```php
<?
    // Exercício: Implemente o algoritmo descrito anteriormente (o que percorre o vetor com 1 e 2)
        function exercicio1 ( &$vetor )
        {
            // primeiro, zera a qtd do elemento "1", e pega o tamanho total do vetor
            $c1 = 0;
            $tamanho = count ( $vetor );

            // varre o vetor, contando quantos "1" tem
            foreach ( $vetor as $elemento )
            {
                if ( $elemento == 1 )
                    $c1 ++;
            }

            // preenche as primeiras posições com "1"...
            for ( $i = 0; $i < $c1; $i ++ )
                $vetor [ $i ] = 1;

            // ...e as demais com "2"
            for ( $i = $c1; $i < $tamanho; $i ++ )
                $vetor [ $i ] = 2;

            return ( $vetor );
        }

    // Exercícios: 1. Implementem a função counting_sort.
        function counting ( &$vetor, $inicial, $final )
        {
            $contador = array ();

            for ( $i = $inicial; $i <= $final; $i ++ )
                $contador [ $i ] = 0;

            foreach ( $vetor as $elemento )
                $contador [ $elemento ] ++;

            $aux = 0;

            for ( $i = $inicial; $i <= $final; $i ++ )
            {
                while ( ( $contador [ $i ] -- ) > 0 )
                    $vetor [ $aux ++ ] = $i;
            }
        }

    // 2.Sua função é estável? Caso negativo, qual alteração deveria ser feita para ser?
    // R: não é estável (muda os índices). Para se tornar estável, precisaria manter os índices (levar junto os índices na ordenação).

    // Exercício: Rescreva a função counting sort para trabalhar com valores negativos
        function counting2 ( &$vetor, $inicial, $final )
        {
            $contador = array ();

            for ( $i = $inicial; $i <= $final; $i ++ )
                $contador [ $i ] = 0;

            foreach ( $vetor as $elemento )
                $contador [ $elemento ] ++;

            $aux = 0;

            for ( $i = $inicial; $i <= $final; $i ++ )
            {
                while ( ( $contador [ $i ] -- ) > 0 )
                    $vetor [ $aux ++ ] = $i;
            }
        }
    // A função acima funciona com números negativos.
```

```php

    // Crie um algoritmo baseado no bucket sort para ordenar valores no intervalo [0,1).
        function bucketsort ( $vetor )
        {
            // começa estabelecendo o primeiro elemento como parâmetro de comparação
            $menor = $vetor [ 0 ];
            $maior = $vetor [ 0 ];

            // varre o vetor para buscar o menor e o maior...
            for ( $i = 1; $i < count ( $vetor ); $i ++ )
            {
                if ( $vetor [ $i ] < $menor )
                    $menor = $vetor [ $i ];

                if ( $vetor [ $i ] > $maior )
                    $maior = $vetor [ $i ];
            }

            // define a quantidade de buckets e zera o $b -> subarrays (os buckets)
            $tam = floor ( ( $maior - $menor ) / count ( $vetor ) ) + 1;
            $b = array ();

            for ( $j = 0 ; $j < count ( $vetor ); $j ++ )
            {
                $indice = floor ( ( $vetor [ $j ] - $menor ) / $tam );
                $b [ $indice ] [] = $vetor [ $j ];
            }

            $vetor_organizado = array ();

            for ( $i = 0; $i < count ( $vetor ); $i ++ )
            {
                // organiza os elementos dentro de cada bucket...
                for ( $k = 1; $k < count ( $b [ $i ] ); $k ++ )
                {
                    $aux = $b [ $i ][ $k ];
                    $j = ( $k - 1 );

                    while ( ( $j >= 0 ) and ( $aux < $b [ $i ][ $j ] ) )
                    {
                        $b [ $i ][ $j + 1 ] = $b [ $i ][ $j ];
                        $j --;
                    }

                    $b [ $i ][ $j + 1 ] = $aux;
                }

                // depois de cada bucket organizado, tira os elementos de cada bucket pra um "vetor organizaado"
                for ( $m = 0; $m < count ( $b [ $i ] ); $m ++ )
                    $vetor_organizado [] = $b [ $i ][ $m ];
            }
        }
?>
```