Generate MAGIC SQUARE

|  |  |
| --- | --- |
| **PROBLEM STATEMENT** | MAGIC SQUARE is a square matrix that has numbers from 1 – NxN in such a way that 1. Sum of each row,  2. Sum of each column,  3. Sum of both diagonals Results into same value.   MAGIC Square of odd size can be filled with a standard algorithm:   * Fill middle column of first row with 1. * Next element will be filled to UP-RIGHT (considering matrix as circular) * If UP-RIGHT position is already occupied.  Fill next value down from current cell. |
| **INPUT FORMAT** | T - Number of Test Cases. Each Test Case has following format N - Dimensions of square matrix. |
| **OUTPUT FORMAT** | If your matrix is correct magic square, validator is printing true otherwise printing false.  Validator has been written already for you.  So just fill in the matrix based on above logic. |
| **CONSTRAINTS** | 1 <= T <= 10 1 <= N <= 52 N is guaranteed to be odd. |

|  |  |  |
| --- | --- | --- |
| **SAMPLE INPUT** | **SAMPLE OUTPUT** | **EXPLANATION** |
| 2 3 5 | true true | Self Explanatory |

Example Input - Output

fillMagicSq(3) ==>

8 1 6

3 5 7

4 9 2

Code

#include<string.h>

#include <stdio.h>

#include <iostream>

using namespace *std*;

#define MAX\_N 100

// Use it for debugging

void printMS(int ms[MAX\_N][MAX\_N], int n) {

for (int i = 0; i < n; i++) {

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

*printf*("%d ", ms[i][j]);

*printf*("\n");

}

}

void fillMagicSq(int ms[MAX\_N][MAX\_N], int n)

{

if (n % 2 == 0)

return;

for (auto i = 0; i < n; i++)

for (auto j = 0; j < n; j++)

{

ms[i][j] = 0;

}

int mid = (n / 2);

ms[0][mid] = 1;

int count = 1;

int i = 0;

int j = mid;

for (auto k = 1; k < n\*n; k++)

{

auto ui = (i - 1 + n) % n;

auto uj = j;

auto ri = ui;

auto rj = (j + 1) % n;

if (ms[ri][rj] == 0)

{

ms[ri][rj] = ++count;

i = ri;

j = rj;

}

else

{

i = (i + 1) % n;

ms[i][j] = ++count;

}

}

printMS(ms, n);

}

int main(void)

{

auto t = 0;

*cin* >> t;

while (t--)

{

int n;

*cin* >> n;

int ms[MAX\_N][MAX\_N];

fillMagicSq(ms, n);

}

return 0;

}