**Andrew Tran**

**CSCI 1107 Lab Experience Two**

You are to do the following from your lab manual. If you use a word processor to reproduce the lab manual you need to also include the question with your answer.

Do Lab 2.1. On pages 11 – 14

Do Project 3.1 on pages 45-47.

Capture the console window for each test run and paste it in your word document.

**What to turn in:**

1. Printouts of your word document and your source code.
2. A hard copy of the questions and answers from your lab manual.
3. Compress the word document and the source code from all of your programs and submit them to the D2L assignment folder called Lab Experience Two.

Problem 1 – Magic Square Checker

#include <iostream>

#include <string>

#include <iomanip>

#include <sstream>

using namespace std;

const int NUM\_ROWS = 15;

const int NUM\_COLUMNS = 15;

typedef int TwoDimArray[NUM\_ROWS][NUM\_COLUMNS];

void printTable(int rows, int cols, TwoDimArray aTable);

bool check(TwoDimArray aTable, int rows, int cols);

int main() {

int rows, cols = 0;

TwoDimArray aTable; //initiate array

cout << "Enter the number of rows " << endl; // collect input

cin >> rows;

cout << "Enter the numbers of columns " << endl;

cin >> cols;

cout << "Enter your elements" << endl;

for (int i = 0; i < rows; i++)

{

for (int j = 0; j < cols; j++)

{

cin >> aTable[i][j];

}

} //collect input

cout << endl<< setw(7); //formatting

printTable(rows,cols,aTable); //function call

bool MagicSquare = check(aTable,rows,cols); //function call and bool init

if (MagicSquare) //function check

cout << setw(7) << "is a magic square" << endl << endl;

else

cout << setw(7) << "is not a magic square" << endl << endl;

}

void printTable(int rows, int cols, TwoDimArray aTable)

//prints the user generated array

{

for (int i = 0; i < rows; i++)

{

for (int j = 0; j < cols; j++)

{

cout << aTable[i][j] << setw(7);

}

cout << endl;

}

}

bool check(TwoDimArray aTable, int rows, int cols)

//checks if the inputted array is a magic square

{

int required = 0;

for (int j = 0; j < rows; j++) //calculate magic square sum requirement

{

required += aTable[0][j];

}

for (int i = 0; i < rows; i++) //calculate actual sum

{

int sum = 0;

for (int j = 0; j < rows; j++)

{

sum += aTable[i][j];

}

if (sum != required) //compare requirement to actual

{

return false;

}

}

for (int i = 0; i < cols; i++) //do the same thing with columns

{

int sum = 0;

for (int j = 0; j < cols; j++)

{

sum += aTable[j][i];

}

if (sum != required)

{

return false;

}

}

int Diag1Sum = 0; //same thing with diagonals

for (int i = 0; i < rows; i++)

{

Diag1Sum += aTable[i][i];

}

int Diag2Sum = 0; //the other diagonal

for (int i = 0; i < rows; i++)

{

Diag2Sum += aTable[rows - 1 - i][i];

}

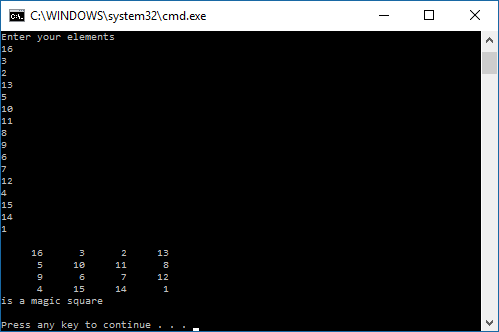
if (Diag1Sum != required || Diag2Sum != required)

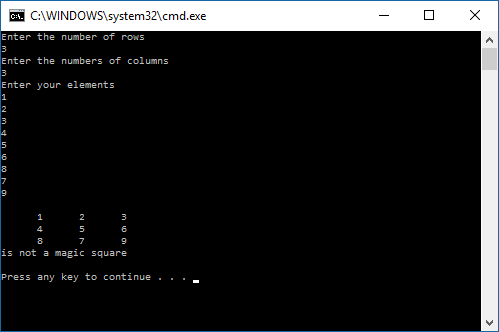
{

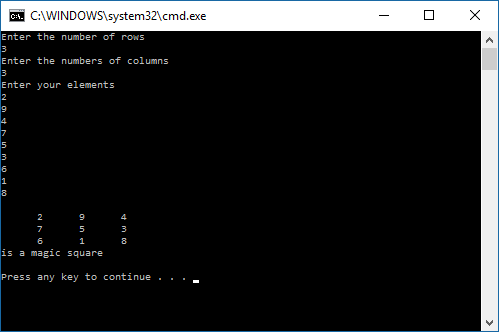
return false;

