Project 5 Documentation

Purpose of the Program

==================

This is a program that is meant to test our ability to create and use dynamic memory as well as review our knowledge about how to manipulate pointers, classes, and the iostream library. The core objective of the project involved creating our own unique String class complete with member variables and functions, with both pointer-based and bracket notation used for array manipulation. The source code consisted of three files, two of which formed a header-source code library pair that contained all the details for implementing the functions of the String class. The final source code file served as a test driver for all of the String class functions. Finally, a Makefile was created to compile the source code files together to form the program.

Program Design

=============

The program works to implement the functions and member variables of a unique String class designated as MyString. The MyString source code consists of two separate files, one of which is a header file called MyString.h while the other file is called MyString.cpp. The MyString.h header file stores all specifications that are required for the MyString, mostly being function prototypes and initialized member variables. The header file also includes the function prototypes for the C-string helper functions of strcpy(), strcmp(), and strlen(), which assist in copying strings, comparing strings, and calculating string lengths respectively.

The MyString class contains the private data members m\_buffer, a char-type pointer which always points to the dynamically allocated data for a MyString object, and m\_size, an int that denotes the total number of characters currently allocated for m\_buffer. In addition, every time the dynamically allocated memory is changed, the m\_size is initialized and updated. The MyString class also has two private helper methods of buffer\_allocate and buffer\_deallocate. Buffer\_allocate will allocate the required size for a char array as well as point m\_buffer to it; in the case that there is already allocated space for m\_buffer, buffer\_allocate will deallocate it prior to reallocation of the new memory. Buffer\_deallocate does the opposite, responsible for deallocating the dynamic memory pointed to by m\_buffer.

Finally, the MyString class has the following public member functions: default constructor, parameterized constructor, copy constructor, destructor, size(), length(), c\_str(), the operator==, operator=, operator+, and operator[]. The default constructor will instantiate new MyString objects with no valid data, while the parameterized constructor instantiates new MyString objects that are initialized to hold a copy of the C-string passed as a parameter. The MyString copy constructor will instantiate a new object that acts as a separate copy of the data of the MyString object being copied, while the destructor destroys the instance of a MyString object. Size() will return the size of the currently allocated char buffer, including the null-terminating character, while length() returns the size of the string without counting the null-terminating character; c\_str() will return a pointer to a char array which represents the C-string equivalent of the calling MyString object’s data. The operator== will check if the calling object represents the same string as another MyString object or not, returning true or false respectively, while the operator= assigns a new value to the calling object’s string data based on the other\_myStr object passed as a parameter. The plus operator assigns a new value to the calling object’s string data after appending the other\_myStr object’s data to the original calling object’s data similar to string concatenation. The bracket operator will allow by-reference accessing of a specific character at the index passed in within the allocated m\_buffer array. Finally, MyString has a friend function operator<< that outputs the data for a MyString object to the file or terminal depending on the type of output stream variable passed in as its parameter.

The MyString.cpp file is solely responsible for the implementation of the MyString class member functions that are first specified in the MyString.h header file, responsible for the actual mechanics of how all the functions will carry out in the program itself. In addition to these source code files, the proj5.cpp source code file acts as a test driver that demonstrates the functionalities of the MyString class. The test driver does this by using a variety of strings and calls to each of the public member functions in the MyString class one-by-one in code sections tagged by the appropriate number corresponding to each function.

Problems of Program

================

During the process of coding for the program, there were some issues that turned up concerning exact details of implementation for the overloaded plus operator (operator+) of the MyString class that had been responsible for the concatenation of the C-string representations of two MyString objects. In addition to this, there had been some minor issues regarding the assignment of data types for parameters with a const char designation in the C-string helper functions, most notably in the strlen() and strcpy() functions. Finally, there was an issue with the inclusion of the iostream library into the source code files so that the program files could make use of the ostream variable. Finally, there was some confusion at first about the specific commands in the Makefile for the program files. However, with minor adjustments to the code, all these issues were resolved in an efficient manner, allowing the program to compile and run properly.

Changes to Program

================

I would not have made any changes to the program and its functionalities since I believe that I made the program robust enough to implement the functions of my MyString class properly in a variety of possible conditions and input situations as evidenced by the success of the test driver. The separation of function specifications and implementations between the MyString.h header file and MyString.cpp source code file certainly made the program files more streamlined and defined, making the program more manageable in terms of debugging and coding, which makes it a feature of the program I would definitely not change. If there was anything, I would have liked to separate the includes for the three libraries that were used into separate source code files, most notably in the header file and MyString.cpp file if possible, since I do not prefer the presence of redundancy of the includes in all the source code files, even the test driver, that is needed to allow the source code to properly compile with the Makefile. Other than that, I am more than happy with the resulting program that came about from my efforts.