1 Problem Description

This lab is the continuation of Lab 4 to get us started in implementing mystring class that intends to mimic the string class in C++ standard library. We have been asked to gather the member function prototype information in Lab 4. To make sure that everyone have the same code base to work with, the declaration of mystring class is provided to us as well as the code for one or two constructors as well as other member functions and operators to get us started (see mystring.cpp file).

Please compare the member function prototypes you gathered with those provided. It is very important to understand the prototypes before we implement the functions.

We will experiment with the default copy constructor and default assignment operator “=” provided by the language. In order words, if we do not include both prototypes in the declaration, the compiler or language will add both the declaration and definition for us for these functions so they can be used even if we do write a single line of code. The default uses so called shallow copy, that is to copy every fields. The default is fine if no field is of pointer type. However, it is not the case for mystring class (see declaration)

In this lab we will see what would happen to mystring class objects if we use the defaults. We need to first see the problem and to understand why the problem occurs. After that, we will implement our copy constructor that uses deep copy (the default uses shallow copy, that is just copy the point value and not the value pointed to by the pointer). The assignment operator implementation as well as the implementation of the other member functions in the declaration will be done in our Homework 3.

2 Purpose

Understand string class and how to implement such a complicated class. Understand memory management. Review pointers, array, and operator overloading.

3 Design

The basic code base for main.cpp, mystring.cpp, and mystring.h are provided for us. By comment out the line

```cpp
#define string mystring
```
the main program uses the C++ string. Uncomment the line, the main program will use mystring class for string class as if we replace the word string by mystring throughout the main program.

We will add a few lines of code to the main program to test the default copy constructor. After that we will implement the copy constructor of our own.

Again we will call function check (developed in Lab 4 by us, which is provided in main.cpp) to see the value of string or mystring object, after each operation.

In the implementation section more details are provided regarding what the main program does and how to implement copy constructor.

4 Implementation

4.1 Testing using C++ string for copy constructor

1. output This is Lab 5 as the first line of the program

2. Use a constructor with value "Hello, World!" to define a string variable called s1 (string s1("Hello, World!"));

3. Use a constructor with value "s1" to define a string variable called s1name,

4. invoke check: check(s1,s1name);

5. invoke check: check(s1,s1name);

6. output Lab 5 ends as the last line of the program

We should get the following output:

This is Lab 5
checking s1
s1 contains Hello, World!
s1 capacity() is 13
s1 length() is 13
s1 size() is 13
s1 max_size() is 1073741820

checking s1
s1 contains Hello, World!
s1 capacity() is 13
s1 length() is 13
s1 size() is 13
s1 max_size() is 1073741820

Lab 5 ends
Press [Enter] to close the terminal ...
4.2 Testing using mystring for default copy constructor

1. Uncomment the line `#define string mystring` in main.cpp
2. Compile and run the same program

You should get a run time error such as `core dump`, `segmentation fault`, or other strange error. Locate the line of code that caused the error as accurately as you can. Please try to explain why we get such a nasty error in your lab report. For now, move on to the next task.

4.3 Implement copy constructor

The copy constructor takes one argument, which is an object of the same type. We will create an object in the constructor based on the argument object. The main concept here is that instead of copying the pointer value stored in `ptr_buffer` field of the argument object, we will copy the content pointed to by the pointer of the argument object. In order to do that, we need to allocate enough space and store its address in our `ptr_buffer` field and then copy the content. It would be helpful to study the implementation of the constructor that takes in a c-style string in mystring.cpp file.

1. Uncomment the copy constructor prototype line in mystring.h
2. Uncomment the definition of copy constructor in mystring.cpp
3. Set member field `len` to the return value of the member function `length()` of the argument object
4. Use `new` to allocate `len+1` char and store the address in member field `ptr_buffer`
5. Set member field `buf_size` to `len+1`
6. Copy the content pointed to by `orig.prt_buffer` to the location pointed to by `ptr_buffer`.
7. Compile and run the same program

We should get the following:

```
This is Lab 5
checking s1
s1 contains Hello, World!
s1 capacity() is 14
s1 length() is 13
s1 size() is 13
s1 max_size() is 1073741820
```

checking s1
s1 contains Hello, World!
s1 capacity() is 14
s1 length() is 13
s1 size() is 13
s1 max_size() is 1073741820

Lab 5 ends
Press [Enter] to close the terminal ...

There are a few minor differences between the two outputs. Please explain them in your lab report.

5 Test and evaluation

Testing have been done during the implementation. You are welcome to try more tests.

6 Report and documentation

A short report about things observed and things learned and understood about the default copy constructor and assignment operator. Any observation about the given mystring.cpp and mystring.h code are welcome (the code has examples and shows the design idea).

7 Lab submission

Get instructions from the Lab instructor.