Gestão e Qualidade de Software - GQS

Atividade 1 - Em laboratório

1. Função is_par(n)

Escreva função is_par(n) que retorna True se n for par;

Testes: números par, ímpar, zero, número negativo.

Código:

```
def is_par(n):
    if n < 0:
        raise ValueError("Número negativo não permitido")
    return n % 2 == 0

# Testes
print(is_par(4))  # True
print(is_par(7))  # False
print(is_par(0))  # True
print(is_par(-3)) # ValueError: Número negativo não permitido
```

2. Função fatorial(n)

Função fatorial(n), retorna fatorial para n >= 0, levanta ValueError se n < 0;

Testes: fatorial(0) == 1, fatorial(5) == 120, fatorial(-1) levanta ValueError.

Código:

```
def fatorial(n):
```

```
    if n < 0:
        raise ValueError("n deve ser maior ou igual a 0")
    result = 1
    for i in range(1, n+1):
        result *= i
    return result

# Testes
print(fatorial(0))  # 1
print(fatorial(5))  # 120
print(fatorial(-1)) # ValueError: n deve ser maior ou igual a 0
```

3. Classe Conta com Métodos depositar(amount) e sacar(amount).

Levanta ValueError em valores negativos e InsufficientFunds quando tentar sacar mais do que tem;

Testes: depósito, saque com sucesso, saque insuficiente, entradas inválidas.

Código:

```
class Conta:
    def __init__(self, saldo=0):
        self.saldo = saldo

    def depositar(self, amount):
        if amount < 0:
            raise ValueError("Depósito não pode ser negativo")
        self.saldo += amount

    def sacar(self, amount):
```

```
        if amount < 0:
            raise ValueError("Saque não pode ser negativo")
        if amount > self.saldo:
            raise ValueError("Saldo insuficiente")
        self.saldo -= amount

# Testes
conta = Conta(100)
conta.depositar(50)
conta.sacar(120)  # Saldo insuficiente
```

4. Função buscar_clima(cidade)

Função buscar_clima(cidade) que chama: requests.get('https://api.exemplo/clima?cidade=...') e retorna temperatura;

Escreva testes de mock que chamem requests.get e validem que sua função trata corretamente a resposta e exceções (ex.: quando json() não contém temperatura).

Código:

```
import requests
from unittest.mock import patch

def buscar_clima(cidade):
    response = requests.get(f'https://api.exemplo/clima?cidade={cidade}')
    data = response.json()
    if 'temperatura' not in data:
        raise ValueError("Resposta não contém temperatura")
    return data['temperatura']
```

```
# Testes com mock
with patch('requests.get') as mocked_get:
    mocked_get.return_value.json.return_value = {'temperatura': 23}
    print(buscar_clima('Curitiba'))  # 23
```