Assignment P6

Due Date: May 2

Purpose
This (last!) project is meant to bring together many aspects of OOP, including classes, methods, overloaded operators, separate files, and more practice with arrays in the form of strings.

Problem
I'm old school (well, just old), so I still sometimes use character arrays when I need to work with strings. But that's silly, because C++ (as opposed to C) has the `string` type and a built-in string library. However, *somebody* had to implement that library. That *somebody* will be *you*! You will implement your own version of the C++ `string` type, called `String` (the capital 'S' is crucial!).

Input
There is no input *per se* in this assignment; rather, you should write a small "driver" program (i.e., `main()`) that uses all of the operators and functions described below. Your driver program is entirely up to you. I will write my own `main()` to test your library. Because of this, it is **essential** that you name and implement functions and operators **exactly** as described below. An abbreviated sample program looks like the following (this does **not** test everything adequately!):

```
#include "Gousie_P6.h"  // your .h file is the only thing that should be included

int main() {
   String s ("This ");  // sets s to "this "
   String t;            // sets t to a null string

   cin >> t;
   s += t;
   cout << s << endl;
   s += " that";
   cout << s << endl;
   cout << s[0] << endl;
   s = "Overloaded operators sure are fun!";
   cout << s << endl;
   cout << s.length() << endl;

   return 0;
}
```

Output
The output depends on the test program and what the user types in. For example, if the above were run, and the words "and all" were typed in to satisfy the input statement, the output screen (minus the input you typed in) would look exactly as follows:

```
This and all
This and all that
T
Overloaded operators sure are fun!
34
```

Specifics

You must create a class called `String`. This class will create a String object that has many of the same capabilities of a C++ string type variable/object. This class should contain definitions for the features described below:

- You may assume that the maximum size string will be 100 characters. Thus, you do not have to worry about doing dynamic memory allocation (although that is how it is actually done in the C++ `string` class). You do not have to do error checking for the length of any string.
- a default constructor and a copy constructor. The default constructor should set the string to null. The copy constructor should copy the argument which is a character array (see example above) to the class instance. Note that this copy constructor is crucial for assignments (the = operator) to work properly.
- an overloaded >> operator the will let the user type in a string terminated by the `enter` key. You may assume the input will be correct. Note this is not the way this operator works with the C++ string class, as a string will also be terminated by a blank space in that implementation.
- an overloaded << operator. This should display the string (only! No `endl`, no other output messages, etc.).
- an overloaded = operator where the left operand is a String object and the right operand is a literal string (i.e., a string of characters within quotes) **or** another String object. Note that you will not have to actually write this method if your copy constructor is correct.
- an overloaded `+=` operator where the left and right operands are both instances of String objects or the right operand is a string constant.
- an overloaded $[i]$ operator that returns the $i^{th} + 1$ character in the string.
- a method called `length()` that returns the length of the string.

In addition, you need:

- a header filed called *lastname*`_P1.h` that includes all of the class definitions and any include files your library needs.
- a *lastname*`_P1.cpp` implementation file that contains all of your methods defined in the header file.
- a *lastname*`_P1main.cpp` driver file that contains your `main()`.

So what's the catch? You can **not use the C++ `string` type or any of the C++ string operators** in your implementation. That is, you can only use the C++ char string functions available in `string.h`, such as `strcpy()`, just to name one, or do all of the copying and/or concatenation yourself using loops. You therefore have to store all of your Strings as character arrays. Remember, the point is that **you** are writing the string functions for the language; assume they don't yet exist, so therefore you can't use them!

Notes

Remember to comment your code. The introductory comment should appear at the top of your .h file, The description should include all of the operators your program supports. The input/output comments should describe your own `main()` and how it uses the available operators.

As usual, all functions/methods should be commented, including the parameters.

Be sure to name your files as specified.

You will have at least three files to submit (header, implementation, and main application). Zip these together using the usual naming conventions with `.zip` as the file extension. For example, my project would be `mgousieP6.zip`. Submit the file in Canvas, as usual.

In some ways I agree and in some I don't disagree.
– Wheaton Professor Laura Muller at AAC&U conference, July 9, 2009