Disciplina	Prof. João Choma	
PROJETO IMPLEMENTAÇÃO E TESTE DE SOFTWARE	Valor	+01 ATV
ATIVIDADE : TESTE ESTRUTURAL	Aluno: Pedro Paulo	Aluno:Giovanne Leite
ESOFT - 6 - N	Aluno: Pedro Toscano	Aluno:

Atividade prática de teste Estrutural Passos:

1. Projetar **casos de teste Estruturais** para avaliar os quatro algoritmos dos itens listados abaixo. Conforme o exemplo abaixo, e o excerto do Livro Didático.
2. Preencher os ARTEFATOS de teste abaixo para os testes projetados.
3. Construa, em sua linguagem de preferência os seguintes algoritmos:
    a. Um algoritmo que lê um número e imprime a lista dos seus divisores
    b. Um algoritmo que lê dois números e calcula o máximo divisor comum pelo método de Euclides.
    c. Um algoritmo que lê as 4 notas de um aluno e diga se ele passou por média, está em final ou reprovou
    d. Um algoritmo em que dado dois números *n* e *k* (n< k), calcule e apresente a combinatória de n elementos tomados k a k

Exemplo de Desenvolvimento: Derivar os casos de teste para um programa que calcula a média das entradas válidas, usando o método do caminho básico.

Caso A: Algoritmo que lê um número e imprime a lista dos seus divisores

```
def lista_divisores(n):
    if not isinstance(n, int) or n <= 0:
        print("Número deve ser inteiro positivo.")
        return
    print(f"Divisores de {n}:")
    for i in range(1, n + 1):
        if n % i == 0:
            print(i, end=" ")

    print()
```

Passo 1:

Passo 2:

Predicados: 3 (validação n, loop i<=n, if %==0). V(G) = 4.

Passo 3:

- Caminho 1: 1-2-Fim (n inválido)
- Caminho 2: 1-3-4-5-6-7-8-5-Fim (n=1, loop once, % true)
- Caminho 3: 1-3-4-5-6-8-5-6-7-8-5-6-8-5-Fim (n=3, loop com % false e true)
- Caminho 4: 1-3-4-5-6-7-8-5-6-7-8-5-6-8-5-6-7-8-5-Fim (n=4, mais iterações com mix)

Passo 4:

- Caminho 1: Entrada n=0; Esperado: "Número deve ser inteiro positivo."
- Caminho 2: Entrada n=1; Esperado: Divisores de 1: 1
- Caminho 3: Entrada n=3; Esperado: Divisores de 3: 1 3
- Caminho 4: Entrada n=4; Esperado: Divisores de 4: 1 2 4

Caso B: Algoritmo que lê dois números e calcula o máximo divisor comum pelo método de Euclides

```
def mdc_euclides(a, b):
    if not isinstance(a, int) or not isinstance(b, int) or a < 0 or b < 0:
        return "Números devem ser inteiros não negativos."
    while b != 0:
        a, b = b, a % b
    return a
```

Passo 1:

Passo 2:

Predicados: 2 (validação, loop while). V(G) = 3.

Passo 3:

- Caminho 1: 1-2 (inválido)
- Caminho 2: 1-3-5 (b=0, sem loop)
- Caminho 3: 1-3-4-3-5 (loop pelo menos 1 vez)

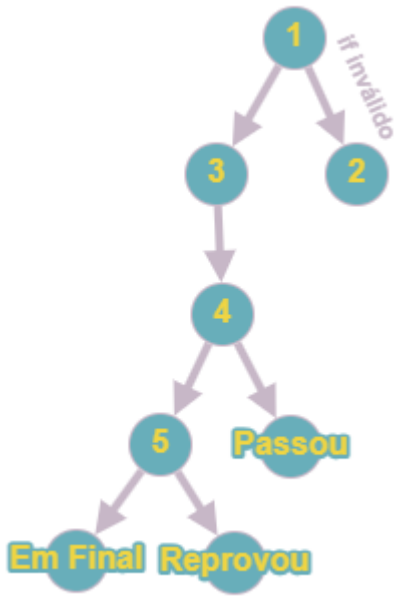
Passo 4:

- Caminho 1: Entradas a=-1, b=2; Esperado: "Números devem ser inteiros não negativos."
- Caminho 2: Entradas a=5, b=0; Esperado: 5
- Caminho 3: Entradas a=15, b=9; Esperado: 3 (loop múltiplo: 15%9=6, 9%6=3, 6%3=0)

Caso C: Algoritmo que lê as 4 notas de um aluno e diz se ele passou por média, está em final ou reprovou

```
def avaliar_aluno(notas):
    if not isinstance(notas, list) or len(notas) != 4 or any(not 0 <= nota <= 10 for nota in notas):
        return "Notas inválidas (devem ser 4 valores entre 0 e 10)."
    media = sum(notas) / 4
    if media >= 7:
        return "passou por média"
    elif media >= 5:
        return "está em final"
    else:
        return "reprovou"
```

Passo 1:



Passo 2:

Predicados: 3 (validação, if >=7, if >=5). V(G) = 4.

