Lab Assignment #11

This week's assignment will have you write a program that has you create your own simple version of the C++ string class. Your solution must not use the standard C++ string class though (only C-style strings). Feel free to work in pairs and ask for help early.

Write a program that defines a `class` called **`String`**. The class must have a data member that is an array of 80 characters to store the characters of the string (therefore, the largest number of characters in an object of type **`String`**, including the null character at the end, is 80). A proper empty object of type String should have a single character it in, the null character (**`'\0'`**), located in the first array element.

The public interface of the class must provide the following:

- A *default constructor* that creates an empty string
- A *constructor* that accepts a C-string parameter and sets the internal character array with it
- A member function called **`copy()`** that accepts a single C-string parameter and copies the value of the argument into the current object, overwriting whatever was in the object before (see NOTE below)
- A member function called **`concat()`** that accepts a single C-string parameter and appends the value of the argument to the end of the internal string in the object (see NOTE below)
- A member function called **`size()`** that returns the size of the internal string in the object, not including the null character
- A member function called **`str()`** that returns a *constant pointer* to the current C string in the object, that same as the `c_str()` member function of the C++ string class does.

NOTE: If the C-string passed to the *single parameter constructor*, **`copy()`** or **`concat()`** member functions would cause the internal character array to be overflowed, then the function should copy whatever can be copied and make sure that the last character is set to the null character (**`'\0'`**).

You are free to use any of the C-string library functions, if you like, or you may implement the functionality without using any library functions. It is up to you.

Your program will consist of three files:

- A header file named `String.h`
- An implementation file for the header file, named `String.cpp`
- A main program file named `main.cpp`

The main program, `main.cpp`, has been provided for you. You will use this file as-is, except for the addition of your preamble. It is your task to create the header and implementation files and then write the correct class definition and member function definitions to make the main work. While the main to use for your final submission is provided, you will likely want/need to use some intermediate driver programs to test and verify your work as you make progress.

Submitting your work

Each of your source code files must have a preamble at the top (see the programming rules and guidelines document in Blackboard for more on the preamble).

Make sure you are in the folder with your source code files (use the `ls` command) and then run the following to create a zip archive of them:

```
$ mkdir lastname_firstname_lab11
$ cp main.cpp lastname_firstname_lab11/main.cpp
$ cp String.h lastname_firstname_lab11/String.h
$ cp String.cpp lastname_firstname_lab11/String.cpp
$ zip -r lastname_firstname_lab11.zip lastname_firstname_lab11/
```

You'll need to change `lastname` and `firstname` to your actual last and first names (**use lowercase letters**) in the steps above. Once you have your zipfile, you can use that file as your submission for the assignment in Blackboard.

Sample output

```
$ ./lab11
s1        =
s1.size() = 0
s2        = Test string
s2.size() = 11
s3        = A test string that is definitely going to be too long and should therefore be t
s3.size() = 79

s1        = Another test string
s1.size() = 19
s2        = The fourth test string is fairly long, although just shy of being truncated.
s2.size() = 76
s3        = Another test string that is definitely too long and should therefore be truncat
s3.size() = 79

s1        = Another test string: is still a string,
even with a newline.
s1.size() = 60
s2        = The fourth test string is fairly long, although just shy of being truncated.PAR
s2.size() = 79
s3        = Another test string that is definitely too long and should therefore be truncat
s3.size() = 79

s1        = Final s1 string.
s1.size() = 16
s2        = Final s2 string.
s2.size() = 16
s3        = Final s3 string.
s3.size() = 16
```