**Fall 2019 CMSC 140 Programming Project 5: Magic Square**

**Chapter(s) Covered:**

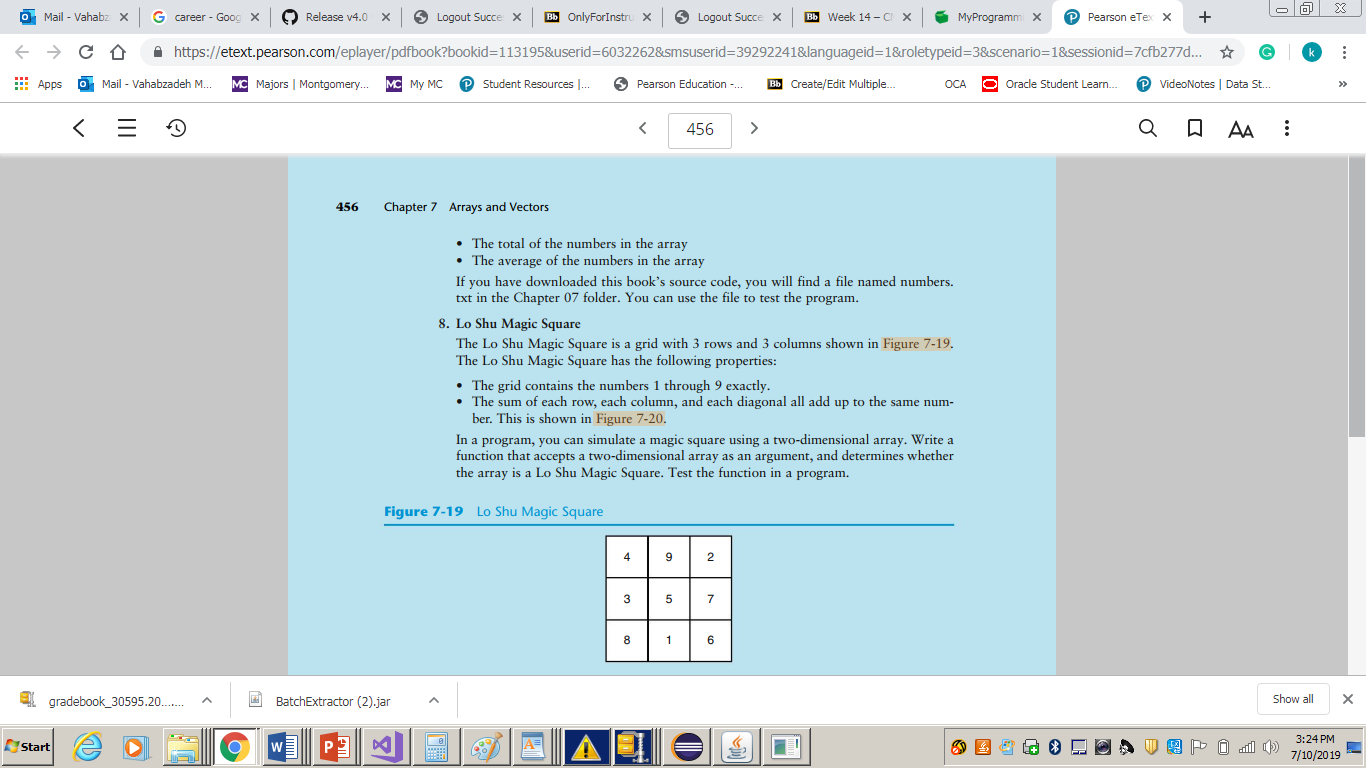
* Chapter 1-8

**Concepts tested by the program:**

* Working with one dimensional parallel arrays
* Use of functions
* Use of loops and conditional statements

