One-Dimensional Arrays

This exercise will give you a chance to work with one-dimensional arrays and to consider the problem of buffer overflow. On the course homepage and in the following directory, you'll find a source file with a partial implementation, hello.c.

/afs/eos.ncsu.edu/courses/csc/csc230/common/www/sturgill/exercise07

In C, strings are normally stored in character arrays, with a null character to mark the end. Of course, we don't have to follow this convention. For this exercise, you're going to store a string as a sequence of characters in an array, without a null marking the end. This will let us use the whole array to store characters (rather than having to reserve space for the null), but we will need a separate variable to keep up with the length and won't be able to use the standard string-handling functions (which is fine since we don't know them anyway).

The hello.c program contains a 10-element char array for holding a name and an int variable for holding the length of the name. Add code after the prompt to read in a name, one character at a time using getchar(), storing characters in the name array. Read until you see a newline (\n), until you fill the 10-character array or until you reach end-of-file.

After reading the name, print out a hello message as illustrated below. Your string won't have a null marking the end, so you'll have to write a loop to print it one character at a time (rather than getting something like printf() to print the whole name for you). If the given string exceeds the capacity of your name array, stop reading and print the error message illustrated below:

You should be able to run the program with a short, ordinary name:

```
$ ./hello
What's your name: David
Hello David
```

Or, you can run it with a name that just barely fits in the array:

```
$ ./hello
What's your name: Bernadette
Hello Bernadette
```

If you run it with a name that's too long, the program won't overflow the buffer. Instead, it will stop and print an error message:

```
$ ./hello
What's your name: Christopher
That name is too long
```

When you're done submit your submit your completed hello.c file via WolfWare Classic.s