The Programming Assignment Report Instructions
CSCE 221

1. The description of an assignment problem

Since a **string** is not included in the C++ language, this assignment was focused on the elementary design, implementation and test of the string class called **my_string**. The purpose of this class is to show how the STL string class is implemented and it provides the overview of the basic C++ concepts that were covered in CSCE 121.

1. The description of data structures and algorithms used to solve the problem.

   (a) Provide definitions of data structures by using Abstract Data Types (ADTs)

   The data structure that was used was a static **array** that terminated in the character '\0'. No dynamic ADTs were used. The array was also included with the **int** values of **sz** and **cap** which held the number of elements of the array and the capacity of the array.

   (b) Write about the ADTs implementation in C++.

   The implementation of the Data Structure was called every time we initialized a my_string variable in the main.cpp file. Depending on the type of arguments, there were different constructors that initialized the data structure. For the basic, 'empty' object, the default pointer was pointing to a new char **array** of size cap, which was set to a default **capacity** of 2 and a **sz** of 0. When sending in an int s in the argument, the constructor would first call a IncreaseCapacity() function that would check whether the **capacity** of the **array** was larger than the size, if no it would double the size of the capacity to allow the new elements to be written in the array. The third constructor was initialized by sending in a string. Again, the IncreaseCapacity() function was called if the number of elements were bigger than the **capacity** of the **array**. After the increase, the string would be assigned to a new array that was accessed though a pointer. We also have one more constructor that will copy the members from another data type to initialize it to its own.

   (c) Describe algorithms used to solve the problem.

   The algorithm that were used were various. One algorithm was to increase the size of the capacity by multiplying it by 2. This was done by reading in the the total size of the elements and determining if the capacity of the original array is large enough to hold all the elements. If not, then a temporary array was created to hold the array then the old array pointer was redirected to point at the temporary array. Another algorithm implemented was to count the number of elements in the incoming string. This was executed with a while statement that would count until the character '\0' was found. Other algorithms involved using for loops to assigned the elements from arrays to new arrays or to append strings to existing arrays.

   (d) Analyze the algorithms according to assignment requirements.

   The algorithm works very well when implemented in the main.cpp file. The constructors initialize the data objects for different input cases. The overloaded operators (+=,=) works very well by manipulating the character arrays that are available.

2. A C++ organization and implementation of the problem solution

   (a) Provide a list and description of classes or interfaces used by a program such as classes used to implement the data structures or exceptions.

       This imitates the string library that is not native to C++. We can implement these functions anywhere that requires a user to input string data in any program.

   (b) Include in the report the class declarations from a header file (.h) and their implementation from a source file (.cpp).

       These files are included in the .tar file.

   (c) Provide features of the C++ programming paradigms like Inheritance or Polymorphism in case of object oriented programming, or Templates in the case of generic programming used in your implementation.

       For this assignment, there was not an instance of Polymorphism except for the inclusion of iostream and stdexcept in the header file.

3. A user guide description how to navigate your program with the instructions how to:

   (a) compile the program: specify the directory and file names, etc.

       To compile the program, it is important to run the make file which will make the header file and the .cpp files to output a single my_string.o file. To run the make file, simply type “make all”. To run the output file, simply type ./my_string in the terminal. This will execute the desired code.

   (b) run the program: specify the name of an executable file.

       The name of the output file to run is my_string. To run simply type ./my_string in the terminal.

4. Specifications and description of input and output formats and files

   (a) The type of files: keyboard, text files, etc (if applicable).

   The user must use a keyboard to input text into a character array which uses the istream overloaded operator to read into the data structure.

   (b) A file input format: when a program requires a sequence of input items, specify the number of items per line or a line termination. Provide a sample of a required input format.

   The input that the user will type will read a string until white space, tab or new line character is found. The string before the character will be read into the data structure.

   (c) Discuss possible cases when your program could crash because of incorrect input (a wrong file name, strings instead of a number, or such cases when the program expects 10 items to read and it finds only 9.)

   The program could crash when the user inputs a string that is longer than 2^64 characters due to the IncreaseCapacity() function that depends on the doubling of an int. The input would be too many characters for the possible number of elements that the capacity int could hold.

5. Provide types of exceptions and their purpose in your program.

   (a) logical exceptions (such as deletion of an item from an empty container, etc.).

