Lab 2: Magic Square

CMPUT 229

Background

What is a Magic Square?

A magic square is a grid of number such that each row, column, and the two main diagonals all add up to the same number. In this example, all of them add to 15.

6	1	8
7	5	3
2	9	4

Storing the Grid

Two ways of storing a 2d array:

Row-major: Each row is stored sequentially, then the next row follows, etc

1	2	3
4	5	6
7	8	9

Then becomes

1	2	3	4	5	6	7	8	9

For this assignment, we will be using the row-major format.

Column-major: Each column is stored sequentially, then the next column, etc

1	2	3
4	5	6
7	8	9

Becomes

1	4	7	2	5	8	3	6	9

Accessing Elements in a Row-Major Array

For a 2d array, we could access an element like so: array[i][k] ([row][column])

When it is stored in row-major order, each row comes directly after the previous.

So to get the k-th element in the i-th row, we have to skip forward by i rows, so we have to know how big each row is. Say they are each N elements.

Then the access becomes array[N*i + k].

Assignment

Overview

For this lab, you will be checking if the **columns** of a Magic Square are valid given what they should sum to.

magicSquare

This is the only required function for this lab.

Description:

- Determines if all of the columns of a Magic Square sum to a target sum.

Arguments:

- a0: Pointer to row-major representation of an NxN 2D array with the numbers.
- a1: N, the dimension of the Magic Square
- a2: The target sum, ie, what a column has to sum to in order to be considered valid

Returns:

- a0: If all of the columns are valid. 1 if they are, 0 otherwise.
- a1: The number of valid columns in the Magic Square

Testing

Testing your solution

Included Tests:

We have provided some tests in the ‘Tests’ folder. They are not exhaustive however, and you should do further testing of your own solution. ‘*.txt’ files correspond to the inputs, and ‘*.out’ files correspond to the expected outputs.

Give the file path as a program argument:

You can pass in the path to the test file (RARS may need the whole path) into the program argument section at the top after you assemble your program. (If there is not an input box for it go to Settings -> Program Arguments Provided To Program

Test Case Format

On the first line, have the target sum

Then on N subsequent lines, have N space separated numbers. (You do not have to explicitly define N)

So this would be a valid test case file:

```
15

6 1 8

7 5 3

2 9 4
```

And would correspond to the output file of:

```
Validity of Magic Square: 1

Number of valid columns:  3
```