <= 1:
        return True
    else:
        # Verifica se o primeiro e o último caracteres são iguais e recorre no meio da palavra
        return palavra[0] == palavra[-1] and verificar_palindromo(palavra[1:-1])

# Exemplo de uso
print(verificar_palindromo("radar"))  # Saída: True
print(verificar_palindromo("python"))  # Saída: False
```

4. **Desenhar um Triângulo de Asteriscos**: Implemente uma função recursiva que desenhe um triângulo de asteriscos com n linhas. Cada linha deve conter um número de asteriscos igual ao número da linha.

```python
def desenhar_triangulo(n, atual=1):
    # Caso base: quando `atual` é maior que `n`, o triângulo está completo
    if atual > n:
        return
    else:
        # Imprime `atual` asteriscos e faz a chamada recursiva para a próxima linha
        print('*' * atual)
        desenhar_triangulo(n, atual + 1)

# Exemplo de uso
desenhar_triangulo(3)
```