}

}







Problem 2.1 – Magic Square Constructor

#include <iostream>

#include <iomanip>

using namespace std;

int main()

{

int x;

cout << "Please enter an odd integer to seed the magic square: ";

cin >> x; //collect input

int MagicSquare[99][99] = { 0 }; //initiate the array

int rows,

cols;

int a = 0;

int b = x / 2;

for (int i = 1; i <= x\*x; i++)

{

MagicSquare[a][b] = i;

rows = (a + 1) % x;

cols = (b + 1) % x;

if (MagicSquare[rows][cols] == 0) //reset

{

a = rows;

b = cols;

}

else

{

a = (a - 1 + x) % x;

}

}

for (int k = 0; k<x; k++) //print the magic square

{

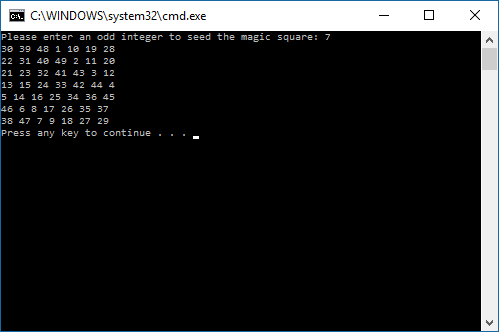
for (int y = 0; y<x; y++)

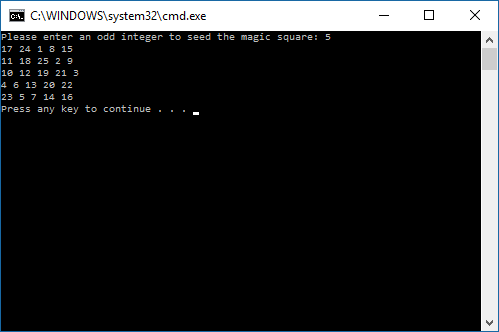
cout << MagicSquare[k][y] << " ";

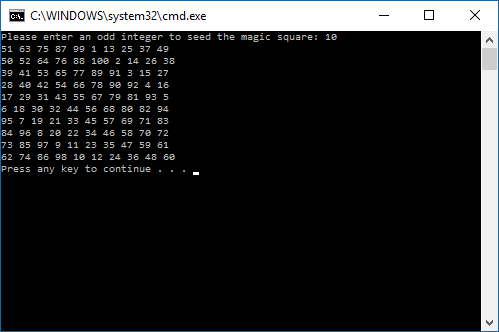
cout << endl;

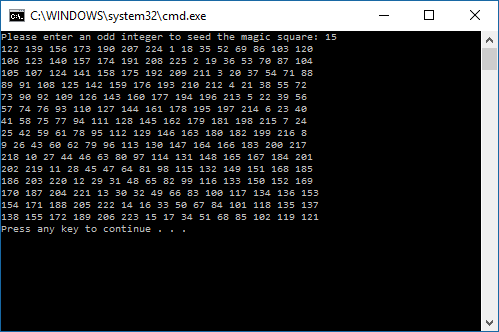
}

}









Problem 2.1 - Addressing

#include<iostream>

using namespace std;

int main() {

short int si1, si2;

int i1, i2;

long int li1, li2;

float f1, f2;

double d1, d2;

long double ld1, ld2;

bool b1, b2;

char c1, c2;

cout << &si1 << endl;

cout << &si2 << endl;

cout << &i1 << endl;

cout << &i2 << endl;

cout << &li1 << endl;

cout << &li2 << endl;

cout << &f1 << endl;

cout << &f2 << endl;

cout << &d1 << endl;

cout << &d2 << endl;

cout << &ld1 << endl;

cout << &ld2 << endl;

cout << &b1 << endl;

cout << &b2 << endl;

cout << &c1 << endl;

cout << &c2 << endl;

}