Course Code: CSE 205
Course Title: Algorithms Theory

KSA – 2 : Assignment

Assignment No. 02

Student Name : Jahidul Islam

Student ID 221002504

Section : 221-D7

Date of Assignment : **25**/12/2023

Date of Submission : 05/01/2024

Submitted to : **Dr. Faiz Al Faisal,** Assistant Professor & Associate Chairperson, CSE, Green University of Bangladesh

To be Filled by the Teacher.

Received Date :

Late Submission :

Obtained Mark :

Remarks :

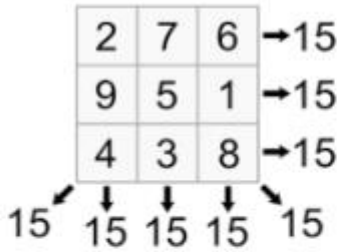
Signature:

Department of Computer Science Engineering
Green University of Bangladesh

1. **TITLE: A Lo Shu Magic Square is a 3x3 grid**

CLASS ASSIGNMENT
CSE 205: Algorithms
SUBMISSION DATE- 01/01/2024 (5:00PM) (STRICT)
(CODING ETHICS WILL BE CHECKED (CAREFUL))

1. In this Assignment, your task will be to check the grid as Lo Shu Magic Square.
2. Lo Shu Magic Square uses the formula of $n(n^2+1)/2 = 15$, (if n is 3, means it's a 3 x 3 grid). Thus, all rows, all columns and all diagonals will have the sum of 15.
3. If it's a grid of 3 x 3, then grid contains the numbers 1 through 9.
4. In this assignment, your task will be to generate a two-dimensional array to simulate a magic square. Write a function/method that generate a two-dimensional array as an argument, and determines whether the array is a Lo Shu Magic Square or not.
5. It must be a complete report with the proper explanation of your program, write algorithm accordingly (No algorithm means 0 marks) and output. Hence, writing the program only will not be enough. (It's the part of your understandability).
6. If I find the similarity with any others and copied from Internet, your report will be rejected and scored as 0. (A viva will be taken if necessary)
7. This assignment carries 5 points from your total grade.
8. Send the program in a report. Report must be in PDF version. And, send me the .c extension file also. [If I can't find the .c/.Java file, you will be scored to zero]



9. Sample Input / output-

```
Please enter the desire grid dimension: 3 3
Please output for 3 x 3 Grid:

2 7 6
9 5 1
4 3 8

This is a Lo Shu Magic Square.
And, the magic number is 15.

Thank you!!!
```

2. PROBLEM STATEMENT:

In this programming assignment, the task is to implement a program that generates a 3x3 grid simulating a Lo Shu Magic Square. A Lo Shu Magic Square is a 3x3 grid where the sum of each row, each column, and both main diagonals is the same. The magic sum is calculated using the formula n * (n^2 + 1) / 2, and for a 3x3 grid, it is 15. Additionally, the grid must contain the numbers 1 through 9, and each number must be unique.

3. OBJECTIVES/AIM:

The main objectives of this programming assignment are as follows:

Implement a C++ program that generates a 3x3 grid simulating a Lo Shu Magic Square.
Ensure that the generated grid adheres to the rules of a Lo Shu Magic Square, including the magic sum and unique numbers.
Allow user input for the 3x3 grid and display the generated square.
Check whether the generated grid is a Lo Shu Magic Square and provide appropriate feedback.

4. PROCEDURE:

1. **User Input:**
    - Prompt the user to input nine numbers for a 3x3 grid.
    - Store the user-input numbers in a two-dimensional vector representing the grid.
2. **Grid Generation:**
    - Check if the input grid is a valid 3x3 square.
    - Verify that the grid contains numbers 1 through 9 without repetition.
    - Display the generated 3x3 square.
3. **Lo Shu Magic Square Check:**
    - Calculate the sum of each row, each column, and both main diagonals.
    - Ensure that the sums are equal to the magic sum of 15.
    - If the conditions are met, declare the grid as a Lo Shu Magic Square.
    - Otherwise, provide appropriate feedback indicating why the grid is not a Lo Shu Magic Square.
4. **Output:**
    - Display the result of whether the generated grid is a Lo Shu Magic Square or not.
    - If it is, show the magic sum and the generated square.
    - If it is not, provide details on why the grid does not meet the criteria.

