**CSC 2110**

**Computer Science I**

**Fall Term 2015**

**Project 02**

**80 points**

**Due 12/07/2014 (12:00 P.M.)**

**The goal of this project is to:**

1. Understand the principle of **abstraction** and **encapsulation** and use them to implement classes.
2. Overload various operators and become aware of their restrictions.
3. Highlights the peculiarities of classes with pointer data members and how to avoid them.
4. Create and work with dynamic two-dimensional arrays.
5. Explore how to sort an array using the bubble sort, selection sort, and insertion sort algorithms
6. Explore how to use recursive functions to implement recursive algorithms
7. Analyze, design, implement, and test a practical real-world application.

**Requirements:**

• Analyze the problem; outline the problem and its solution requirements.

• Design an algorithm to solve the problem.

• Implement the algorithm in C++, and verify that the algorithm works.

• Each class must contain a header file (.h) and an implementation file (.cpp).

**Restrictions:**

You must work individually. Use only material from class or from the text book (chapters 1-16). All code must be the work of the individual. Do not share your code or copy from external resources.

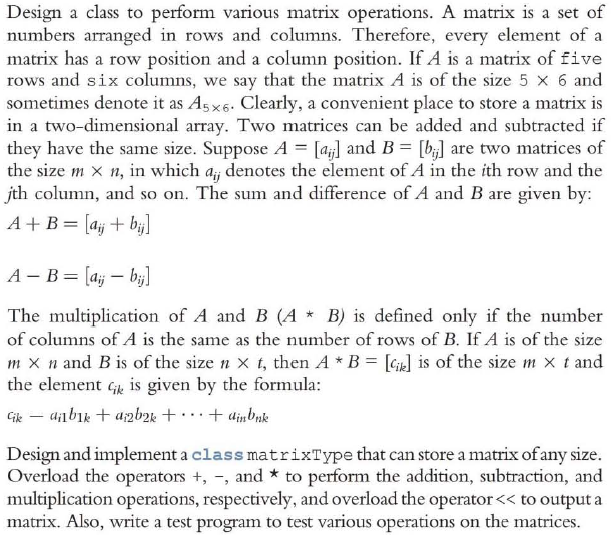
**Grading:**

The grade of each program will be based on the creation of a program that works correctly, up to some details (40%), clear problem analysis and algorithm design (10%), the appropriate use of classes (20%), the production of clear output, with readable formatting and without unnecessary repetition (15%), composition of informative comments (10%), and testing the program with different inputs (05%). Programs must compile.

**Submission**

* Create the application program from scratch using visual studio C++ 2013.
* Type your analysis and algorithm for each problem in this file.
* Solve each problem and include the source file of each problem and this file in a folder. Name the folder (CSC2110\_Project\_02). Compressed the folder and upload it to the blackboard using the appropriate folder by the due date. No email or hard copy is accepted.

**Problem 01 (40 points)**



Remember, classes with pointer member variables must:

* Explicitly overload the assignment operator
* Include the copy constructor
* Include the destructor

**Analysis:**

In this exercise we will have to create two 2d dynamic arrays to perform and act like matricies. And for these matricies we will have to be able to add, subtract and multiply them with eachother. We will have to make sure that two matricies are able to be multiplied so we will have to check the columns of the first vs the rows of the second.

**Algorithm:**

First we will have to make the header file with function for defaut constructor and filling the matrix. On top of that we will have to create an input function to fill the matrix with values. I will need to have a comparison function to make sure that the two matricies are compatible with eachother for multiplication. In the Matrix class there will be a double pointer variable for the 2d array and the row and column variables. The last functions needed will be the overload operators for addition, subtraction, multiplication and output. For addition and subtraction its as easy as using the like terms but in the multiplication algorithm you need to make sure that you are doing the proper multiplication method for matricies.

