Desafio 1: Lógica de Programação

1. Sequência de Fibonacci

Crie um programa que, dado um número inteiro `n`, retorne os primeiros `n` números da sequência de Fibonacci.

Exemplo de entrada:
`n = 6`

Exemplo de saída:
`[0, 1, 1, 2, 3, 5]`

2. Implementação de Algoritmo de Busca Binária

Implemente um algoritmo de busca binária que, dado um número inteiro alvo e uma lista de números inteiros ordenada de forma crescente, retorne o índice da primeira ocorrência do número na lista. Caso o número não esteja presente, retorne -1.

Regras:

- O algoritmo deve utilizar a estratégia de busca binária (divisão da lista em partes menores).
- A entrada será sempre uma lista já ordenada.

Exemplo de entrada:
Lista: `[5, 12, 18, 23, 45, 70, 89]`
Alvo: `23`

Exemplo de saída:
`3`

---

3. Cálculo de Números Perfeitos

Um número perfeito é um número inteiro positivo que é igual à soma de todos os seus divisores positivos, excluindo ele mesmo. Implemente um programa que verifique se um número dado é perfeito.

Exemplo de entrada:
`n = 28`

Exemplo de saída:
`True`

Explicação:
Os divisores de 28 são: `1, 2, 4, 7, 14`
Soma: `1 + 2 + 4 + 7 + 14 = 28` → Portanto, `28` é um número perfeito.

---

4. Substring Palindrômica Mais Longa

Implemente um programa que, dada uma string, encontre a maior substring palindrômica dentro dela.

Regras:

- Uma substring palindrômica é uma sequência de caracteres que pode ser lida da mesma forma da esquerda para a direita e da direita para a esquerda.
- Caso haja múltiplas substrings de mesmo tamanho, retorne qualquer uma delas.

Exemplo de entrada:
`"babad"`

Exemplo de saída:
`"bab"`` ou "aba"`

---

5. Simulação de Saque em Caixa Eletrônico

Implemente um programa que receba um valor monetário inteiro e retorne a quantidade mínima de notas e moedas necessárias para compor esse valor. O programa deve sempre priorizar as notas de maior valor primeiro.

Notas e moedas disponíveis:

- Notas: `100, 50, 20, 10, 5, 2`
- Moedas: `1`

Regras:

- O valor sempre será inteiro e positivo.
- Deve-se minimizar a quantidade de cédulas e moedas utilizadas.

Exemplo de entrada:
`valor = 130`

Exemplo de saída:

```
1 nota de 100

1 nota de 20
```

1 nota de 10

Desafio 2:

Sistema Web para Consulta e Armazenamento de Endereços

Objetivo:

Desenvolver um sistema web que permita ao usuário consultar um endereço pelo CEP, armazenar os registros consultados e exibir a lista de registros com opções de ordenação.

Requisitos Funcionais:

1. O usuário poderá digitar um CEP e buscar as informações do endereço através da API pública ViaCEP.
2. Os dados retornados devem ser armazenados para consulta posterior. O formato de armazenamento pode ser escolhido pelo candidato (banco de dados, localStorage, JSON, etc.).
3. Deve haver uma interface para listar os endereços armazenados, permitindo que o usuário veja as informações salvas.
4. O usuário poderá ordenar a lista de endereços de forma crescente ou decrescente pelos seguintes campos:
    - Cidade
    - Bairro
    - Estado

Requisitos Técnicos:

✅ A busca do endereço deve ser realizada via **backend**, fazendo uma requisição HTTP para a API do ViaCEP.
✅ A aplicação deve exibir mensagens de erro para CEPs inválidos ou falhas na requisição.
✅ A interface deve ser intuitiva, podendo ser desenvolvida com **HTML, CSS e JavaScript puro ou frameworks como React/Vue/Angular**.
✅ A ordenação dos registros deve ser implementada no backend.
✅ O candidato pode utilizar qualquer tecnologia para armazenar os dados (Ex: localStorage, IndexedDB, banco de dados relacional ou NoSQL).