Lab 2

Move and Copy Semantics

In this workshop, you are to compare move and copy operations on an object that contains a dynamically allocated array of string objects.

**LEARNING OUTCOMES**

Upon successful completion of this workshop, you will have demonstrated the abilities to

* implement copy semantics for a class with a resource
* implement move semantics for a class with a resource
* identify the processing-intensive operations in copy and move assignments
* retrieve data from a text file using an **ifstream** object

**SPECIFICATIONS**

Text Class

Design and code a class named **Text** that manages a dynamically allocated array of **string**s.  Upon instantiation, a **Text** object receives nothing or a reference to an unmodifiable string.  The string holds the name of the text file that contains the records to be stored in an object of this class.  If the file does not exist, the **Text** object assumes a safe empty state.  If the file exists, the one-argument constructor allocates memory for the number of lines contained in the file and copies them into memory.  To review the syntax for reading from a text file using an **ifstream** object see the chapter in your notes entitled [Custom File Operators](https://scs.senecac.on.ca/~oop244/pages/content/files.html).  See also [cplusplus.com](https://www.cplusplus.com/reference/cstdlib/rand/)

Your design also includes the following member functions:

* a copy constructor
* a copy assignment operator
* a move constructor
* a move assignment operator
* a destructor
* a member function named **size\_t size() const** that returns the number of records of text data

Define your class and its implementation in namespace **w2**.  Store your class definition in a header file named **Text.h** and your member function definitions in an implementation file named **Text.cpp**.

Input

A text file named **gutenberg\_shakespeare** is available with this lab. Make sure you include this file as a command line argument.

Output

The output of this program is based on the speed and processing power of the machine that it’s being run on. What you are looking for in the output is the following:

* Size of all the objects should be the same
* Copy operations should be slower than move operations

EXAMPLE

Default Constructor 380 nanoseconds - a.size = 0

Custom Constructor 9982078037 nanoseconds - b.size = 124457

Copy Assignment 422828094 nanoseconds - a.size = 124457

Move Assignment 761 nanoseconds - a.size = 124457

Copy Constructor 399876151 nanoseconds - c.size = 124457

Move Constructor 1141 nanoseconds - d.size = 124457

Destructor 429119369 nanoseconds