**Desafio: Sistema de Gerenciamento de Produtos**

Crie um programa que simule o gerenciamento de um estoque de produtos. Cada produto é representado apenas por uma **string** (o nome do produto). O programa deve permitir adicionar produtos, remover, pesquisar por nomes similares, e até simular compras com base no estoque disponível.

**Regras do Desafio:**

1. Use um array chamado estoque para armazenar os produtos.
2. Crie as seguintes funções:
   * adicionarProduto(produto) - Adiciona um novo produto ao estoque (não permite duplicados).
   * removerProduto(produto) - Remove um produto do estoque.
   * buscarProdutosPorTrecho(trecho) - Retorna todos os produtos que contêm o trecho especificado.
   * listarProdutosOrdenados() - Exibe os produtos em ordem alfabética.
   * simularCompra(listaDeCompras) - Recebe um array de produtos desejados e exibe quais podem ser comprados (baseado no estoque).

**Bônus (Desafios Extras):**

1. **Relatório de Estoque**:
   * Crie uma função relatorioEstoque() que exibe o número total de produtos no estoque e a lista completa.
2. **Remover por Posição**:
   * Crie uma função removerPorIndice(indice) que remove um produto com base na sua posição no array.
3. **Resetar Estoque**:
   * Adicione uma função limparEstoque() para limpar completamente o array estoque.

Esse desafio inclui lógica mais complexa, mas ainda utiliza conceitos fundamentais de arrays e métodos como filter, includes, sort, e splice. 🚀

**Dicas Gerais para Resolver o Desafio**

1. **Leia o desafio com atenção:**
   * Antes de começar a escrever, entenda o que cada função deve fazer.
   * Certifique-se de saber como os métodos de array (filter, includes, sort, splice) funcionam.
2. **Teste frequentemente:**
   * Após implementar cada função, teste-a com exemplos simples.
   * Por exemplo, se estiver implementando adicionarProduto, teste com nomes de produtos únicos e duplicados.
3. **Use console.log para entender o que está acontecendo:**
   * Coloque console.log dentro das funções para acompanhar o fluxo de dados.
   * Exemplo: "console.log(estoque)" após adicionar ou remover produtos para verificar o estado do array.
4. **Divida o problema em partes menores:**
   * Se uma função parece complicada, pense no que ela precisa fazer e divida em passos.
   * Por exemplo, na função buscarProdutosPorTrecho:
     + Primeiro, use filter para percorrer o array.
     + Depois, aplique a lógica de verificar se o produto contém o trecho usando includes.
5. **Trate casos especiais:**
   * Pense no que acontece quando o array está vazio. Evite erros verificando o tamanho do array antes de executar ações.
   * Exemplo: No caso de listarProdutosOrdenados, verifique se estoque.length === 0.

**Dicas Específicas para as Funções**

* **adicionarProduto(produto)**:
  + Certifique-se de verificar se o produto já existe no array com includes.
  + Lembre-se de validar a entrada. Por exemplo, você pode usar if (!produto.trim()) para impedir entradas vazias ou apenas espaços.
* **removerProduto(produto)**:
  + Use indexOf para encontrar a posição do produto antes de removê-lo com splice.
  + Sempre verifique se o produto está no array (index !== -1) antes de tentar remover.
* **buscarProdutosPorTrecho(trecho)**:
  + Use o método filter para criar um novo array contendo apenas os produtos que passam no teste.
  + Torne a busca "case insensitive" convertendo tudo para letras minúsculas (toLowerCase()).
* **listarProdutosOrdenados()**:
  + Crie uma cópia do array antes de ordená-lo, para não modificar o array original diretamente.
  + Use o método sort para organizar os produtos em ordem alfabética.
* **simularCompra(listaDeCompras)**:
  + Use o método filter duas vezes para separar os produtos disponíveis e indisponíveis.
  + Certifique-se de que listaDeCompras seja um array antes de começar a lógica.