**Problem 02 (40 points)**

In C++, the largest int value is 2147483647. So, an integer larger than this cannot be stored and processed as an integer. Similarly, if the sum or product of two positive integers is greater than 2147483647, the result will be incorrect. One way to store and manipulate large integers is to store each individual digit of the number in an array.

Design a class named largeIntegers such that an object of this class can store an integer of **any number of digits.** Add operations to add, subtract, multiply, and compare integers stored in two objects. Also add constructors to properly initialize objects and functions to set, retrieve, and print the values of objects.

**Analysis:**

For this problem we will have to create large array objects for the large integers and have each index in the array represent a digit in the number. This is so we can use numbers larger than 2147483647. We will also have to be able to write functions that will allow us to add subtract multiply and compare the numbers in the arrays.

**Algorithm:**

First will be the constructor for the arrays, this will initialize all the values in the array to 0, and the length of the array. Then the input and output function that will simply take a string that has numbers and go through and input the individual characters of the string into the entries in the array. Then go through the array and output each entry. In the overload function we have to make sure that the parameters are the right side array and that they are const so they aren’t altered through the execution of the code. The private variables are the array and the length of the array.

**Extra Credit 01 (20 points)**

Write a program that creates three identical arrays, listl, list2 , and list3 of 5000 elements. The program then sorts listl using bubble sort, list2 using selection sort, and list3 using insertion sort and outputs the number of comparisons and item assignments made by each sorting algorithm.

**Analysis:**

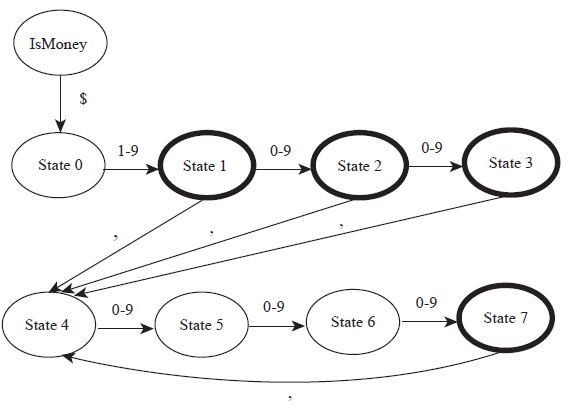
For this program we will need 3 lists with 5000 entries where each entry is a random number. We will then use these lists to run the three sorts, selection, insertion and bubble. And while these run count how many comparisons and replacements are made and output the counters.

**Analysis:**

First will be the creation of the list with simple array declarations with 5000 in the brackets. Then we will create and initialize the counters to 0. Afterward using a random number generator will fill each entry in each list with a random number. And then define three function, one for bubble, insertion and selection sort each. In each function evertime there is a comparison the initial counter will increment and everytime two numbers switch places the assignment counter will increment. After properly defining the function we will call them in the main function using the lists and output the counter values in the results.

**Extra Credit 02 (20 points)**

The diagram below is an example of a *deterministic finite state automaton*, or DFA. This particular DFA describes an algorithm to determine if a sequence of characters is a properly formatted monetary amount with commas. For example, “$1,000” and “$25” and “$551,323,991,391” are properly formatted but “1,000” (no initial $) and “$1000” (missing comma) and “$5424,132” (missing comma) are not.



**Analysis:**

For this problem we will need to create a state machine that at each part of a monetary value will compare it to standard monetary format whether a number, comma or dollar sign should be in that spot. If it should be there the program will move on if not it will stop and say the format is invalid.

**Algorithm:**

First will be the dynamic character array that will take the money value we enter and turn each item in the value into entries in the array(i.e. $, ‘,’ and a number). Next will be the enumerator that will represent the different states in our state machine. And setting the default state to ismoney and checking the amount starts with the appropriate ‘$’. After that we will have to loop through the array and checking for the appropriate values depending on the state the enumerator is in. If at anypoint before reaching the end of the value the state machine finds an inappropriate value it will return an error value.