```php
<?
    // Exercícios: Implementar uma solução recursiva para o problema do corte de hastes.
        function cortar ( $n, $p )
        {
            if ( $n <= 0 )
                return ( false );

            $maximo = PHP_INT_MIN;

            for ( $i = 0; $i < $n; $i ++ )
                $maximo = max ( $maximo, $p [ $i ] + cortar ( $n - $i - 1, $p ) );

            return ( $maximo );
        }

    // Exercício: Implementar solução recursiva com memoização
        function cortar_memoi1 ( $n, $p )
        {
            $rm [ 0 ] = 0;

            for ( $i = 1; $i <= $n; $i ++ )
                $rm [ $i ] =- 1;

            return ( cortar_memoi2 ( $n, $p, $rm ) );
        }

        function cortar_memoi2 ( $n, &$p, &$r)
        {
            $q =- 1;

            if ( $rm [ $n ] >= 0 )
                return ( $rm [ $n ];

            if ( $n == 0 )
                $q = 0;
            else
            {
                for ( $i = 1; i <= $n; $i ++ )
                    $q = max ( $q, $p [ $i ] + cortar_memoi2 ( $n - $i, $p, $rm ) );
            }

            $rm [ $n ] = $q;

            return ( $q );
        }

    // Exercício: Implementar solução bottom-up
        function cortar_bottomup ( $n, $p )
        {
            $r [ 0 ] = 0;

            for ( $j = 1; $j <= $n; $j ++ )
            {
                $q = -1;

                for ( $i = 1; $i <= $j; $i ++ )
                    $q = max ( $q, $p [ $i ] + $r [ $j - $i ] );

                $r [ $j ] = $q;
            }

            return ( $r [ $n ] );
        }

    // 1. Verificar para qual tamanho de "n" o algoritmo recursivo de corte de haste para de responder.
    // R: para n=26, ocorre erro de time-limit. Para n=25 já ficou bem lento, mas ainda funcionou.

?>
```