**Logo

Description automatically generated San Francisco Bay University**

**CS360 - Programming in C and C++**

**Homework Assignment #1**

**Due day: 5/31/2022**

**Instruction:**

1. **Push the answer sheets/source code to Github**
2. **Please follow the code style rule like programs on handout.**
3. **Overdue homework assignment submission can’t be accepted.**

**4. Take academic honesty and integrity seriously (Zero Tolerance of Cheating & Plagiarism)**

1. This program design is to calculate complex number. Complex values are denoted by a parenthesized pair of values separated by a comma representing the real and imaginary part of the variable. For example, *(1, 2)* indicates that the real part is *1* and the imaginary part is *2*. A complex number can also be represented by the magnitude and angle format like this *(1 > 45)* indicating a complex value with a magnitude of *1* and an angle of *45* degrees

You will need to implement the *Complex* class, and provide operations for the plus, minus, multiply, and divide calculations. You will **NOT** need an exponentiation operator for this assignment. The *Complex* class will need a constructor with no arguments (default constructor), one with two arguments with initial values of both the real and imaginary part, and a third constructor that builds a complex number from a *const string&,* such as *Complex("123, 456").* You will likely need the *length()* and *empty()* methods that give the length of a string and a Boolean *true* value if the string is empty. You will also need a member function to calculate the magnitude of the complex value, the angle of the value, and the complex conjugate of the value. Finally, you will create a *Print()* method in your *Complex* class to print the value of the complex number.

1. Design a program to implement matrix operations, such as add, subtract and multiply (we won’t do divide). In order to do this, we will create a class called *Matrix* that processes a two-dimensional matrix. This class contains a constructor that builds the matrix with data from a character string. To describe a matrix with a string, we use parenthesis to delineate the rows of the matrix. For example: *(1,2,3),(4,5,6),(7,8,9)* would represent the matrix:

The three types of matrix operations should be covered in the method(s). We will also use a *Not A Matrix* flag in our matrix class to indicate that the matrix is invalid. This would be set when the size of the matrices being added or multiplied are not compatible.

**Specific Program Requirements**

1. You must define and implement a *Matrix* class, with a constructor with a string argument, to construct a matrix with initial contents. In this case, the size of the matrix is apparent from the input string.
2. Since your Matrix class allocates memory in the constructor, you MUST implement a destructor that frees the memory.
3. You must implement a *IsNaM* function that returns a Boolean *true/false* indicating whether the matrix is *Not a Matrix*.
4. The *Matrix* class must implement indexing operator (operator[]) to access individual elements in the matrix
5. All matrix operations must be implemented as member functions