CSCE 221 Cover Page
Homework Assignment #01

First Name: Joseph Last Name: Martinsen UIN: 323009961

User Name: Josephmart E-mail address: josephmart@tamu.edu

Please list all sources in the table below including web pages which you used to solve or implement the current homework. If you fail to cite sources you can get a lower number of points or even zero, read more in the Aggie Honor System Office http://aggiehonor.tamu.edu/

Type of sources	Help with Pointers	Help with input operator
People	Jonathan Arauco	Tristan Partin
Web pages (provide URL)		
Printed material		
Other Sources		

I certify that I have listed all the sources that I used to develop the solutions/code to the submitted work.

"*On my honor as an Aggie, I have neither given nor received any unauthorized help on this academic work.*"

Your Name (signature) Joseph Martinsen Date 02/01/2017

The Programming Assignment Report Instructions
CSCE 221

1. The description of an assignment problem.

   The purpose of this program assignment was to introduce to the student to the idea of data structures by giving them hands on experience by creating a basic dynamic string array called my_string

2. 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 my_string used a list data structure to store the data. The container holds characters that make up the my_string.

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

   A pointer to the my_string is allocated on the stack. The data for the my_string is then allocated on the heap. If there is not enough room to add more characters, a new my_string object is created allowing with more memory allocation in order to hold the data.

   (c) Describe algorithms used to solve the problem.

   There were two crucial algorithms implemented in the my_string class. One was for resizing when needing to increase memory allocation. The other was for inserting a character in a certain position.

   The first algorithm resize functions as follows. First compare if size of component being added is greater than the current capacity of the my_string. If it does, create a temp my_string to hold the current my_string. Change the capacity to be $2(size + 1)$. Next create a new pointer with size being the new capacity. Finally, Iterate through adding the temp values back to the newly made my_string function.
   The second algorithm insert is as follows. First compare if the passed in index is greater than 0 and less than my_string size. If not, throw out of range exception. If the condition is met, do as follows. Create a temp my_string that holds all the values of the current my_string after the passed in index. The current my_string size should now be the current my_string size plus the size of the passed in my_string. Next call the resize and pass in the current my_string size. Next, add the new my_string to the current my_string starting at the passed in index. Finally, add the temp my_string back to the current my_string

   (d) Analyze the algorithms according to assignment requirements.

   - ☑ size() returns the number of characters (length) of s.
   - ☑ capacity() returns the length in bytes of the allocated memory. It cannot be smaller than size() for the same string.
   - ☑ empty() returns true if the string s is empty and false otherwise.
   - ☑ operator[](i) returns the character at index i of s, without performing arrays bounds checking

- ☑ at(i) returns the character at index i of s, with performing arrays bounds checking.
  An out_of_range exception is thrown if i is not in range of the string size (between 0 and size()-1).
- ☑ operator+=(q) appends the string q to s.
- ☑ operator+=(c) appends the character c to s.
- ☑ insert(i, s) inserts the string s before the position i in s and returns a reference to the resulting string. This function is optional for extra credit.
- ☑ default constructor creates an empty string without any memory allocation.
- ☑ constructor with an int argument n creates an empty string with allocated memory of size n bytes.
- ☑ constructor with a C-string creates a string with the content taken from the C-string.
- ☑ copy constructor makes a copy of the argument string.
- ☑ destructor deallocates allocated memory and makes an empty string.
- ☑ copy assignment assigns a string to another string (s = q).

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

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

   Two includes that were used in the my_string files were iostream and stdexcept

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

   Code Block 1: Declarations from my_string.h

```
#include <iostream>
#include <stdexcept>
```

   Code Block 2: Usage of iostream

```
        if (first) first = false;
        q.resize(q.sz + 1);
        q[q.sz] = c;
        q.sz++;
    }
    return is;
```

   Code Block 3: Usage of stdexcept

```
throw std::out_of_range("Out of range");
```

   (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.

   The my_string class utilized OOP in order to function.

4. 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.

   In order to compile the program, navigate to directory **221-A1-code** from the terminal. Within that directory, type the command **make**.

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

   From the **221-A1-code** directory, from terminal, enter **./my_string**

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

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

   N/A

   (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.

   N/A

   (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.)

   I do not believe that this program will crash but an exception will be thrown if a my_string object named A is created and then $A[i]$ is implemented where $i > A$.capacity() && $i < 0$

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

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

   N/A

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

   A runtime exception will be thrown if a my_string object named A is created and then $A[i]$ is implemented where $i > A$.capacity() && $i < 0$

7. 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.

```
[joseph@josephsolus] [~/Documents/CSCE221/A1/221-A1-code] [master *]
$ ./my_string
Testing my_string class:
v1 = first second
v1 size = 12
v1 capacity = 26
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 = 4
v4 capacity = 10
is v4 empty: false

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

Enter a string:
Linux
v6 = Linux
v6 + " " + v2 = Linux first
v6 + last char of v6 = Linuxx
```

```
[joseph@josephsolus] [~/Documents/CSCE221/A1/221-A1-code] [master *]
$ ./my_string
Testing my_string class:
Calling v1 out of range with []
Segmentation fault (core dumped)
```

```
[joseph@josephsolus] [~/Documents/CSCE221/A1/221-A1-code] [master *]
$ ./my_string
Testing my_string class:
Calling v1 out of range with .at()
Out of range: Out of range
```