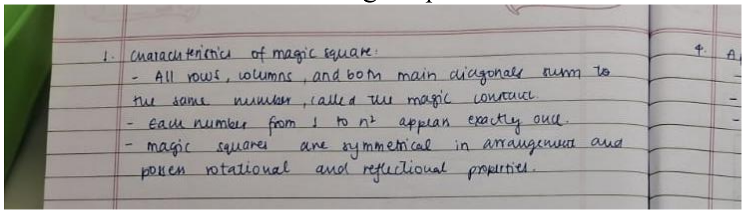
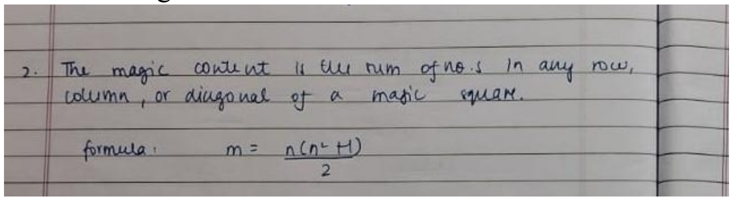
Symbiosis Institute of Technology

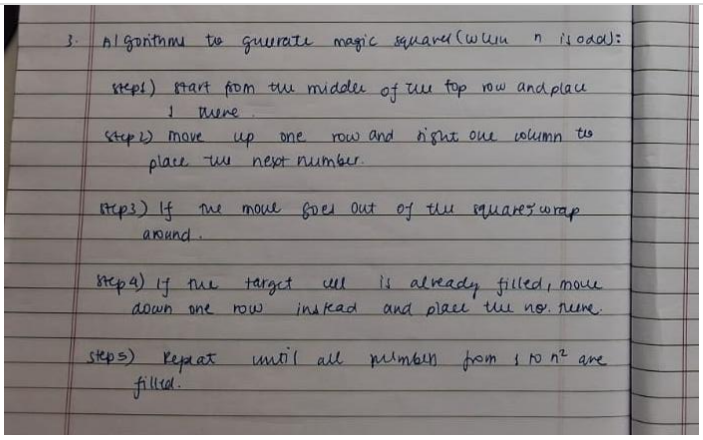
Department of Computer Science and Engineering

Academic Year 2025-26

Design Analysis of Algorithm– Lab

Batch 2023-27 - Sem V

Lab Assignment No:- 7	
Name of Student	Deepti Pal
PRN No.	23070122081
Batch	2023-27 TY CSE
Class	CSE- A3
Academic Year & Semester	2025-26 TY, 5th semester
Date of Submission	6 October 2025, Monday
Title of Assignment:	Design and Analyze an Algorithm to Generate Magic Square of size n×n, where n ≥ 3 and n is odd.
Theory: (Handwritten)	1. Write the characteristics of magic square.  1. characteristics of magic square: - All rows, columns, and both main diagonals sum to the same number, called the magic constant. - Each number from 1 to n^2 appears exactly once. - magic squares are symmetrical in arrangement and possess rotational and reflectional properties. 2. What is a magic constant? How to calculate it?  2. The magic constant is the sum of no.s in any row, column, or diagonal of a magic square. formula: $m = \frac{n(n^2+1)}{2}$ 3. Discuss the algorithms to generate magic squares of the following types ● Odd-order magic squares (n is odd) -Siamese method



3. Algorithms to generate magic square (when n is odd):

step1) start from the middle of the top row and place 1 there.

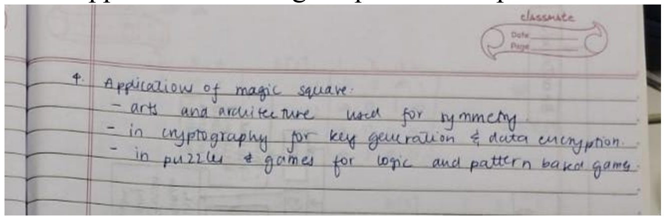
step2) move up one row and right one column to place the next number.

step3) If the move goes out of the square, wrap around.

step4) If the target cell is already filled, move down one row instead and place the no. there.

step5) Repeat until all number from 1 to n^2 are filled.