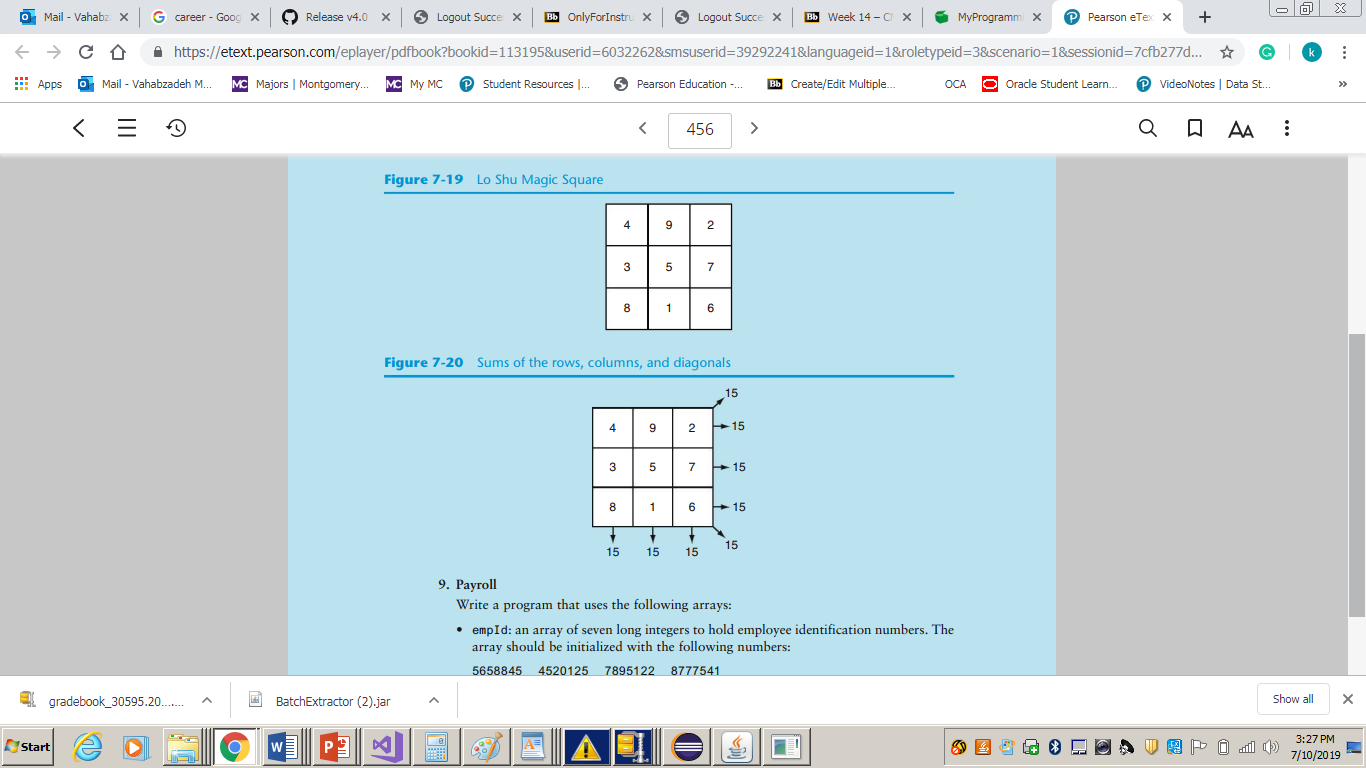
**Project Description**

The Lo Shu Magic Square is a grid with 3 rows and 3 columns shown below.



The Lo Shu Magic Square has the following properties:

* The grid contains the numbers 1 – 9 exactly
* The sum of each row, each column and each diagonal all add up to the same number.

This is shown below:

Write a program that simulates a magic square using 3 one dimensional parallel arrays of integer type.

**Each one the arrays corresponds to a row of the magic square**.

The program asks the user to enter the values of the magic square row by row and informs the user if the grid is a magic square or not.

See the [sample outputs](#SampleOutput) for more clarification.

