Purpose: Write a MIPS program that will create a variation of a magic square.

A template file has been provided to base your program on.

Instructions

This program will ask the user to define the size and limits of a square 2 dimensional array (matrix) which the program will populate with semi-random values.

The matrix should be stored in row major order.

The program will be required to use a linear congruential generator to provide randomized numbers.

The goal of the program is to generate a matrix such that the sum of each column and each row is equal to a value the user specified. In order to do this, the program must repeatedly pick a row and column value to increase by 1, if that row or column is not already at its limit.

Required Functions

printMatrix – This function will print the provided 2D array to the terminal.

getRandomNumber – Uses a linear congruentional generator to create random numbers between a specified minimum and maximum range. You must update the global variable previousRandomNumber.

createMagicSquare – Uses the random number generator function to populate a 2D array with values such that each column and row has a sum equal to the specified amount. The getColumnTotal and getRowTotal functions must be used to ensure that the matrix is generated properly.

getColumnTotal – Returns the total sum of the specified column in the array.

getRowTotal – Returns the total sum of the specified row in the array.

Linear Congruential Generator

Random numbers will be provided by using a formula that will cycle between a large range of values. The previous value generated will need to be utilized to get the next value in the sequence.

This random value can then be used to create a random number in a desired range.

nextRandom = (previousRandom*A+C)%M
$M=2^{16}+1$
A=75
C=74

Make sure to overwrite the previousRandom number with the newly generated one.

The new random value can be turned into a range bound random value like so:

```
randomNumber = nextRandom % (maximumValue-minimumValue+1) + minimumValue
```

For example if a value from 1 to 6 was desired:

```
randomNumber = nextRandom % (6-1+1) + 1
randomNumber = nextRandom%6 + 1
```

nextRandom%6 will result in a value between 0 and 5.

Simplified Semi-Magic Square

The "magic" square for this program will not require all the properties of a true magic square (see: https://en.wikipedia.org/wiki/Magic_square). It only requires that the sum of each row and column in the matrix is equal to a specified amount.

Submission

Once completed, upload the MIPS assembly source code file (.asm) to the class website.

Example Execution (Error Checking)

```
Magic Square Size (2-10): 0

Size must be between 2 and 10.

Magic Square Size (2-10): 50

Size must be between 2 and 10.

Magic Square Size (2-10): 5
Magic Number: 4

Magic number must be between the square size and 1000.

Magic Number: 1005

Magic number must be between the square size and 1000.

Magic Number: 300

67 54 68 50 61
51 64 53 60 72
73 62 61 52 52
57 49 52 74 68
52 71 66 64 47
```

Example Execution (4x4 Magic Number 50)

```
Magic Square Size (2-10): 4
Magic Number: 50

17 10 12 11
10 17 13 10
9 13 11 17
14 10 14 12
```

Example Execution (7x7 Magic Number 7)

```
Magic Square Size (2-10): 7
Magic Number: 7

0 1 1 1 1 2 1
0 0 1 0 2 2 2
2 1 1 2 0 1 0
3 0 2 0 1 0 1
1 1 0 1 3 1 0
1 3 2 0 0 0 1
0 1 0 3 0 1 2
```

Example Execution (10x10 Magic Number 1000)

```
Magic Square Size (2-10): 10
Magic Number: 1000

114 97 113 99 88 103 99 98 98 91
88 80 95 115 98 103 89 120 94 118
85 113 93 115 94 105 95 97 104 99
123 89 109 87 108 92 116 96 87 93
94 107 93 84 120 98 105 107 107 85
86 120 97 101 90 99 93 116 85 113
99 85 109 102 118 102 117 64 109 95
127 96 108 82 101 94 97 97 106 92
105 109 98 107 84 97 95 99 99 107
79 104 85 108 99 107 94 106 111 107
```