1 Problem Description

In our earlier labs, we have studied and used functions, arrays, pointers, c-style strings, vectors, c++ strings, and classes. We have used loops as well as other control statements to implement our algorithms. We know the importance of type and the role it plays in programming languages.

In this lab and lab five, as well as homework three, we will study the standard library string class (or type) in more detail and implement a mystring class (or type) of our own that mimics the behavior of string (this should remind you of our Lab two and homework one where we implemented library functions).

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 main program of this lab simply uses many member functions of class string. In the implementation section more details are provided regarding what the main program does and the introduction of a helper function to see the values after each operation.

During the development of the main program, we will gather the prototype of the member functions used and make them the member functions of our mystring class. The following shows a partially filled mystring class declaration. We will implement all the member functions we have used in the main program in later lab and assignment.

Prototype of member functions and what they suppose to do:

```cpp
class mystring {
public:
    // types with scope in this class
    typedef unsigned int size_type;
    typedef char * iterator;
    typedef const char * const_iterator;

    // constructor
    // constructor prototypes you have used listed below
    .
    .
```
// copy constructor
mystring(const mystring& orig);

// destructor
virtual `mystring();

// iterators
iterator begin();
iterator end();

//=== memory related ===
// size(), length(), reserve(...), capacity(), empty(), etc
// prototypes you have used listed below
.
.
.

//=== overloading operators ===

// assignment operator (=) prototypes you have used listed below
.
.
.

// array notation ([]) prototypes you have used listed below
.
.
.

// append (+=) prototypes you have used listed below
.
.
.

//=== methods that modify the string ===
// append, insert, replace prototypes you have used listed below
.
.
.

2
our implementation uses the following
private:
  // pointer to the memory location where string is stored as a c-style
  // string
  char * ptr_buffer;
  // the size of the memory in terms of bytes or characters
  size_type buf_size;
  // number of characters currently in the memory not including the
  // terminating null character
  size_type len;
};

4 Implementation

The file that contains the main program is called main.cpp. The actions of the
main program are listed as follows. To help us check the "state" of a string
object, we will implement a function with the following prototype:

void check (string s, string name);

The first argument is the string object to be check, and the second argument
is the name of the variable for the object. For example, we have

string s1;
s1 ="Hello, World!";

we can call

check(s1, "s1");

we could get the following printed on the screen:

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

First implement the check function as outlined above. The following lists the
actions of main program. After each action, the function check should
be called to check the string value.
1. Use the default constructor to define two string variables call s0, s2
2. Use member function empty to test if s0 is empty and output the result.
3. Use another constructor that takes the value "Hello, World!" to define a string variable called s1.
4. use assignment to assign to s2 the value of s1.
5. use push_back to add '!' to the end of s2.
6. use += so s1 is the concatenation of s1 and s2.
7. use insert to insert a space in s1 between the first Hello, world! and the second Hello, world!
8. use assignment and + to assign s0 the concatenation of s1, " ", and s2.
9. use reserve to reserve 50 characters for s0
10. introduce a variable c of type char and assign the value of s0[12] to it. Output the value of c. (remember to check s0 afterward).
11. use append to append s2 to s2.
12. use replace to replace the first "!!" in s2 with =.
13. introduce a variable p of type pointer to constant c-style string (i.e. const char * p;) to capture the return value of c_str of s2. output the value of p.
14. use == to check if p and s2 are the same and output the answer.
15. s0[12]=s2[12];
16. use clear on s0
17. use clear on s2
18. output value of p

5 Test and evaluation

Test the main program. Carefully observe the output and the corresponding code in the main program to get a better idea of what each member function does.

6 Report and documentation

A short report about things observed and things learned and understood. Properly document and indent the source code. Document the mystring class member functions.
7 Lab submission

Get instructions from the Lab instructor.