#**Nome**: João Mateus dos Santos Dias

#**Email**: joaomateusdossantos2001@gmail.com

#**Tel**.: 945 940 178

#**Whatssap**: 945 940 178

Questão 1:

A senha é um número de 4 dígitos com as condições:

- Todos os dígitos diferentes,
- Soma dos dígitos = 10,
- O número termina com 0.

Resolução:

- Como o número termina com 0, o último dígito é fixo: 0.
- Restam 3 dígitos (nas três primeiras posições) para somar 10.
- Esses 3 dígitos devem ser diferentes entre si e diferentes de 0.

Neste caso, precisamos encontrar todas as combinações de 3 dígitos diferentes de 0 cuja soma seja 10.

Dígitos possíveis: 1 a 9.

Combinações:

- (1, 2, 7)
- (1, 3, 6)
- (1, 4, 5)
- (2, 3, 5)

Essas são as únicas combinações possíveis (considerando a soma 10 e dígitos diferentes).

Agora, para cada grupo de três dígitos, eles podem ser organizados de 6 maneiras.

Então:

- 4 combinações × 6 formas de organizar = **24 senhas possíveis**.

Resposta: 24 senhas possíveis.

Código exemplo em Python:

```
from itertools import permutations

def contar_senhas_possiveis():
    digitos = [1,2,3,4,5,6,7,8,9]
    combinacoes = []
    for p in permutations(digitos, 3):
        if sum(p) == 10:
            combinacoes.append(p)
    total_senhas = len(combinacoes) * 1  # Cada combinação gera 1 senha
adicionando o 0 no final
    return total_senhas

# Resultado:
qtd_senhas = contar_senhas_possiveis()
print(f"Questão 1: Existem {qtd_senhas} senhas possíveis.")
```

Questão 2:

Sequência dada: 2, 6, 12, 20, 30, 42, ?

Padrão:

- 6 - 2 = 4
- 12 - 6 = 6
- 20 - 12 = 8
- 30 - 20 = 10
- 42 - 30 = 12

As diferenças são: 4, 6, 8, 10, 12 — A cada número na sequência sofre um aumento de 2.

Logo, próxima diferença = 14.

Portanto:

- 42 + 14 = **56**.

Resposta: o próximo número é 56.

Código exemplo em Python:

```python
def proximo_na_sequencia(seq):
    diffs = [seq[i+1] - seq[i] for i in range(len(seq)-1)]
    proxima_dif = diffs[-1] + 2
    return seq[-1] + proxima_dif

# Resultado:
sequencia = [2, 6, 12, 20, 30, 42]
proximo = proximo_na_sequencia(sequencia)
print(f"Questão 2: O próximo número da sequência é {proximo}.")
```

Questão 3:

Soma de termos pares da sequência de Fibonacci até N.

Regras:

- N pode ser muito grande (até 10^{18}),
- Considerar apenas termos pares.

Na sequência de Fibonacci, a cada 3 termos aparece um número par. Exemplo:

- **0 (par)**, 1, 1, **2 (par)**, 3, 5, **8 (par)**, 13, 21, **34 (par)**...

Então podemos gerar apenas os termos pares rapidamente, sem calcular todos.

Algoritmo eficiente:

- Começamos com:
    - a = 0 (F(0))
    - b = 2 (F(3))
- Novo termo par: new_term = 4 * b + a
- Atualizamos a e b.

Código exemplo em Python:

```python
def soma_fibonacci_pares(N):
    a, b = 0, 2  # Termos pares iniciais: F(0)=0 e F(3)=2
    soma = 0
    while b <= N:
        soma += b
        a, b = b, 4*b + a  # Geração eficiente apenas de termos pares
    return soma

# Exemplo de uso:
N = 10**18
soma_pares = soma_fibonacci_pares(N)
print(f"Questão 3: A soma dos termos pares de Fibonacci até {N} é
{soma_pares}.")
```