4. Give applications of magic square and explain them in brief.



classmate

Date

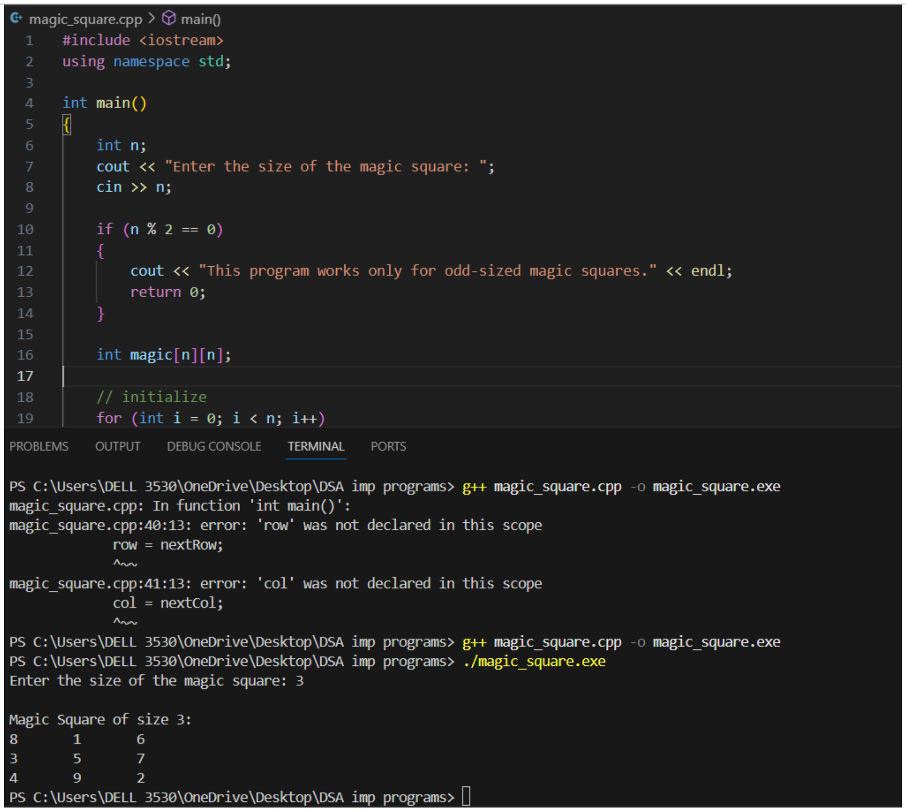
Page

4. Application of magic square.
- arts and architecture used for symmetry.
- in cryptography for key generation & data encryption.
- in puzzles & games for logic and pattern based game.

Source code

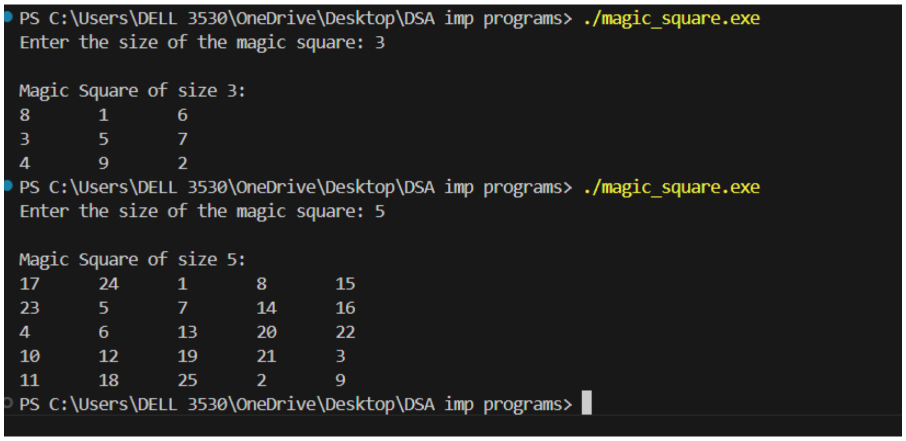
```cpp
#include <iostream>
using namespace std;

int main()
{
    int n;
    cout << "Enter the size of the magic square: ";
    cin >> n;

    if (n % 2 == 0)
    {
        cout << "This program works only for odd-sized 
magic squares." << endl;
        return 0;
    }

    int magic[n][n];

    // initialize
    for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
            magic[i][j] = 0;

    int i = 1;
    int r = 0;
    int c = n / 2;

    while (i <= n * n)
    {
        magic[r][c] = i++;

        int nextRow = (r - 1 + n) % n;
        int nextCol = (c + 1) % n;

        if (magic[nextRow][nextCol] != 0)
        {
            r = (r + 1) % n;
        }
        else
        {
            r = nextRow;
            c = nextCol;
        }
    }

    // Print
    cout << "\nMagic Square of size " << n << ":\n";
    for (int i = 0; i < n; i++)
    {
        for (int j = 0; j < n; j++)
            cout << magic[i][j] << "\t";
        cout << endl;
    }

    return 0;
}
```

