Recorrido de una cadena de caracteres

Problema 1

Escribe la especificación, la cabecera y el código de una función que, dado un **`string`**, devuelva el número de caracteres de ese **`string`** que son letras del alfabeto inglés mayúsculas o minúsculas.

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*.

Problema 2

Escribe ahora una función que, dado un vector de cadenas de caracteres, donde cada cadena representa una palabra, devuelva una cadena de caracteres en el que las palabras del vector han sido concatenadas utilizando *CamelCase*:

```
/*
 * Pre:  El vector «palabras» tiene al menos «numPalabras» componentes y todos los caracteres de
 *       todas las componentes del vector «palabras» son letras del alfabeto inglés.
 * Post: Devuelve la cadena resultante de concatenar todas las palabras del vector «palabras»
 *       utilizando CamelCase.
 */
string concatenarEnCamelCase(const string palabras[],
                             const unsigned numPalabras);
```

La primera palabra del vector debe mantenerse en la cadena resultante tal y como está en el vector de entrada.

A modo de ejemplo, se muestran los resultados que debería devolver la función, dadas las siguientes declaraciones:

```
const string QUIJOTE[] = {"en", "un", "lugar", "de", "la", "mancha", "de", "cuyo", "nombre",
       "no", "quiero", "acordarme"};
const string JAVA[] = {"Array", "index", "out", "of", "bounds", "exception"};
const string SAL[] = {"", "na", "", "", "cl", ""};
```

`concatenarEnCamelCase(QUIJOTE, 12)`	debe devolver "enUnLugarDeLaManchaDeCuyoNombreNoQuieroAcordarme"
`concatenarEnCamelCase(QUIJOTE, 0)`	debe devolver ""
`concatenarEnCamelCase(JAVA, 6)`	debe devolver "ArrayIndexOutOfBoundsException"
`concatenarEnCamelCase(SAL, 6)`	debe devolver "NaCl"

Problemas basados en exámenes

Problema 3

(Basado en el problema 2 del examen de 2ª 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 numero, const string sufijo);
```

El enunciado del examen de 2ª convocatoria del curso 2017-18 (disponible en Moodle) te pude 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`**.

En la solución de esta versión, puede ser útil la función `to_string`, declarada en el módulo de biblioteca predefinido `<string>`: http://www.cplusplus.com/reference/string/to_string/

Problema 4

(Basado en el problema 3 del examen de 1ª 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"             ""
 *       "H2SO4"            "hso"
 */
string limpiar(const string palabra);
```

El enunciado del examen de 2.ª convocatoria del curso 2017-18 (disponible en Moodle) te pude 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` en sensiblemente más sencilla.