# San Francisco Bay University

## CS360 - Programming in C and C++
## Homework Assignment #1

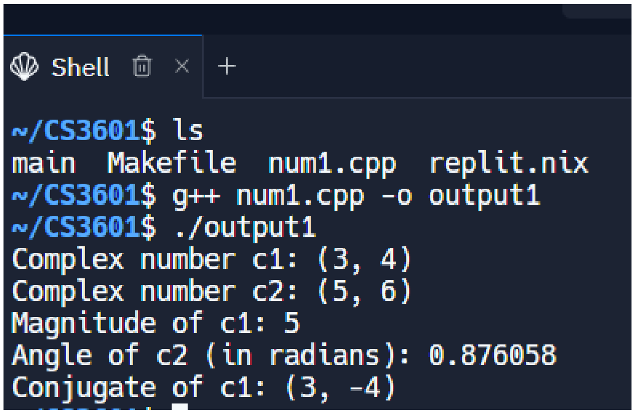
**Due day: 2/14/2024**

**Kritika Regmi**
**ID:19702**

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.
   ANS: Code is in the link attached.

   Output:

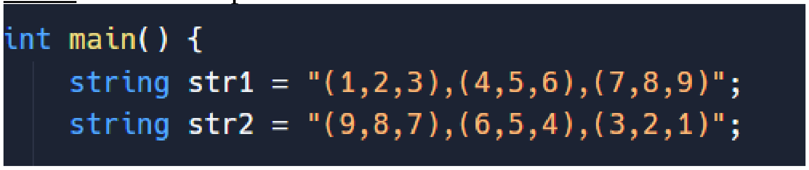


```
~/CS3601$ ls
main  Makefile  num1.cpp  replit.nix
~/CS3601$ g++ num1.cpp -o output1
~/CS3601$ ./output1
Complex number c1: (3, 4)
Complex number c2: (5, 6)
Magnitude of c1: 5
Angle of c2 (in radians): 0.876058
Conjugate of c1: (3, -4)
```

2. 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: $\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$

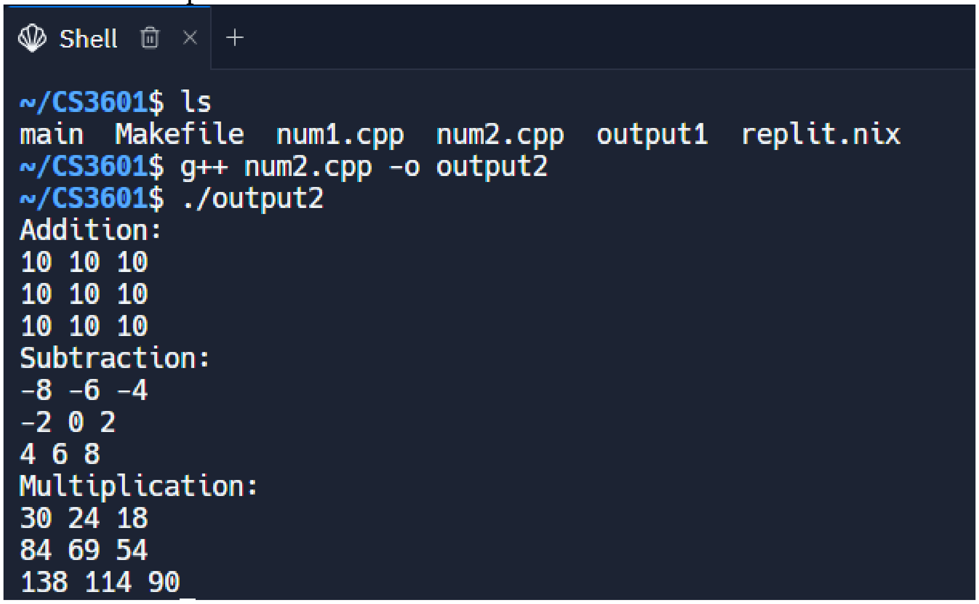
   The three types of matrix operations should be covered in the method(s). We will also use a *Not A Matrix* flag in our matrix class to indicate that the matrix is invalid. This would be set when the size of the matrices being added or multiplied are not compatible.

ANS: Here is the part in the code where matrix is defined.



```
int main() {
    string str1 = "(1,2,3),(4,5,6),(7,8,9)";
    string str2 = "(9,8,7),(6,5,4),(3,2,1)";
```

This is the output.



```
~/CS3601$ ls
main  Makefile  num1.cpp  num2.cpp  output1  replit.nix
~/CS3601$ g++ num2.cpp -o output2
~/CS3601$ ./output2
Addition:
10 10 10
10 10 10
10 10 10
Subtraction:
-8 -6 -4
-2 0 2
4 6 8
Multiplication:
30 24 18
84 69 54
138 114 90
```

Entire code is in the link attached.