Output Screenshots (if applicable)	



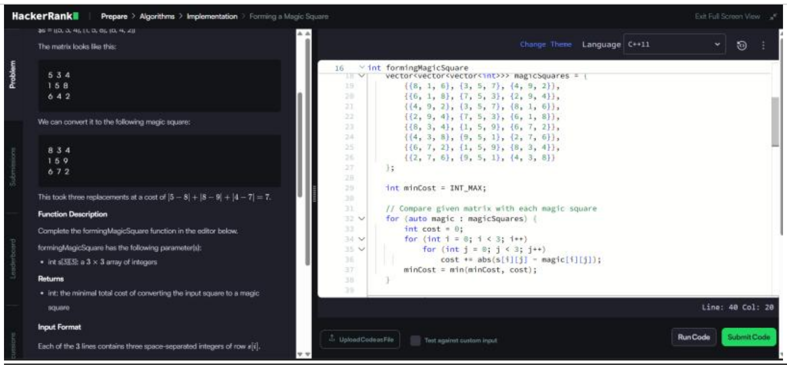
```
magic_square.cpp > main()
#include <iostream>
using namespace std;

int main()
{
    int n;
    cout << "Enter the size of the magic square: ";
    cin >> n;

    if (n % 2 == 0)
    {
        cout << "This program works only for odd-sized magic squares." << endl;
        return 0;
    }

    int magic[n][n];

    // initialize
    for (int i = 0; i < n; i++)
```

```
PS C:\Users\DELL 3530\OneDrive\Desktop\DSA imp programs> g++ magic_square.cpp -o magic_square.exe
magic_square.cpp: In function 'int main()':
magic_square.cpp:40:13: error: 'row' was not declared in this scope
             row = nextRow;
             ^~~
magic_square.cpp:41:13: error: 'col' was not declared in this scope
             col = nextCol;
             ^~~
PS C:\Users\DELL 3530\OneDrive\Desktop\DSA imp programs> g++ magic_square.cpp -o magic_square.exe
PS C:\Users\DELL 3530\OneDrive\Desktop\DSA imp programs> ./magic_square.exe
Enter the size of the magic square: 3

Magic Square of size 3:
8       1       6
3       5       7
4       9       2
PS C:\Users\DELL 3530\OneDrive\Desktop\DSA imp programs>
```



```
PS C:\Users\DELL 3530\OneDrive\Desktop\DSA imp programs> ./magic_square.exe
Enter the size of the magic square: 3

Magic Square of size 3:
8       1       6
3       5       7
4       9       2
PS C:\Users\DELL 3530\OneDrive\Desktop\DSA imp programs> ./magic_square.exe
Enter the size of the magic square: 5

Magic Square of size 5:
17      24      1       8       15
23      5       7       14      16
4       6       13      20      22
10      12      19      21      3
11      18      25      2       9
PS C:\Users\DELL 3530\OneDrive\Desktop\DSA imp programs>
```

Problems Solved from Hacker Rank	1. https://www.hackerrank.com/challenges/magic-square-forming/problem



HackerRank | Prepare > Algorithms > Implementation > Forming a Magic Square

The matrix looks like this:

```
5 3 4
1 5 8
6 4 2
```

We can convert it to the following magic square:

```
8 3 4
1 5 9
6 7 2
```

This took three replacements at a cost of $|5-8|+|8-9|+|4-7|=7$.