Passo 3:

- Caminho 1: 1-2 (inválido)
- Caminho 2: 1-3-4-Return (passou)
- Caminho 3: 1-3-4-5-Return (em final)
- Caminho 4: 1-3-4-5-Return (reprovou)

Passo 4:

- Caminho 1: Notas=[10,10,10]; Esperado: "Notas inválidas..."
- Caminho 2: Notas=[8,8,8,8]; Esperado: "passou por média" (média=8)
- Caminho 3: Notas=[6,6,6,6]; Esperado: "está em final" (média=6)
- Caminho 4: Notas=[4,4,4,4]; Esperado: "reprovou" (média=4)

Caso D: Algoritmo que calcula a combinatória de n elementos tomados k a k

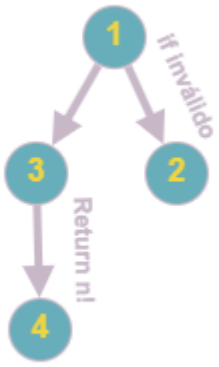
```
def fatorial(m):
    if not isinstance(m, int) or m < 0:
        return 0
    res = 1
    for i in range(2, m + 1):
        res *= i

    return res
```

```
def combinatoria(n, k):
    if not isinstance(n, int) or not isinstance(k, int) or n < 0 or k < 0 or k > n:
        return 0

    return fatorial(n) // (fatorial(k) * fatorial(n - k))
```

Passo 1:



Passo 2:

Predicados: 4 (validação, loops em 3 fatoriais). V(G) = 5.

Passo 3:

- Caminho 1: 1-2 (inválido)
- Caminho 2: 1-3-4 (k=0, fatoriais simples sem loop full)
- Caminho 3: 1-3-4 (n=5, k=1, loop pequeno)
- Caminho 4: 1-3-4 (n=5, k=2, loops com multi false/true em mul)
- Caminho 5: 1-3-4 (n=0, k=0, edge)

Passo 4:

- Caminho 1: n=5, k=6; Esperado: 0
- Caminho 2: n=5, k=0; Esperado: 1
- Caminho 3: n=5, k=1; Esperado: 5
- Caminho 4: n=5, k=2; Esperado: 10
- Caminho 5: n=0, k=0; Esperado: 1 (convencional)

PLANOS DE TESTE A SER DESCRITO :

ITENS A TESTAR / ABORDAGEM:

N°	**Item**	**Especificação**	**ABORDAGEM:**
1	Lista Divisores	Algoritmo a: lê n, imprime divisores	Teste de caminho básico para todos

2	MDC Euclides	Algoritmo b: lê a,b, calcula MDC	
3	Avaliar Aluno	Algoritmo c: lê 4 notas, avalia	
4	Combinatória	Algoritmo d: combinatória de n elementos tomados k a k	

CRONOGRAMA DE TESTES

ID	Tarefa	Início	Fim	Esforço	Pré	Pessoa	Obs
01	Implementar algoritmos	19/09/2025	21/09/2025	3H	–	Pedro Paulo	
02	Projetar casos de teste	19/09/2025	21/09/2025	3H	01	Pedro Paulo	Usar Python
03	Preencher artefatos	19/09/2025	21/09/2025	3H	02	Pedro Paulo	
04	Executar e validar testes	19/09/2025	21/09/2025	3H	03	Pedro Paulo	

AMBIENTE DE TESTE

Ambiente	Descrição
Hardware	Computador pessoal (Intel i5, 16GB RAM)
Software	Python 3.12
Ferramental	VS Code IDLE para edição e execução

IDENTIFICAÇÃO DE CASO DE TESTE / IDENTIFICAÇÃO DE PROCEDIMENTO DE TESTE

Nº	Caso de Teste	Identificação do Caso de Teste		Procedimento	Identificação do Procedimento de Teste
1	Divisores Inválido	CT-DIV-01		Executar função	PT-DIV-01
2	MDC Válido Loop	CT-MDC-01		Executar função	PT-MDC-01
3	Aluno Passou	CT-ALU-01		Executar função	PT-ALU-01
4	Combinatória Inválida	CT-COMB-01		Executar função	PT-COMB-01

5	Combinatória Válida	CT-COMB-02		Executar função	PT-COMB-02

CASO DE TESTE (Exemplo para CT-DIV-01)

Identificação	**CT-DIV-01**	
Itens a Testar	**Lista Divisores (caminho inválido)**	
Entradas	**Campo**	**Valor**
	N	**0**
Saídas Esperadas	**Campo**	**Valor**
		"Número deve ser inteiro positivo."
Ambiente	**Python 3.12**	
Procedimento	**PT-DIV-01**	
Dependência	**Nenhuma**	

PROCEDIMENTO DE TESTE (Exemplo para PT-DIV-01)

Identificação	**PT-DIV-01**
Objetivo	**Testar caminho de entrada inválida no algoritmo de divisores**
Requisitos	**Python instalado, função implementada**
Fluxo	**1. Executar lista_divisores(0)** **2. Verificar saída "Número deve ser inteiro positivo."** **3. Registrar resultado**