```java
/**
 * Solucións aos exercicios de funcións recursivas (EPyED tema 2)
 *
 * @author Xoan C. Pardo
 * @version Curso 2011-12
 */
public class Recursivas
{
    /**
     * Constructor for objects of class Recursivas
     */
    public Recursivas()
    {
    }

    /**
     * Factorial
     *
     * @param  n >= 0
     * @return factorial de n
     */
    public int factorial(int n)
    {
        if (n==0)
            return 1;   // caso base
        else
            return n * factorial(n-1);   // caso recursivo
    }

    /**
     * Fibonacci
     *
     * @param  n >= 0
     * @return numero fibonacci de orde n
     */
    public long fibonacci(int n)
    {
        if (n <= 1)
            return n;   // caso base
        else
            return fibonacci(n-1) + fibonacci(n-2);   // caso recursivo
    }

    /**
     * Suma de N enteiros
     *
     * @param  n > 0
     * @return suma dos n primeiros enteiros
     */
    public int sumaN (int n)
    {
        if (n==1)
            return 1;   // caso base
        else
            return n + sumaN(n-1);   // caso recursivo
    }

    /**
     * Potencia
```

```java
     *
     * @param  x   base
     * @param  n   potencia (n >= 0)
     * @return     n-ésima potencia de x
     */
    public int potencia(int x, int n)
    {
        if (n == 0)
            return 1;   // caso base
        else
            return x * potencia(x, n-1);   // caso recursivo
    }

    /**
     * Busca nun vector
     *
     * Implementación usando unha función inmersora
     *
     * @param   vector  (length >= 0)
     * @param   x
     * @return  -1 se non se atopa, a posicion se se atopa
     */
    public int busca(int[] vector, int x)
    {
        return buscaInmersora(vector, x, 0); // caso base
    }

    private int buscaInmersora(int[] vector, int x, int pos)
    {
        if (pos >= vector.length)
            return -1;  // caso base 1
        else if (vector[pos] == x)
            return pos; // caso base 2
        else
            return buscaInmersora(vector, x, pos+1);   // caso recursivo
    }

    /**
     * Suma dun vector
     *
     * Implementación usando unha función inmersora
     *
     * @param   vector  (length >= 0)
     * @return  suma dos elementos do vector
     */
    public int sumaVector(int[] vector)
    {
        return sumaInmersora(vector, 0); // caso base
    }

    private int sumaInmersora(int[] vector, int pos)
    {
        if (pos >= vector.length)
            return 0; // caso base
        else
            return vector[pos] + sumaInmersora(vector, pos+1);   // caso
                recursivo
    }

    /**
```

```java
 * Producto escalar
 *
 * Implementación usando unha función inmersora
 *
 * @param   a  (length >= 0)
 * @param   b  (length >= 0)
 * @pre     length(a) == length(b)
 * @return  producto escalar de a e b
 */
public int produtoEscalar(int[] a, int[] b)
{
   return produtoInmersora(a, b, 0); // caso base
}

private int produtoInmersora(int[] a, int[] b, int pos)
{
    if (pos >= a.length)
        return 0; // caso base
    else
        return a[pos]*b[pos] + produtoInmersora(a, b, pos+1);   // caso
            recursivo
}

/**
 * Buscar elemento repetido
 *
 * Implementación usando unha función inmersora
 *
 * @param   a  (length >= 0)
 * @return  -1 se non hai repetidos, posición do primeiro elemento
     repetido
 */
public int repetido(int[] vector)
{
    return repetidoInmersora(vector, 0);  // caso base
}

private int repetidoInmersora(int[] vector, int pos)
{
    if (pos >= vector.length)
        return -1;   // caso base 1
    else if (buscaInmersora(vector, vector[pos], pos+1) != -1)
        return pos;  // caso base 2
    else
        return repetidoInmersora(vector, pos+1);   // caso recursivo
}

/**
 * Maximo común divisor por Euclides
 *
 * @param   a
 * @param   b
 * @return  mcd de a e b
 */
public int mcd(int a, int b)
{
   if (b == 0)
        return a; // caso base
   else
        return mcd(b, a % b);   // caso recursivo
```

```
    }
}
```