```php
<?
    // Implementem um algoritmo, de forma mais simples possível, para encontrar um elemento em um vetor de inteiros positivos.
        function busca ( $vetor, $x )
        {
            for ( $i = 0; $i < count ( $vetor ); $i ++ )
            {
                if ( $vetor [ $i ] == $x )
                    return ( $i );
            }

            return ( false );
        }

    // Qual o tempo de execução dos algoritmos no pior caso? Qual você usaria em seu projeto?
    // R: o tempo de execução do primeiro algoritmo foi menor. No meu projeto, eu usaria o primeiro.

    // DESAFIO: implemente uma solução para a busca binária.
        function busca_binaria ( $vetor, $x )
        {
            $menor = 0;
            $maior = ( count ( $vetor ) - 1 );

            while ( $menor <= $maior )
            {
                $meio = floor ( ( ( ( $maior - $menor ) / 2 ) + $menor ) );

                if ( $vetor [ $meio ] < $x )
                    $menor = ( $meio + 1 );
                elseif ( $vetor [ $meio ] > $x )
                    $maior = ( $meio - 1 );
                else
                    return ( floor ( $meio ) );
            }

            return ( false );
        }

    // 1. Implementar a busca linear e a busca binária para um vetor de tamanho 10^9
        for ( $miga = 1; $miga <= ( pow ( 10, 9 ) ); $miga ++ )
            $v [] = $miga;
        echo "Busca linear: " . busca ( $v, 10 ) . "<br>";
        echo "Busca binária: " . busca_binaria ( $v, 10 ) . "<br>";

    // 2. Procure em um vetor ordenado de inteiros o valor mais próximo de um número informado pelo usuário
        function busca_mais_proximo ( $vetor, $x )
        {
            for ( $i = 0; $i < count ( $vetor ); $i ++ )
            {
                if ( ( $x > $vetor [ $i ] ) and ( $x < $vetor [ $i + 1 ] ) )
                {
                    $a = $x - $vetor [ $i ];
                    $b = $vetor [ $i + 1 ] - $x;

                    if ( $a < $b )
                        return ( $i );
                    else
                        return ( $i + 1 );
                }
            }
        }

    // 3. Dado um vetor contendo 2^i elementos no formato {a1,a2,...,an,b1,b2,...,bn}, reorganize os elementos do vetor na seguinte forma {a1,b1,a2,b2,...,an,bn}. Obs.: Não é permitido usar memória adicional.
        function reorganiza ( $vetor )
        {
            for ( $i = 0; $i < ( count ( $vetor ) / 2 ); $i ++ )
            {
```

```
67                  // insere ao final do vetor
68                  $vetor [] = $vetor [ $i ];
69                  $vetor [] = $vetor [ ( count ( $vetor ) / 2 ) + $i ];
70
71                  // retira os "primeiros das filas" como se o vetor tivesse duas
                    filas...
72                  unset ( $vetor [ $i ] );
73                  unset ( $vetor [ $meio + $i ] );
74              }
75          }
76  ?>
```