**Project Specifications**

Input for this project:

* Values of the grid (row by row)

Output for this project:

* Whether or not the grid is magic square
* Programmer’s full name
* Project number
* Project due date

**Processing Requirements**

Use the following template to start your project:

#include<iostream>

using namespace std;

// Global constants

const int ROWS = 3; // The number of rows in the array

const int COLS = 3; // The number of columns in the array

const int MIN = 1; // The value of the smallest number

const int MAX = 9; // The value of the largest number

// Function prototypes

bool isMagicSquare(int arrayRow1[], int arrayRow2[], int arrayRow3[], int size);

bool checkRange(int arrayRow1[], int arrayRow2[], int arrayRow3[], int size, int min, int max);

bool checkUnique(int arrayRow1[], int arrayRow2[], int arrayRow3[], int size);

bool checkRowSum(int arrayrow1[], int arrayrow2[], int arrayrow3[], int size);

bool checkColSum(int arrayrow1[], int arrayrow2[], int arrayrow3[], int size);

bool checkDiagSum(int arrayrow1[], int arrayrow2[], int arrayrow3[], int size);

void fillArray(int arrayRow1[], int arrayRow2[], int arrayRow3[], int size);

void showArray(int arrayrow1[], int arrayrow2[], int arrayrow3[], int size);

int main()

{

/\* Define a Lo Shu Magic Square using 3 parallel arrays corresponding to each row of the grid \*/

int magicArrayRow1[COLS], magicArrayRow2[COLS], magicArrayRow3[COLS];

// Your code goes here

return 0;

}

// Function definitions go here

Create and use following functions:

* **void fillArray(int arrayrow1[], int arrayrow2[], int arrayrow3[], int size)** - Accepts 3 int arrays and a size as arguments, and fills the arrays out with values entered by the user. First argument corresponds to the first row of the magic square, second argument to the second row and the third argument to the third row of the magic square
* **void showArray(int arrayRow1[], int arrayRow2[], int arrayRow3[], int size)** - accepts 3 int arrays and a size as arguments and displays their content.

Example:

1 3 5 (arrayRow1)

