#include <iostream>

#include <iomanip>

using namespace std;

//constants

// number of rows in the array

const int ROWS = 3;

// number of columns in the array

const int COLS = 3;

// value of the smallest number

const int MIN = 1;

// value of the largest number

const int MAX = 9;

// 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()

{

//magic araay

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

char again = 'y';

do {

//function to fill magic array

fillArray(magicArrayRow1, magicArrayRow2, magicArrayRow3, COLS);

//function to showArray

showArray(magicArrayRow1, magicArrayRow2, magicArrayRow3, COLS);

if (isMagicSquare(magicArrayRow1, magicArrayRow2, magicArrayRow3, COLS))

{

//prompt user that this is magic square

cout << "This is a Lo Shu magic square." << endl;

}

else

//prompt user that this is not magic square

cout << "This is not a Lo Shu magic square." << endl;

// prompt user whether user want to play again or not

cout << "\n\nDo you want to try again? ";

cin >> again;

} while (again == 'y' || again == 'Y');

system("pause");

return 0;

}

// function to fill the array of magic square

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

{

int rowNumber = 0;

for (int columnNumber = 0; columnNumber < size; columnNumber++)

{

//prompt user to enter the number of row and column

cout << "Enter the number for row " << rowNumber

<< " and for column " << columnNumber << " : ";

cin >> arrayRow1[columnNumber];

}

rowNumber++;

for (int columnNum = 0; columnNum < size; columnNum++)

{

//prompt user to enter the number of row and column

cout << "Enter the number for row " << rowNumber

<< " and for column " << columnNum << " : ";

cin >> arrayRow2[columnNum];

}

rowNumber++;

for (int colNum = 0; colNum < size; colNum++)

{

//prompt user to enter the number of row and column

cout << "Enter the number for row " << rowNumber

<< " and for column " << colNum << " : ";

cin >> arrayRow3[colNum];

}

}

// function to show the filled array

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

{

int rowNum = 0;

for (int colNum = 0; colNum < size; colNum++)

{

cout << arrayrow1[colNum] << " ";

}

cout << endl;

rowNum++;

for (int colNum = 0; colNum < size; colNum++)

{

cout << arrayrow2[colNum] << " ";

}

cout << endl;

rowNum++;

for (int colNum = 0; colNum < size; colNum++)

{

cout << arrayrow3[colNum] << " ";

}

cout << endl;

}

// check range of the user input number

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

{

bool status = true;

for (int colNum = 0; colNum < COLS; colNum++)

{

if (arrayRow1[colNum] < min || arrayRow1[colNum] > max)

{

status = false;

}

else if (arrayRow2[colNum] < min || arrayRow2[colNum] > max)

{

status = false;

}

else if (arrayRow3[colNum] < min || arrayRow3[colNum] > max)

{

status = false;

}

}

return status;

}

//function to check input by the user is unique or not

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

{

int i = 0, j = 0, k = 0;

bool status = true;

while (i < (sizeof(arrayRow1) / sizeof(arrayRow1[0]))

&& j < (sizeof(arrayRow2) / sizeof(arrayRow2[0]))

&& k < (sizeof(arrayRow3) / sizeof(arrayRow3[0])))

{

if ((arrayRow1[i] == arrayRow2[j]) && (arrayRow2[j] == arrayRow3[k]))

{

if (i > 1)

{

status = false;

}

i++;

j++;

k++;

}

else if (arrayRow1[i] < arrayRow2[j])

i++;

else if (arrayRow2[j] < arrayRow3[k])

j++;

else

k++;

}

return status;

}

// function to check row sum

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

{

bool status = true;

int sumRow1 = arrayrow1[0] + arrayrow1[1] + arrayrow1[2];

int sumRow2 = arrayrow2[0] + arrayrow2[1] + arrayrow2[2];

int sumRow3 = arrayrow3[0] + arrayrow3[1] + arrayrow3[2];

if ((sumRow1 != sumRow2) ||

(sumRow1 != sumRow3) ||

(sumRow2 != sumRow3))

{

status = false;

}

return status;

}

//function to check column sum

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

{

bool status = true;

int sumCol1 = arrayrow1[0] + arrayrow2[0] + arrayrow3[0];

int sumCol2 = arrayrow1[1] + arrayrow2[1] + arrayrow3[1];

int sumCol3 = arrayrow1[2] + arrayrow2[2] + arrayrow3[2];

if ((sumCol1 != sumCol2) ||

(sumCol1 != sumCol3) ||

(sumCol2 != sumCol3))

{

status = false;

}

return status;

}

// function to check diagonal sum

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

{

bool status = true;

int sumDiag1 = arrayrow1[0] + arrayrow2[1] + arrayrow3[2];

int sumDiag2 = arrayrow1[2] + arrayrow2[1] + arrayrow3[0];

if (sumDiag1 != sumDiag2)

{

status = false;

}

return status;

}

//function to check magic square

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

{

bool status = false;

bool isInRange = checkRange(arrayRow1, arrayRow2, arrayRow3, size, MIN, MAX);

bool isUnique = checkUnique(arrayRow1, arrayRow2, arrayRow3, size);

bool isRowEqual = checkRowSum(arrayRow1, arrayRow2, arrayRow3, size);

bool isColEqual = checkColSum(arrayRow1, arrayRow2, arrayRow3, size);

bool isDiagEqual = checkDiagSum(arrayRow1, arrayRow2, arrayRow3, size);

if (isInRange && isUnique && isRowEqual

&& isColEqual && isDiagEqual)

{

status = true;

}

return status;

}

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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 | 1 3 2  5 4 9  6 7 8 | No | no | somewhat |
| 2 | 10 2 3  4 15 6  7 8 -8 | 10 2 3  4 15 6  7 8 -8 | No | no | somewhat |
| 3 | 4 9 2  3 5 7  8 1 6 | 4 9 2  3 5 7  8 1 6 | Yes | yes | somewhat |
| 4 | 482  4 15 6  3 8 2 | 482  4 15 6  3 8 2 | no | no | somewhat |









