Programming Assignment #4

This short assignment will give you more practice with pointers. The program you write will scan through bytes (characters) performing various operations as described below. There are seven functions that you need to implement to complete the assignment.

Functions and Descriptions
`const char *find_byte(const char *from, const char *to, char byte);` Given two pointers(*from* and *to*), scan the range looking for the first occurrence of *byte*, returning a pointer to it. If *byte* is not found, return NULL. The pointers are guaranteed to be pointing into the same array and that the address of *from* is less than or equal to the address of *to*.
`const char *find_any_byte(const char *from, const char *to, const char *bytes, int count);` Given two pointers (*from* and *to*), scan the range looking for the first occurrence of any byte that is in the *bytes* array, returning a pointer to it. If none of the characters in the *bytes* array is found, return NULL. The count parameter tells you how many characters are in the *bytes* array. The pointers are guaranteed to be pointing into the same array and that the address of *from* is less than or equal to the address of *to*.
`int count_bytes(const char *from, const char *to, char byte);` Given two pointers(*from* and *to*), scan the range counting the number of occurrences of *byte*. The return value is the number of times that *byte* appeared in the range. returning a pointer to it. The pointers are guaranteed to be pointing into the same array and that the address of *from* is less than or equal to the address of *to*.
`int count_any_bytes(const char *from, const char *to, const char *bytes, int count);` Given two pointers (*from* and *to*), scan the range counting occurrences of any character that is in the *bytes* array. The return value is the sum of all of the characters found. The count parameter tells you how many characters are in the *bytes* array. The pointers are guaranteed to be pointing into the same array and that the address of *from* is less than or equal to the address of *to*.
`int compare_bytes(const char *location1, const char *location2, int count);` Given two pointers(*location1* and *location2*), compare each character byte-by-byte to determine if the ranges are the same (i.e. contain the same characters in the same order). If the bytes (characters) are all the same, the function returns 0. If the the bytes pointed to by *location1* are less than the bytes pointed to by *location2*, the function returns a negative number (any negative number). Otherwise, the function returns a positive number (any positive number). The count tells the function how many bytes to compare.
`void exchange_bytes(char *location1, char *location2, int count);` Given two pointers(*location1* and *location2*), exchange (swap) the bytes from each. The count parameter tells you how many bytes to swap. You are guaranteed that the two locations do not overlap in memory. Notice that the pointers are not marked const. This is because you are modifying the characters in both ranges. You are not allowed to create any arrays. You must do the exchange within the ranges provided.
`void copy_bytes(char *from, char *to, int count);` Given two pointers(*from* and *to*), copy count bytes (characters) from the *from* pointer to the *to* pointer. The count parameter tells you how many bytes to copy. It is possible that the two ranges overlap, so you must handle this! This means you will need to think before you write any code. Notice that the pointers are not marked const, meaning that you are modifying the arrays in-place. You are not allowed to create any new arrays in your code.

Notes: (Failure to follow these instructions may cause you to fail the assignment.)

1. Pay attention to the constness of the parameters. If a pointer is marked const, then you are only reading the data that is pointed to; you won't be writing any data. If the pointer is NOT marked const, then you will be modifying the data that is pointed at and it will likely be random garbage.
2. You can not use the subscript operator anywhere in your code (not even in a comment). You must use pointer notation for everything. I will be checking to see if either the left bracket [or right bracket] appears anywhere in your submission. You'll receive an automatic 0 if those are found anywhere (for failure to follow instructions).
3. You can not create any arrays in your code. This is simply because you don't need any. Creating an array will cause you to receive a 0. Again, you don't need any arrays. If you think you need one, then you don't understand the assignment. Ask questions.
4. If you think before you code, you will see that you can call `find_byte` from the `find_any_byte` function and you can call the `count_bytes` from the `count_any_bytes` function. This is called code reuse and programmers strive to do this as much as possible.
5. You should work on the functions in the order listed, since they are listed roughly by the order of difficulty.
6. If you wish to write additional "helper" functions to use, they must be marked with the **`static`** keyword.
7. You must not include any header files in your `bytes.c` file as you don't need them. The exception is stddef.h, which is already included (for the definition of NULL). **Including any other header files will result in an automatic 0.**
8. There is additional information posted on the website with a test driver that demonstrates all of the functions.

Details

You are given a file called **`driver.c`**, which includes the main function with several test cases for you to use. There are seven functions prototyped in the driver, one for each of the functions you are to write. As always, you must implement these functions *exactly as prototyped.* A sample command line to compile this assignment looks like this:

```
gcc -O -Wall -Wextra -Werror -ansi -pedantic driver.c bytes.c PRNG.c -o bytes
```

This will compile and link both files and produce an executable file named **`bytes.exe`** on Windows. This assignment will not require you to include any header files other than the one already included in your code, so don't. All of the code that you write must be in `bytes.c`, since you will not be turning in any other files.

All of the functions will be doing some sort of looping. You can use either a **for** loop, **while** loop, or **do** loop.

What to submit

You must submit **`bytes.c`** to the appropriate submission server.

Refer to the web page for this assignment for any additional details on how to submit this assignment. **Do not submit any other files than the ones listed.**

Files	Description
bytes.c	The C file. This file contains all of the source code for the program, properly formatted and documented as discussed in class.

If you've forgotten how to submit files, the details about how to submit are posted on the course website and in the syllabus. Failure to follow the instructions will result in a poor score on the assignment.

Make sure your name and other info is on all documents (paper and electronic).