**Due: 7/12 (11:59PM)**

**You are to code the following three source files in C++:**

- **MyString.h**
  - In this file, you declare a class named *MyString*.
  - This class contains 3 **private** attributes, *str*, *size*, and *strCount*.
  - *str* is a `char*` that points to a dynamic array of characters.
  - *size* is an integer that contains the current size of the string.
  - *strCount* is a `static` member variable that keeps track of the number of *MyString* objects that currently exist in memory. Its value is initially 0 before any *MyString* object is created.
  - Define public getters and setters for the 2 instance variables, as well as a getter for the static variable.
- **MyString.cpp**
  - In this file, you provide definitions for 3 constructors and a destructor for *MyString*.
  - The first constructor takes a `string` parameter, then creates a dynamic char array in the heap and copies the input string content into the array. Note that the size of the dynamic array should match the length of the input string. Also, *strCount* should be incremented by 1 since a new string has been created.
  - The second constructor is a default constructor that takes no parameters. Here, the 2 instance attributes are initialized as *""* (empty string) and 0, respectively. Again, *strCount* should be incremented by 1.
  - The third constructor is a *copy constructor* that allows the following statement:

    ```
    MyString y = x; // at declaration
    ```

    so that the *MyString* y is initialized as the same string as x. However, make sure their respective arrays are separately created in the heap (even though their contents are the same). Again, *strCount* should be incremented by 1.
  - Also define a destructor function for the class that releases the dynamic array. The destructor must also decrement *strCount* by 1.
  - Add a *display* function that displays the current contents of the string on the screen.
  - Overload operator = to allow assignment (away from declaration statement) as follows:

    ```
    x = y; // where x and y are both MyString objects
    ```

Note that this assignment must perform *deep copy*, which means the entire source dynamic array must be copied to the target dynamic array, not just the pointer.

- Overload operator += to allow a statement like:

```
x += y; // where x and y are both MyString objects
```

This operator must append *MyString* y at the end of x, so that x becomes a merged string. Note that the previous dynamic array of x must be destroyed and a new dynamic array must be created with an updated *size*.

- **hw4.cpp**
  - In this file, you define main function that tests *MyString* class. You must create at least 3 *MyString* objects.
  - Show that the first constructor properly works via a statement like: `MyString s1(``abc'');`
  - Show that the default constructor properly works via a statement like: `MyString s1;`
  - Show that the copy constructor properly works via a statement like: `MyString s2 = s1;`
  - Show that the overloaded operator = properly works via a statement like: `s2 = s1;`
  - Show that the overloaded operator += properly works via a statement like: `s1 += s1;`
  - Show that the *strCount* is properly updated via a statement like: `cout << s1.getStrCount();`
  - Show that the display function properly works via a statement like: `s1.display();`
  - Note that the display function must be called every time a *MyString* object is created or updated. Otherwise the respective functionality will not be properly demonstrated and *could lead to loss of points*.

**What to submit:**

- All source files (.cpp, .h) needed for compilation

**How to submit:**

- Use Canvas Assignment Submission menu to submit the assignment electronically at Canvas.
- Make sure to zip all your files into `hw4.zip`, then submit your `hw4.zip` as a single file.

**Policy**

- Make sure all your C++ programs properly compile and run on Eclipse C++.
- Projects will be graded 20% on style/standards and 80% on proper execution. Make sure to follow the *Coding Standards* posted on the course webpage. In particular, use proper indentation on each line of your code.
- At the beginning of each file (.cpp, .h), provide comments specifying the author, date, and a brief description of the file.
- Each source file (.cpp, .h) must contain enough comments here and there to make it easy to follow your code. Insufficient comments could lead to loss of points.
- Non-compilable program will get almost no credit (e.g., executable code not produced due to compile errors).
- Non-working program will get almost no credit (e.g., the executable is terminated immaturely due to run-time errors).
- Copying other's code is strictly prohibited. If identical (or nearly identical) submissions are found among students, every student involved will get automatic zero for the assignment. The same goes for copying existing source code from the internet.