



### 1. Recorrido de una cadena de caracteres

```
/*  
 * Pre: ---  
 * Post: Ha devuelto el número de caracteres de la cadena «cad» cuyo valor es el de una letra  
 * mayúscula o minúscula del alfabeto inglés.  
 */  
unsigned int contarLetras(const string cad);
```

### 2. CamelCase

Se denomina *CamelCase* al estilo de escritura que se aplica a frases o palabras compuestas, cuando las palabras simples que las componen se escriben todas juntas unas a otras, sin dejar espacios intermedios, y los inicios de las distintas palabras se marcan intercalando letras mayúsculas, con la posible excepción de la letra inicial de la primera palabra. Por ejemplo, el apellido escocés *MacLean*, la marca *CinemaScope*, la fórmula química *NaCl* o el identificador de C++ *numeroPalabrasEnCamelCase* son ejemplos en los que se ha aplicado el estilo *CamelCase*.

Diseña la siguiente función C++:

```
/*  
 * Pre: Todos los caracteres de «cadena» son letras del alfabeto inglés.  
 * Post: Ha devuelto el número de palabras individuales que forman «cadena»  
 */  
unsigned int numeroPalabrasEnCamelCase(string cadena);
```

A modo de ejemplo, las siguientes invocaciones a `numeroPalabrasEnCamelCase` deberían producir los siguientes resultados:

<code>numeroPalabrasEnCamelCase("")</code>	debe devolver 0
<code>numeroPalabrasEnCamelCase("a")</code>	debe devolver 1
<code>numeroPalabrasEnCamelCase("EINA")</code>	debe devolver 1
<code>numeroPalabrasEnCamelCase("camelCase")</code>	debe devolver 2
<code>numeroPalabrasEnCamelCase("iPad")</code>	debe devolver 2
<code>numeroPalabrasEnCamelCase("ArrayIndexOutOfBoundsException")</code>	debe devolver 6
<code>numeroPalabrasEnCamelCase("numeroPalabrasEnCamelCase")</code>	debe devolver 5

### 3. Basado en el problema 2 del examen de 2.ª convocatoria del curso 2017-18

Escribe el cuerpo de la siguiente función:

```
/*  
 * Pre: «numero» < 100.  
 * Post: Devuelve la cadena de caracteres resultante de concatenar la cadena  
 * «prefijo» con la representación decimal de exactamente dos dígitos de  
 * «numero» y con la cadena «sufijo».  
 */  
string concatenar(const string prefijo, const unsigned int numero, const string sufijo);
```

El enunciado del examen de 2.ª convocatoria del curso 2017-18 (disponible en Moodle) te puede dar más información sobre el contexto en el que se utiliza esta función. En cualquier caso, en ese curso se trabajó con *null terminated strings* en lugar de con la clase `string`.



### 4. Basado en el problema 3 del examen de 1.<sup>a</sup> convocatoria del curso 2018-19 (2,5 puntos)

Escribe el código de la función limpiar cuya especificación se muestra a continuación:

```
/*
 * Pre: «palabra» es una cadena de caracteres que no contiene ningún
 * carácter blanco (ni espacios en blanco, tabuladores o caracteres de
 * cambio de línea).
 * Post: Devuelve una cadena de caracteres cuyo contenido es el de las letras
 * del alfabeto inglés de «palabra», en el mismo orden que en «palabra»,
 * pero siempre en sus versiones minúsculas. Cualquier otro carácter de
 * «palabra» no ha sido copiado en la cadena devuelta.
 *
 * Ejemplos:
 *
 * «palabra»      Valor devuelto
 * -----
 * ""             ""
 * "En"           "en"
 * "un"           "un"
 * "Mancha,"      "mancha"
 * "corredor."    "corredor"
 * "- ¡Oh!"       "oh"
 * "¿Duermen?"    "duermen"
 * "1604"         ""
 * "H2S04"        "hso"
 */
string limpiar(const string palabra);
```

El enunciado del examen de 2.<sup>a</sup> convocatoria del curso 2017-18 (disponible en Moodle) te puede dar más información sobre el contexto en el que se utiliza esta función. En cualquier caso, en ese curso se trabajó con *null terminated strings* en lugar de con la clase `string`. La solución utilizando objetos de la clase `string` es sensiblemente más sencilla.