**Question**

A class Matrix is defined to handle operations related to a matrix. The class has methods to input elements into the matrix, calculate the sum of the left and right diagonals, and display the matrix along with the sums of the diagonals. The details of the members of the class are given below:

Class name: Matrix  
Data member/instance variable:  
arr[][]: a 2D integer array to store the elements of the matrix  
M: an integer to store the number of rows in the matrix  
N: an integer to store the number of columns in the matrix

Member functions/methods:  
Matrix2(int mm, int nn): parameterized constructor to initialize the data members M and N, and create the 2D array arr  
void input(): to input the elements of the matrix  
int ldsum(): to calculate and return the sum of the left diagonal of the matrix  
int rdsum(): to calculate and return the sum of the right diagonal of the matrix  
void display(): to display the original matrix and the sums of the left and right diagonals

Specify the class Matrix2 giving the details of the constructor, void input(), int ldsum(), int rdsum(), and void display(). Also, write the main method to create an object of the class and call the methods to perform the operations.

**Algorithm**

Start

1. Define a class `Matrix2` with the following instance variables:

- `arr[][]`: a 2D integer array to store the elements of the matrix.

- `M`: an integer to store the number of rows in the matrix.

- `N`: an integer to store the number of columns in the matrix.

2. Define a constructor for the class `Matrix2`:

- Accept parameters `mm` and `nn`.

- Initialize `M` with `mm`.

- Initialize `N` with `nn`.

- Create a 2D array `arr` with dimensions `M` x `N`.

3. Define a method `input()` for the class `Matrix2`:

- Create a `Scanner` object to read input from the user.

- Prompt the user to enter the elements of the matrix.

- Use nested loops to read and store the elements in the 2D array `arr`.

- Close the `Scanner` object.

4. Define a method `ldsum()` for the class `Matrix2`:

- Initialize `s` to 0.

- Check if `M` is equal to `N`.

- If true, use nested loops to iterate through the matrix and sum the elements where the row index equals the column index.

- Return the sum `s`.

5. Define a method `rdsum()` for the class `Matrix2`:

- Initialize `s1` to 0 and `k` to `N-1`.

- Check if `M` is equal to `N`.

- If true, use a loop to iterate through the matrix and sum the elements where the column index is `k`, decrementing `k` with each iteration.

- Return the sum `s1`.

6. Define a method `display()` for the class `Matrix2`:

- Print the original matrix by iterating through the 2D array `arr`.

- Call the `ldsum()` method and store the result in `x`.

- Call the `rdsum()` method and store the result in `y`.

- Print the left diagonal sum and the right diagonal sum.

7. Define a `main` method for the class `Matrix2`:

- Create a `Scanner` object to read input from the user.

- Prompt the user to enter the number of rows.

- Prompt the user to enter the number of columns.

- Create an object of the class `Matrix2` with the entered row and column numbers.

- Call the `input()` method to read the matrix elements.

- Call the `display()` method to show the matrix and the diagonal sums.

- Close the `Scanner` object.

End

**Variable Description**

|  |  |  |
| --- | --- | --- |
| **Variable** | **Data Type** | **Purpose** |
| arr[][] | int[][] | To store the elements of array |
| M | int | To store number of rows |
| N | int | To store number of columns |
| s | int | To store sum of left diagonal elements |
| s1 | int | To store sum of right diagonal elements |
| k | int | To counter loop |
| x | int | To store the sum of left diagonal |
| y | int | To store the sum of right diagonal |
| r | int | To store the number of rows entered by the user |
| c | int | To store the number of columns entered by the user |