Function Description

Complete the formingMagicSquare function in the editor below.

formingMagicSquare has the following parameter(s):

- int s[3][3]: a 3 × 3 array of integers

Returns

- int: the minimal total cost of converting the input square to a magic square

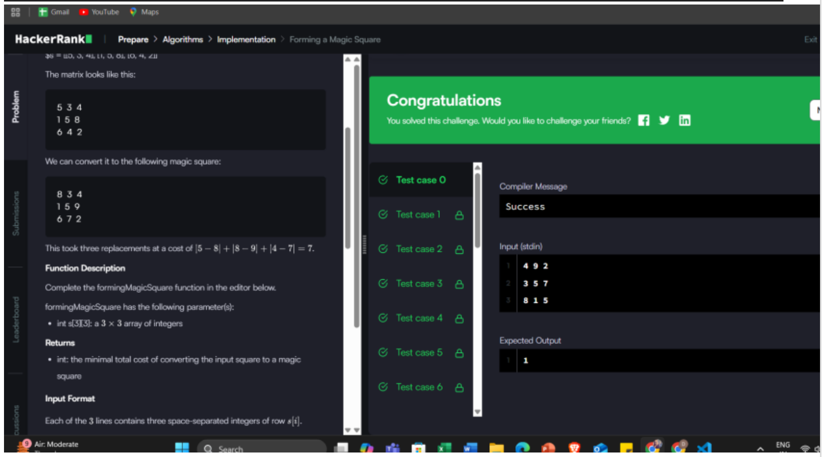
Input Format

Each of the 3 lines contains three space-separated integers of row $s[i]$.

Language C++11

```cpp
int formingMagicSquare
    vector<vector<vector<int>>> magicSquares = {
        {{8, 1, 6}, {3, 5, 7}, {4, 9, 2}},
        {{6, 1, 8}, {7, 5, 3}, {2, 9, 4}},
        {{4, 9, 2}, {3, 5, 7}, {8, 1, 6}},
        {{2, 9, 4}, {7, 5, 3}, {6, 1, 8}},
        {{8, 3, 4}, {1, 5, 9}, {6, 7, 2}},
        {{4, 3, 8}, {9, 5, 1}, {2, 7, 6}},
        {{6, 7, 2}, {1, 5, 9}, {8, 3, 4}},
        {{2, 7, 6}, {9, 5, 1}, {4, 3, 8}}
    };

    int minCost = INT_MAX;

    // Compare given matrix with each magic square
    for (auto magic : magicSquares) {
        int cost = 0;
        for (int i = 0; i < 3; i++)
            for (int j = 0; j < 3; j++)
                cost += abs(s[i][j] - magic[i][j]);
        minCost = min(minCost, cost);
    }
```

Line: 40 Col: 20



HackerRank | Prepare > Algorithms > Implementation > Forming a Magic Square

The matrix looks like this:

```
5 3 4
1 5 8
6 4 2
```

We can convert it to the following magic square:

```
8 3 4
1 5 9
6 7 2
```

This took three replacements at a cost of $|5-8|+|8-9|+|4-7|=7$.

Function Description

Complete the formingMagicSquare function in the editor below.

formingMagicSquare has the following parameter(s):

- int s[3][3]: a 3 × 3 array of integers

Returns

- int: the minimal total cost of converting the input square to a magic square

Input Format

Each of the 3 lines contains three space-separated integers of row $s[i]$.

Congratulations

You solved this challenge. Would you like to challenge your friends?

