Lab 4: ASCII Number Manipulation

Write a program that manipulates the following scrambled string to produce useful display information on the LCD, as described below:

`"This4iS7a2sTRING!"`

1. [2 marks] Your program will parse out the non-numeric characters into another string, replacing the numeric characters with spaces. Then, the top line of the LCD display will show just the non-numeric characters, with all alphabetical characters in uppercase, with spaces to replace the numeric characters, on the top line, as follows:
   `"THIS IS A STRING!"`
2. [1 mark] The second line will show the same characters, but in lowercase, as follows:
   `"this is a string!"`
3. [2 marks] Your program will also parse out the numeric characters and add them together. Once this is done, the third line will show the sum of the numeric characters, as a hexadecimal value, as follows:
   `"0x0D"`
4. [2marks] Finally, your program will show the sum of the numeric characters, as a decimal (i.e. BCD) value on the fourth line, as follows, for the given string:
   `"13"`

To verify that you are actually doing the math and the numeric manipulation, your instructor may ask you on-the-fly to change the string to something like `"onE9MoRe7f0R8Fun?"`.

Code Submission: [3 marks]

Place your "main.c" file for this project, clearly identified as yours, in the Lab 4 Dropbox so your instructor can grade your descriptive file headers, your code documentation and your use of libraries, and, where appropriate, discuss your coding techniques. Also include your "LCD_Lib.c" library.

Hints:

1. When declaring a constant string, you can do something like `char String[]="Hello!";` The CodeWarrior cross-compiler makes the string long enough for, and includes, the NULL.
2. When declaring an empty variable string, you need to indicate the length, including space for a NULL terminator, like `char AnotherString[18];` Write a NULL into the last location.
3. One way to handle this exercise is to build four separate strings, one each for uppercase and lowercase, one each for the hex and BCD representations of the sum.
4. Step through the raw string character by character, watching for the NULL terminator (0x00).
5. The 'numbers' in the raw string are ASCII characters, so you'll have to make them into real numbers (i.e. hexadecimal) before you can add them.
6. If you want BCD, you have to start with hexadecimal values – the BCD converter doesn't work on ASCII values!
7. However, to display them on the LCD, your numbers have to be converted back, character by character, to ASCII.
8. Since there's no point in continuously looping through the program, end it with a HALT.