

# COMSC 165

Spring 2020

## Programming Assignment 5

Worth 20 points (2% of your grade)

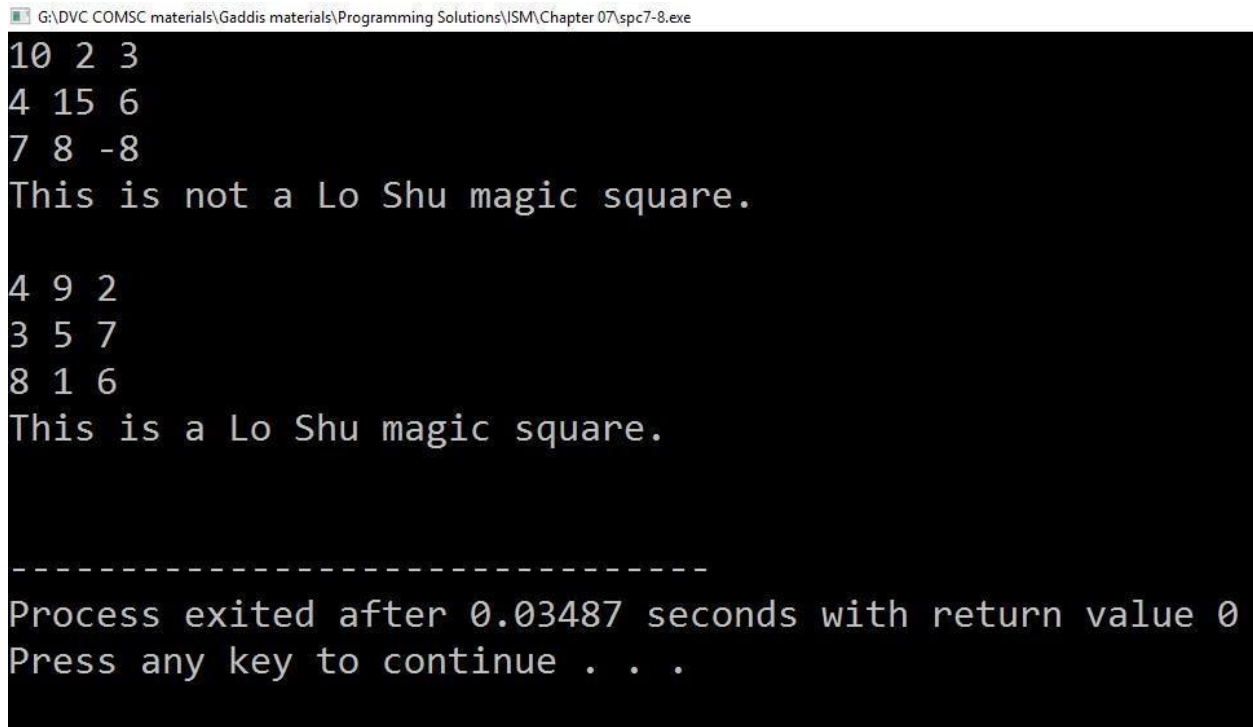
**DUE: Friday, 7/3/20 by 11:59 P.M. on  
Canvas**

**Start** by downloading the **165\_assign5.cpp** file from the Programming Assignment 5 folder on Canvas

**NOTE:** Your submission for this assignment should be a single **.cpp** file and a single **.pdf** file. The following naming convention should be used for naming your files: **firstname\_lastname\_165\_assign5.cpp** and **firstname\_lastname\_165\_assign5.pdf**. The pdf file that you submit should contain the screenshots of your sample runs of the program (see below). For example, if your first name is “James” and your last name is “Smith”, then your files should be named James\_Smith\_165\_assign5.cpp and James\_Smith\_165\_assign5.pdf.

**COMMENTS (5% of programming assignment grade):** Your program should have at least **ten (10)** different detailed comments explaining the different parts of your program. Each individual comment should be, at a minimum, a sentence explaining a particular part of your code. You should make each comment as detailed as necessary to fully explain your code. You should also number each of your comments (i.e., comment 1, comment 2, etc.).

**SAMPLE RUNS (5% of programming assignment grade):** You should submit screenshots of at least **five (5)** different sample runs of your program. Each sample run needs to use different values in the 2D array\* and your sample runs should **NOT** be the same as the sample run that is used in this write-up for this assignment. You should also number each of your sample runs (i.e., sample run 1, sample run 2, etc.). Each of your sample runs should be similar to this format:



```
G:\DVC COMSC materials\Gaddis materials\Programming Solutions\ISM\Chapter 07\spc7-8.exe
10 2 3
4 15 6
7 8 -8
This is not a Lo Shu magic square.

4 9 2
3 5 7
8 1 6
This is a Lo Shu magic square.

-----
Process exited after 0.03487 seconds with return value 0
Press any key to continue . . .
```

\*You can have multiple sample runs with the values 1-9 in the 2D array, but the values should be in a different order within the 2D array for each sample run.