Test case 0 ✓
Test case 1 ✓
Test case 2 ✓
Test case 3 ✓
Test case 4 ✓
Test case 5 ✓
Test case 6 ✓

Compiler Message

```
Success
```

Input (stdin)

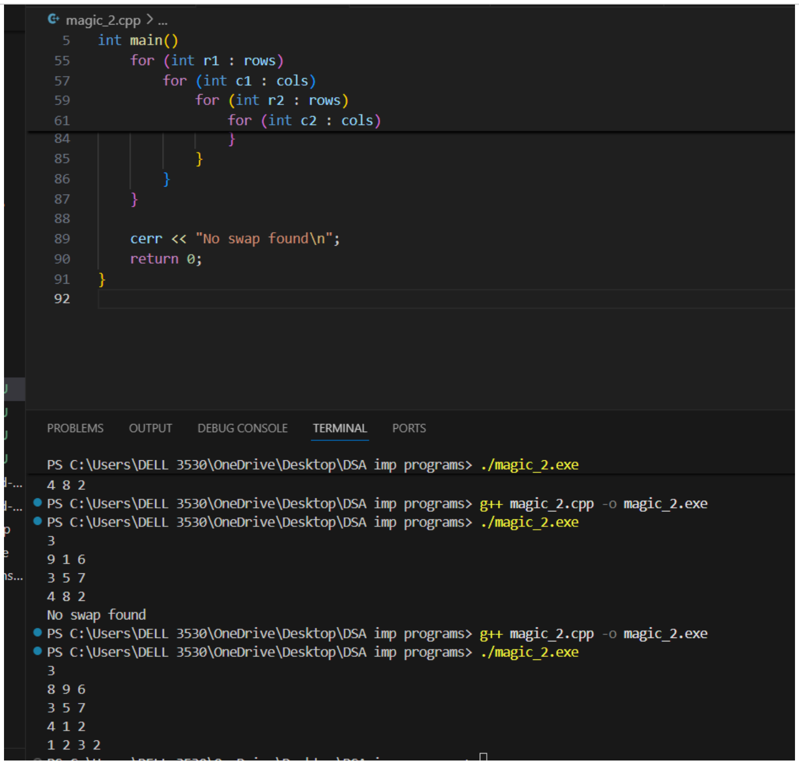
```
4 9 2
3 5 7
8 1 5
```

Expected Output

```
1
```

2. https://codeforces.com/gym/104872/problem/F

```cpp
#include <bits/stdc++.h>
using namespace std;
using ll = long long;

int main()
{
    ios::sync_with_stdio(false);
    cin.tie(nullptr);

    int n;
    cin >> n;
    vector<vector<int>> a(n, vector<int>(n));
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < n; ++j)
            cin >> a[i][j];

    vector<ll> row(n, 0), col(n, 0);
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < n; ++j)
        {
            row[i] += a[i][j];
            col[j] += a[i][j];
        }
    // Find the most common sum (the correct magic sum)
    unordered_map<ll, int> freq;
    for (auto x : row)
        freq[x]++;
    for (auto x : col)
        freq[x]++;

    ll S = 0;
    int best = 0;
    for (auto &p : freq)
        if (p.second > best)
            best = p.second, S = p.first;

    vector<int> badRows, badCols;
    for (int i = 0; i < n; ++i)
        if (row[i] != S)
            badRows.push_back(i);
    for (int j = 0; j < n; ++j)
        if (col[j] != S)
            badCols.push_back(j);

    // If everything looks "good", we still need to check
all possible swaps
    vector<int> rows = badRows.empty() ? vector<int>(n) :
badRows;
    vector<int> cols = badCols.empty() ? vector<int>(n) :
badCols;
    if (badRows.empty())
        iota(rows.begin(), rows.end(), 0);
    if (badCols.empty())
        iota(cols.begin(), cols.end(), 0);

    // Try all pairs of cells among the relevant ones
    for (int r1 : rows)
    {
        for (int c1 : cols)
        {
            for (int r2 : rows)
            {
                for (int c2 : cols)
                {
                    if (r1 == r2 && c1 == c2)
                        continue;
                    int v1 = a[r1][c1], v2 = a[r2][c2];

                    ll new_r1 = row[r1] - v1 + v2;
                    ll new_r2 = row[r2] - v2 + v1;
                    ll new_c1 = col[c1] - v1 + v2;
                    ll new_c2 = col[c2] - v2 + v1;

                    bool ok = true;
                    if (r1 != r2 && (new_r1 != S || new_r2
!= S))
                        ok = false;
                    if (c1 != c2 && (new_c1 != S || new_c2
!= S))
                        ok = false;
                    if (ok)
                    {
                        cout << r1 + 1 << " " << c1 + 1 <<
" "
                             << r2 + 1 << " " << c2 + 1 <<
"\n";
                        return 0;
                    }
                }
            }
        }
    }

    cerr << "No swap found\n";
    return 0;
}
```

