CSCE 221 Cover Page
Programming Assignment #1
Due July 11 by midnight to eCampus

First Name Laura Last Name Austin UIN 524006473

User Name laustin254 E-mail address laustin254@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: Aggie Honor System Office

Type of sources		
People		
Web pages (provide URL)	http://www.cplusplus.com/reference/vector/vector/assign/ http://www.learncpp.com/cpp-tutorial/712-handling-errors-assert-cerr-exit-and-exceptions/	htt
Printed material	"*Discrete Structures*" "*Introduction to Program Design and Concepts*"	
Other Sources		

I certify that I have listed all the sources that I used to develop the solutions/codes 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 Laura Austin Date 7/11/2017

Programming Assignment 1 (130 points)

In the first phase of the assignment, implement in C++ a class `My_vec` that can hold data of character type (`char`). The description of the functions for data manipulation is provided in the first set of the lecture notes, see the slide 8. In the second phase, write a generic version of the class `My_vec` that can handle any type of data.

1. Description of the Assignment Problem:

The assignment was to create and implement a vector that holds the type char. This vector imitates the classic vector class from stdlibfacilities.h. The objective was to get the vector to insert, remove, evaluate size, call specific indices in the vector, replace at specific indices, as well as finding the maximum element in the vector and sorting the vector.

The other parts of this assignment had to do with templating this vector class My_vec, which implies that any user can create a vector of any type, whereas the first portion of the assignment was specific to the type char.

2. The Description of data structures and algorithms used to solve the problem.

(a) The data structures used were arrays, and for part 1, they specifically stored the type char. However, for part 2, the arrays specifically stored any user type when the class was called in the main.cpp file. The operations that support the data for part 1 and part 2 were insert, delete, replace (update), search (by calling a specific index of the array), replace, sort, and finding the maximum element index.

(b) Abstract Data Structure's implementation in C++ is recognized as a class, according to slide 6 out of 27 on lecture notes "Data Structures & Algorithms." Furthermore, the implementations are containers, which include the sequences, under which vector is classified.

(c) The algorithms I used to solve the problem included referencing insertion, as well as sort. In insert, there had to obviously be error checking which would end the algorithm, but if there was not an error thrown, then the value of size was checked against the value of capacity. If size is greater than or equal to capacity, then there must be more memory allocated for the vector (because the vector is full). Memory is doubled. Then contents of the vector must be copied, and the new element added. Since the new element is added, there must be an entire shift of the vector, as well as an increase of size. For sorting, I used the function created that found the maximum index. The idea was to check the first 2 elements of the vector for the max index, and see whichever one had the maximum index, then the elements essentially swap according to the sort being from low to high or high to low, and iterate this process the same number of size times.

(d) The algorithm sort ran an iteration of size - 1 times in best case, whereas the algorithm insertion ran the number of size times at best case.

3. C++ organization and implementation:

(a) The class used was My_vec, and it implemented the data structures mentioned above.

(b) Class declarations were My_vec(); ~My_vec(); My_vec(const My_vec& vec); My_vec& operator=(const My_vec& vec); int get_size() const; int get_capacity() const; char& operator[](int i) const; char& operator[](int i); bool is_empty() const; char& elem_at_rank(int r) const; void insert_at_rank(int r, const char& elem); void replace_at_rank(int r, const char& elem); void remove_at_rank(int r),

There implementation is in the .cpp file that is attached in the tar file on eCampus.

(c) The templated version is also in the tar file on eCampus. Essentially I just replaced every time "char" came up in the .cpp file of My_vec, then put the definitions of the functions in the header file, and classified them under template <typename T> My_vec.

4. How to navigate the program:

(a) compile the program.

the directory will be named Austin-Laura-PA1 which is my first and last name, programming assignment 1. Within that, there are two more tar files, one labeled Austin-Laura-PAp1.tar, containing the first part's files: My_vec.cpp, My_vec.h, main.cpp, and the other labeled Austin-Laura-PAp2.tar, containing the second part's files: TemplatedMy_vec.h, main.cpp.

(b) The executable file will be main.cpp, which you can run by typing ./main, I believe.

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

(a) In this specific program I do not believe there is a text input or text output file. All the output will be written to the command window, and not to a text file.

(b) Since the file must run based off of the .cpp file, all one must do to test the file is compile the main.cpp file, compile the My_vec.cpp, compile the My_vec.h and run the main file, and for part two the only two files that must be compiled are TemplatedMy_vec.h and main.cpp.

(c) The file that will probably abort due to failure in try-catch blocking methods will be the templated version of the assignment. It should compile, but when tested with the main file, it will probably abort. However, if all the try catches were written with "cerrs" (which would keep it from aborting), the .h file would not compile. Therefore I chose to implement the throw method of error checking in order to ensure the file compiles, despite the fact that the main file may abort midway.

6. Types of exceptions and their purpose:

(a) My program is full of many logical exceptions, because working within the data structure array there can be many mishaps regarding memory. Firstly, ranks that are called to that are less than 0 or larger than the size-1 cause an error. Also, if the user attempts to input an element into the 5 rank in a vector that is only of size 3 elements, then the user faces an error on their part. This happens because vectors are contigous data structures.

7. Pictures of the program:

Part 1:

```
[laustin254]@linux2 ~/PA1-17b-eCampus> (23:58:38 07/11/17)
:: ./main
[B]
v Size : 1
[A]
[B]
v Size : 2
RANK OUT OF RANGE
[A]
[B]
v Size : 2
[A]
v Size : 1
E
 v Size : 1
[E]
RANK OUT OF RANGEThis is v1:
[E]

This is v2:
[K]
v2 size: 1
[1]
[2]
[3]
[4]
[5]
v2 Size: 5
find max of v2 is 4
[1]
[5]
[5]
[5]
[5]
[1]
[2]
[3]
[4]
[5]
RANK OUT OF RANGE
```

Part 2:

```
[laustin254]@linux2 ~/PA1-17b-eCampus/PAlpart2> (00:01:45 07/12/17)
:: ./a.out main
[B]
v Size : 1

[A]
[B]
v Size : 2

terminate called after throwing an instance of 'char const*'
Aborted
```

as mentioned earlier, this code compiles however it does abort due to try catch errors.