5. IMPLEMENTATION:

CODE IN JAVA:

```cpp
//jahidulZaid
#include <bits/stdc++.h>
using namespace std;
#define optimize() ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
#define endl '\n'

bool isLoShuMagicSquare(vector<vector<int>>& grid) {
    if (grid.size() != 3 || grid[0].size() != 3 || grid[1].size() != 3 || grid[2].size()
!= 3) {
        cout << "The grid is not a 3x3 square." << endl;
        return false;
    }

    vector<bool> found(10, false);
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3; j++)
            if (grid[i][j] < 1 || grid[i][j] > 9 || found[grid[i][j]]) {
                cout << "Invalid number in the grid." << endl;
                return false;
            } else {
                found[grid[i][j]] = true;
            }

    int magicSum = 15;
    for (int i = 0; i < 3; i++) {
        int rowSum = 0;
        int colSum = 0;
        for (int j = 0; j < 3; j++) {
            rowSum += grid[i][j];
            colSum += grid[j][i];
        }
        if (rowSum != magicSum || colSum != magicSum) {
            cout << "This is not a Lo Shu Magic Square." << endl;
            return false;
        }
    }

    if (grid[0][0] + grid[1][1] + grid[2][2] != magicSum ||
        grid[0][2] + grid[1][1] + grid[2][0] != magicSum) {
        cout << "This is not a Lo Shu Magic Square." << endl;
        return false;
    }

    cout << "This is a Lo Shu Magic Square." << endl;
    return true;
}

void displaySquare(const vector<vector<int>>& grid) {
    cout << "Generated 3x3 Square:" << endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << grid[i][j] << " ";
        }
```

```
        cout << endl;
    }
}

int main() {
    vector<vector<int>> userGrid(3, vector<int>(3));
    cout << "Enter numbers for a 3x3 grid:" << endl;
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3; j++)
            cin >> userGrid[i][j];

    displaySquare(userGrid);
    isLoShuMagicSquare(userGrid);

    return 0;
}
```

6. TEST RESULT / OUTPUT:

- Objective 1:

```
PS F:\cp resources\cf> cd "f:\cp resources\cf\" ; if ($?)
{ g++ add2.cpp -o add2 } ; if ($?) { .\add2 }
Enter numbers for a 3x3 grid:
2 9 4
7 5 3
6 1 8
Generated 3x3 Square:
2 9 4
7 5 3
6 1 8
This is a Lo Shu Magic Square.
PS F:\cp resources\cf>
```

```
PS F:\cp resources\cf> cd "f:\cp resources\cf\" ; if ($?)
{ g++ add2.cpp -o add2 } ; if ($?) { .\add2 }
Enter numbers for a 3x3 grid:
2 9 4
7 5 3
6 1 4
Generated 3x3 Square:
2 9 4
7 5 3
6 1 4
Invalid number in the grid.
PS F:\cp resources\cf>
```

7. COMPLEXITY ANALYSIS:

➢ Time Complexity Analysis:

1. **User Input:**
   - The time complexity for taking user input for a 3x3 grid is O(1), as it involves a fixed number of input operations.
2. **Grid Generation:**
   - Checking if the input grid is a valid 3x3 square involves constant time operations (O(1)).
   - Verifying the uniqueness of numbers and storing them in a vector has a time complexity of O(1) since the grid size is fixed.
3. **Lo Shu Magic Square Check:**
   - Calculating the sum of each row and column involves iterating over a constant number of elements (3), resulting in O(1) time complexity.
   - Checking both main diagonals requires O(1) operations.
   - Overall, the time complexity for checking whether the grid is a Lo Shu Magic Square is O(1).

➢ Space Complexity Analysis:

1. **User Input:**
   - The space complexity for storing the user-input grid is O(1) since the size is fixed (3x3).
2. **Grid Generation:**
   - The space complexity for storing the generated grid is O(1) as it involves a fixed number of elements.
3. **Lo Shu Magic Square Check:**
   - The space complexity for storing a vector to check the uniqueness of numbers is O(1) since the grid size is fixed.
   - Other variables used in the Lo Shu Magic Square check (like **magicSum**, **rowSum**, **colSum**) have constant space complexity.
4. **Overall Space Complexity:**
   - The overall space complexity of the program is O(1) since the memory usage remains constant regardless of the input size.

8. CONCLUSION:

- The procedure involves initializing the input, applying a dynamic programming approach through separate functions, displaying the results, and providing insights into the time complexity.
- The program successfully addresses the grid addition problem, yielding the highest and lowest totals attainable through the defined rules.

This procedure outlines the steps taken to execute the program, emphasizing clarity in input handling, dynamic programming, and results presentation.