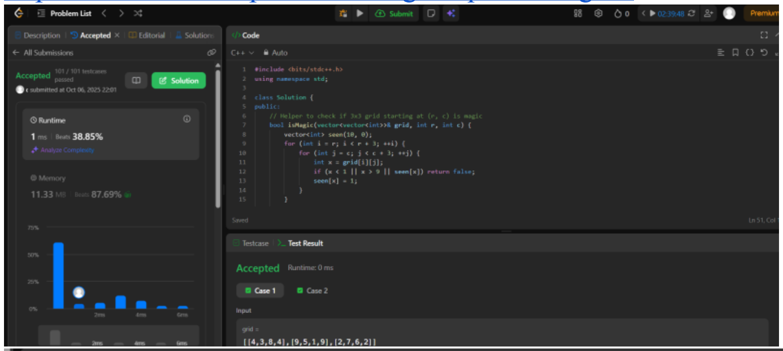


```
magic_2.cpp > ...
int main()
    for (int r1 : rows)
        for (int c1 : cols)
            for (int r2 : rows)
                for (int c2 : cols)
                }
            }
        }
    }

    cerr << "No swap found\n";
    return 0;
}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS C:\Users\DELL 3530\OneDrive\Desktop\DSA imp programs> ./magic_2.exe
4 8 2
PS C:\Users\DELL 3530\OneDrive\Desktop\DSA imp programs> g++ magic_2.cpp -o magic_2.exe
PS C:\Users\DELL 3530\OneDrive\Desktop\DSA imp programs> ./magic_2.exe
3
9 1 6
3 5 7
4 8 2
No swap found
PS C:\Users\DELL 3530\OneDrive\Desktop\DSA imp programs> g++ magic_2.cpp -o magic_2.exe
PS C:\Users\DELL 3530\OneDrive\Desktop\DSA imp programs> ./magic_2.exe
3
8 9 6
3 5 7
4 1 2
1 2 3 2
```

3. https://leetcode.com/problems/magic-squares-in-grid/

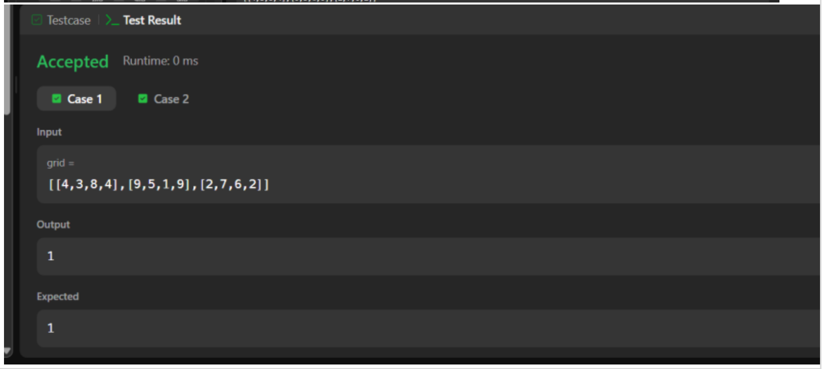


Accepted 101 / 101 testcases passed
submitted at Oct 06, 2025 22:01

Runtime 1 ms Beats 38.85%

Memory 11.33 MB Beats 87.69%

```cpp
#include <bits/stdc++.h>
using namespace std;

class Solution {
public:
    // Helper to check if 3x3 grid starting at (r, c) is magic
    bool isMagic(vector<vector<int>>& grid, int r, int c) {
        vector<int> seen(10, 0);
        for (int i = r; i < r + 3; ++i) {
            for (int j = c; j < c + 3; ++j) {
                int x = grid[i][j];
                if (x < 1 || x > 9 || seen[x]) return false;
                seen[x] = 1;
            }
        }
```



Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 Case 2

Input

grid =
[[4,3,8,4],[9,5,1,9],[2,7,6,2]]

Output

1

Expected

1

Conclusion	Thus, we have studied magic square generation using different algorithms.