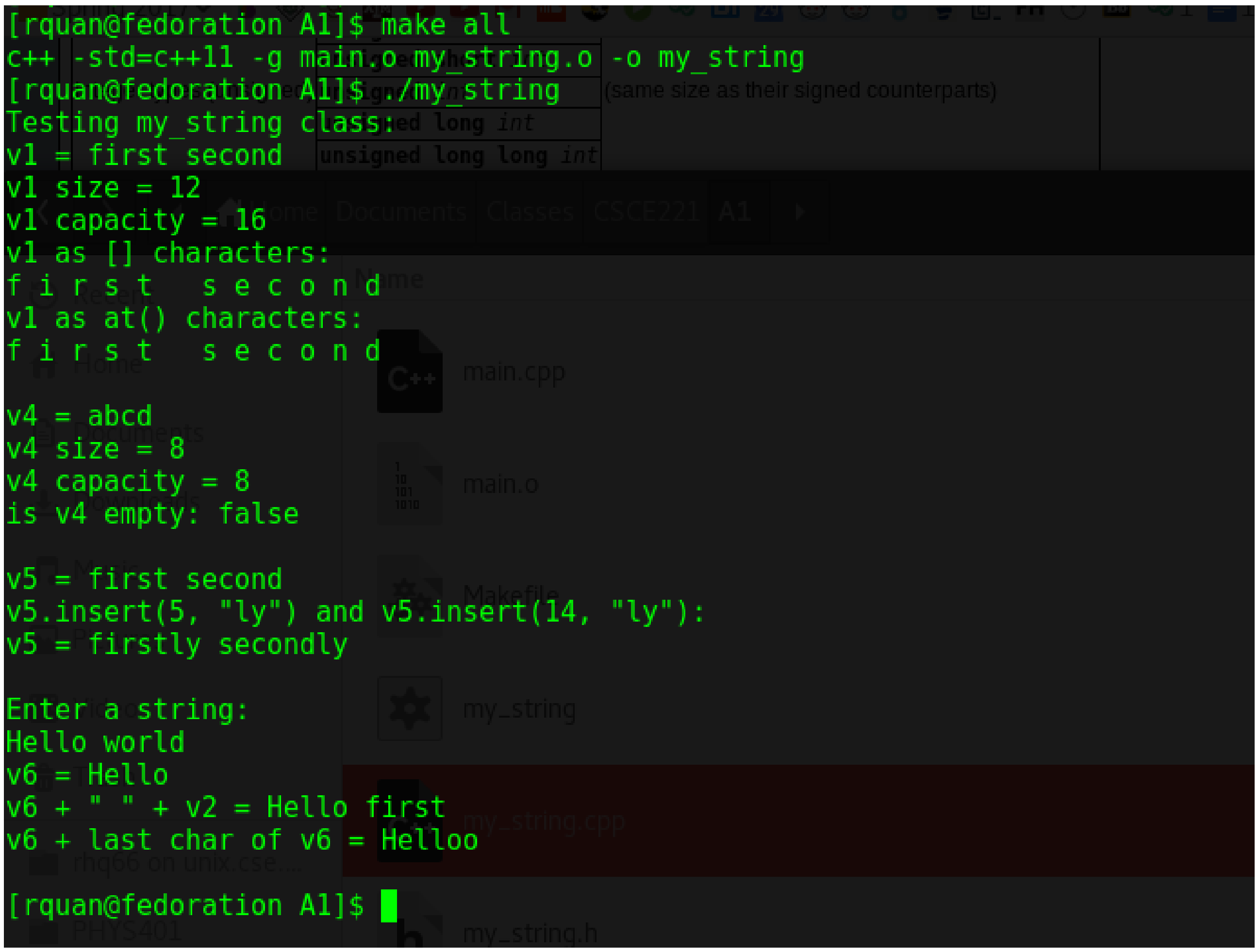
   The exception used is an out_of_range Error when the at(int i) function is called. If the input i is negative or bigger than the size of the array, an exception will be thrown.

   (b) runtime exception (such as division by 0, etc.)

   If there is an error in the constructor and the capacity is set to 0, then the IncreaseCapcity() function will crash due to the capacity being multiplied by 2 and thus equaling 0 again. No string will be able to input into the function.

(c) Test your program for correctness using valid, invalid, and random inputs (e.g., insertion of an item at the beginning, at the end, or at a random place into a sorted vector). Include evidence of your testing, such as an output file or screen shots with an input and the corresponding output

By testing the program with the included main.cpp, every input was successful. Here are some screenshots of the output.



```
[rquan@fedoration A1]$ make all
c++ -std=c++11 -g main.o my_string.o -o my_string
[rquan@fedoration A1]$ ./my_string
Testing my_string class:
v1 = first second
v1 size = 12
v1 capacity = 16
v1 as [] characters:
f i r s t   s e c o n d
v1 as at() characters:
f i r s t   s e c o n d

v4 = abcd
v4 size = 8
v4 capacity = 8
is v4 empty: false

v5 = first second
v5.insert(5, "ly") and v5.insert(14, "ly"):
v5 = firstly secondly

Enter a string:
Hello world
v6 = Hello
v6 + " " + v2 = Hello first
v6 + last char of v6 = Helloo

[rquan@fedoration A1]$ 
```