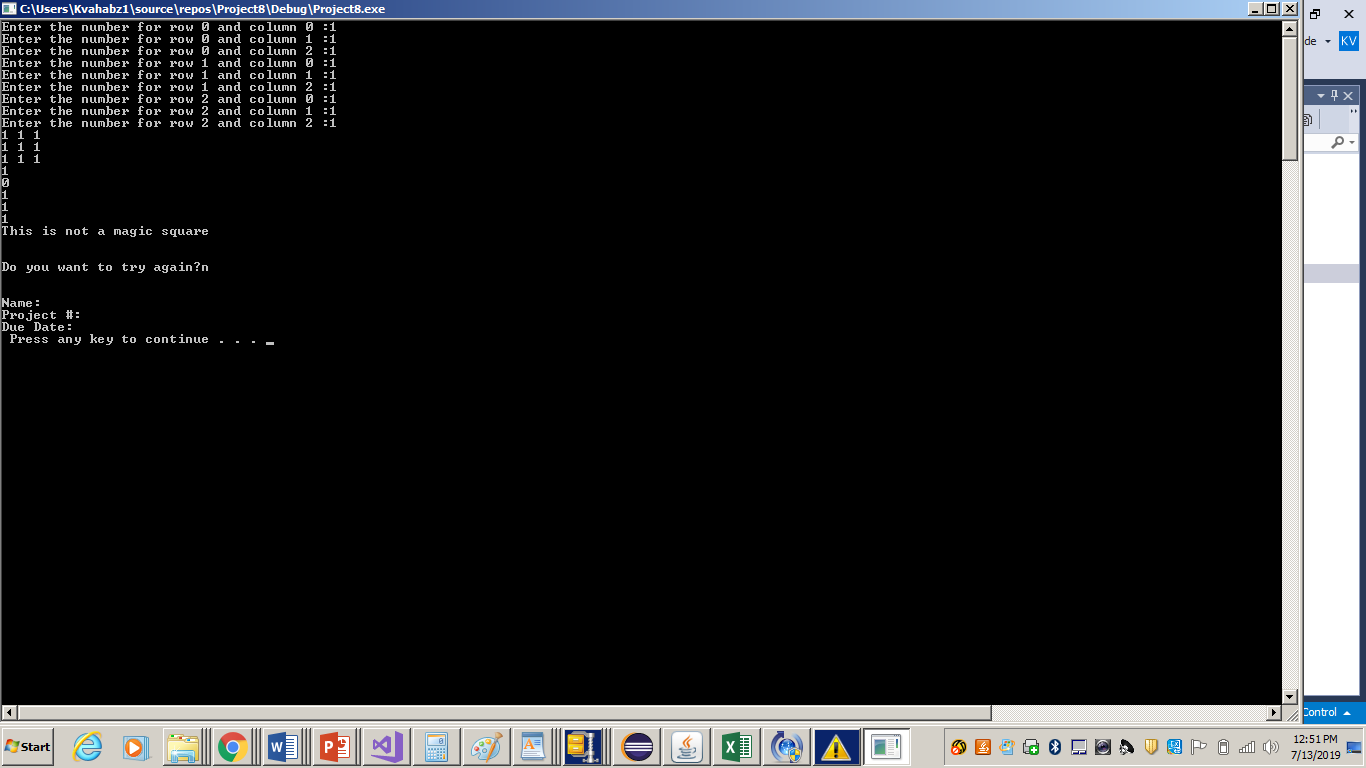
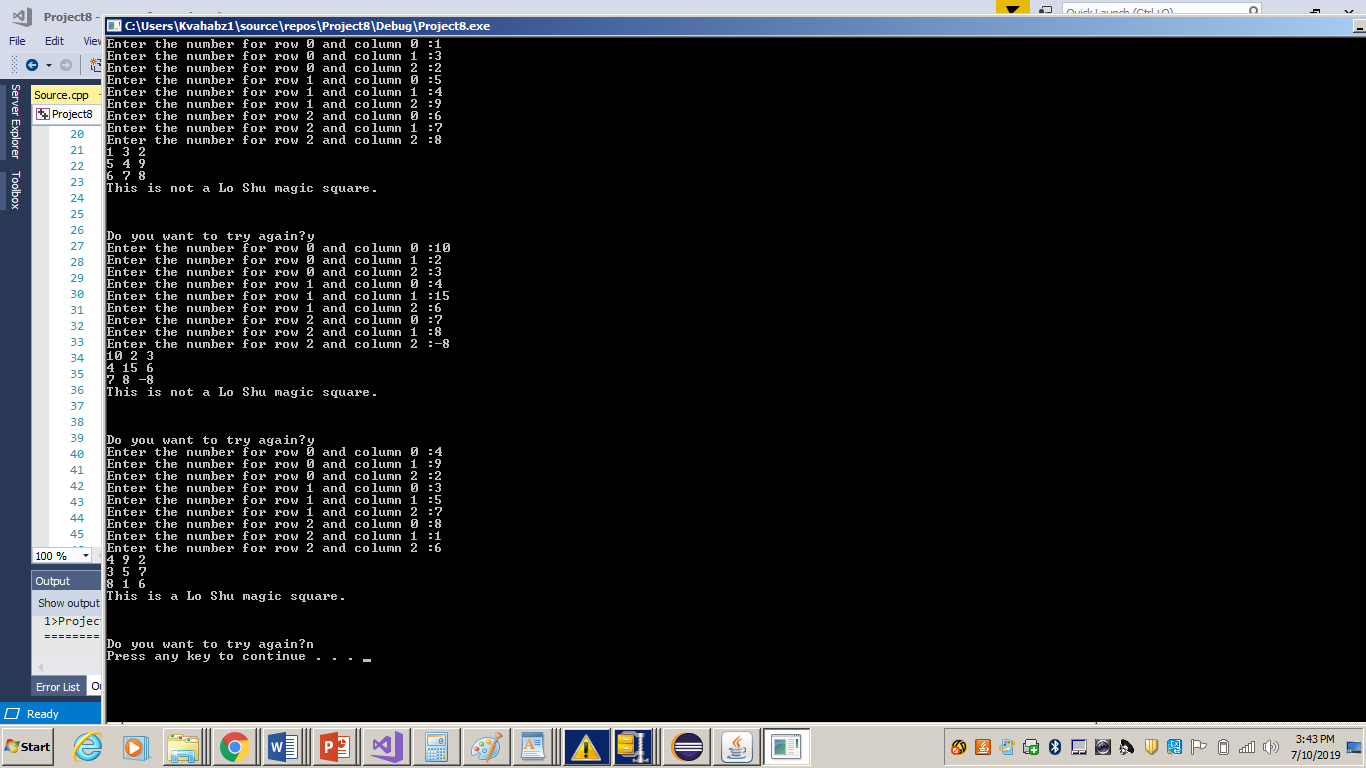
6 7 9 (arrayRow2)