For your programming assignment this week you will be implementing a program that checks whether a 2D array is a **Lo Shu Magic Square**. A Lo Shu Magic Square has the following properties:

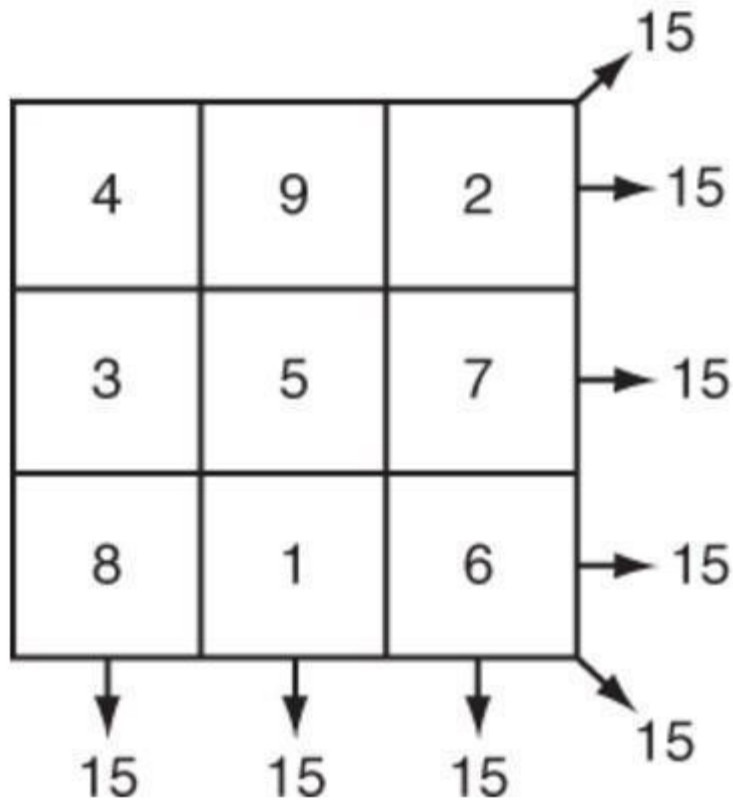
- It has three rows and three columns (i.e., a 3 X 3 grid).
- It stores the numbers 1 – 9 *exactly*. This means each of the numbers 1 – 9 must occur on the 3 X 3 grid exactly once.\*
- The sum of each row, each column, and each diagonal all add up to the same number.

\*If the 2D array contains any numbers other than 1-9, and/or if the 2D array has duplicates of any of the numbers, then the 2D array is **NOT** a Lo Shu Magic Square.

Here is an example of a 2D array that is a Lo Shu Magic Square:

```
// Create a magic two-dimensional array.
```

```
int magicArray[ROWS][COLS] = { {4, 9, 2},  
                                {3, 5, 7},  
                                {8, 1, 6} };
```



The above 2D array is a Lo Shu Magic Square because each row, each column, and each diagonal all add up the same number (15).

Here is an example of a 2D array that is **NOT** a Lo Shu Magic Square:

```
// Create a normal two-dimensional array.
int normalArray[ROWS][COLS] = { {10, 2, 3},
                                   {4, 15, 6},
                                   {7, 8, -8} };
```

The above 2D array is **NOT** a Lo Shu Magic Square for the following reasons:

- (1) It does not contain the numbers 1 – 9 exactly
- (2) Each row, each column, and each diagonal do not add up the same value

You will be implementing the following six (6) functions:

```
bool isMagicSquare(int[][COLS]);
```

```
// *****
// The isMagicSquare function accepts a two-dimensional *
// int array as an argument, and returns true if the *
// array meets all the requirements of a magic square. *
// Otherwise, it returns false. *
// *****
```

```
bool checkRange(int[][COLS]);
```

```
// *****
// The checkRange function accepts a two-dimensional int *
// array as an argument, and returns true if the values *
// are within the specified range (1- 9). Otherwise, it returns *
// false. *
// *****
```

```
bool checkUnique(int[][COLS]);
```

```
// *****  
// The checkUnique function accepts a two-dimensional int *  
// array as an argument, and returns true if the values *  
// in the array are unique. Otherwise, it returns false (if there are duplicates). *  
// *****
```

```
bool checkRowSum(int[][COLS]);
```

```
// *****  
// The checkRowSum function accepts a two-dimensional *  
// int array as an argument, and returns true if the sum *  
// of the values in each of the array's rows are equal. *  
// Otherwise, it returns false. *  
// *****
```

```
bool checkColSum(int[][COLS]);
```

```
// *****  
// The checkColSum function accepts a two-dimensional *  
// int array as an argument, and returns true if the sum *  
// of the values in each of the array's columns are *  
// equal. Otherwise, it returns false. *  
// *****
```

```
bool checkDiagSum(int[][COLS]);
```

```
// *****  
// The checkDiagSum function accepts a two-dimensional *  
// int array as an argument, and returns true if the sum *  
// of the values in each of the array's diagonals are *  
// equal. Otherwise, it returns false. *
```

```
// *****
```

**Sample Run:**

```
10 2 3
```

```
4 15 6
```

```
7 8 -8
```

```
This is not a Lo Shu magic square.
```

```
4 9 2
```

```
3 5 7
```

```
8 1 6
```

```
This is a Lo Shu magic square.
```

```
-----
```

```
Process exited after 0.03487 seconds with return value 0
```

```
Press any key to continue . . .
```