8 2 4 (arrayRow3)

* **bool isMagicSquare(int arrayRow1[], int arrayRow2[], int arrayRow3[], int size)** - accepts 3 int arrays and a size as arguments and returns true if all the requirements of a magic square are met. Otherwise, it returns false. First argument corresponds to the first row of the magic square, second argument to the second row and the third argument to the third row of the magic square.
* **bool checkRange(int arrayRow1[], int arrayRow2[], int arrayRow3[], int size, int min, int max)** - accepts 3 int arrays, a size and a min and max value as arguments and returns true if the values in the arrays are within the specified range min and max. Otherwise, it returns false. First argument corresponds to the first row of the magic square, second argument to the second row and the third argument to the third row of the magic square.
* **bool checkUnique(int arrayRow1[], int arrayRow2[], int arrayRow3[], int size)** - accepts 3 int arrays and a size as arguments, and returns true if the values in the arrays are unique (only one occurrence of numbers between 1-9). Otherwise, it returns false. First argument corresponds to the first row of the magic square, second argument to the second row and the third argument to the third row of the magic square.
* **bool checkRowSum(int arrayRow1[], int arrayRow2[], int arrayRow3[], int size)** - accepts 3 int arrays and a size as arguments and returns true if the sum of the values in each of the rows are equal. Otherwise, it returns false. First argument corresponds to the first row of the magic square, second argument to the second row and the third argument to the third row of the magic square.
* **bool checkColSum(int arrayRow1[], int arrayRow2[], int arrayRow3[], int size)** - accepts 3 int arrays and a size as arguments and returns true if the sum of the values in each of the columns are equal. Otherwise, it returns false. First argument corresponds to the first row of the magic square, second argument to the second row and the third argument to the third row of the magic square.
* **bool checkDiagSum(int arrayrow1[], int arrayrow2[], int arrayrow3[], int size)** - accepts 3 int arrays and a size as arguments and returns true if the sum of the values in each of the array's diagonals are equal. Otherwise, it returns false. First argument corresponds to the first row of the magic square, second argument to the second row and the third argument to the third row of the magic square.

**[NOTE: You can create and use more functions as needed.]**

**Sample Output**



**NOTE**: Be sure to check also

1. CMSC140 Common Project Submission Requirements (.docx)
2. CMSC140 Grading Rubric\_CheckList-Project 5 (.xlsx)

**Test Plan Template**

Test your program with at least 3 more test cases. Use the given data as an example. Record your data for input and output in the following table. **Make sure your tests cover all the possible scenarios.**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case #** | **Input** | **Actual Input** | **Expected Output**  **Is Magic?** | **Actual Output**  **Is Magic?** | **Did the test pass?** |
| 1 | 1 3 2  5 4 9  6 7 8 |  | No |  |  |
| 2 | 10 2 3  4 15 6  7 8 -8 |  | No |  |  |
| 3 | 4 9 2  3 5 7  8 1 6 |  | Yes |  |  |
| 4 |  |  |  |  |  |