Matrix Rotation 90^0

Write a program to declare a square matrix A[ ][ ] of order MxM where 'M' is the number of rows and the number of columns, such that M must be greater than 2 and less than 10. Accept the value of M as user input. Display an appropriate message for an invalid input. Allow the user to input integers into this matrix. Perform the following tasks:

(a)Display the original matrix

(b) Rotate the matrix 90° clockwise as shown below:

(c) Find the sum of the elements of the four corners of the matrix.

Test your program for the following data and some random data:

Example 1

INPUT :

M = 4

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

OUTPUT :

MATRIX AFTER ROTATION

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

Sum of the corner elements = 18

```
/**
* @Question Year : ISC Practical 2015 Question 2
*/

import java.util.*;
class Q2_ISC2015
{
   public static void main(String args[])throws Exception
   {
      Scanner sc=new Scanner(System.in);
      System.out.print("Enter the size of the matrix : ");
      int m=sc.nextInt();

      if(m<3 || m>9)
         System.out.println("Size Out Of Range");
      else
      {
```

```
        int A[][]=new int[m][m];

        /* Inputting the matrix */
        for(int i=0;i<m;i++)
        {
           for(int j=0;j<m;j++)
           {
              System.out.print("Enter an element : ");
              A[i][j]=sc.nextInt();
           }
        }

        /* Printing the original matrix */
        System.out.println("*************************");
        System.out.println("The Original Matrix is : ");
        for(int i=0;i<m;i++)
        {
           for(int j=0;j<m;j++)
           {
              System.out.print(A[i][j]+"\t");
           }
           System.out.println();
        }
        System.out.println("*************************");

        /*Rotation of matrix begins here */
        System.out.println("Matrix After Rotation is : ");
        for(int i=0;i<m;i++)
        {
           for(int j=m-1;j>=0;j--)
           {
              System.out.print(A[j][i]+"\t");
           }
           System.out.println();
        }
        System.out.println("*************************");

        int sum = A[0][0]+A[0][m-1]+A[m-1][0]+A[m-1][m-1]; // Finding sum of corner elements
        System.out.println("Sum of the corner elements = "+sum);
     }
  }
}
```

Lower Triangular Matrix or not

Write a Program in Java to input a 2-D square matrix and check whether it is a Lower Triangular Matrix or not.

Lower Triangular Matrix : A Lower Triangular matrix is a square matrix in which all the entries above the main diagonal (↘) are zero. The entries below or on the main diagonal themselves may or may not be zero.

Example:

$$\begin{bmatrix} 5 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 \\ 4 & 9 & 4 & 0 \\ 6 & 8 & 7 & 2 \end{bmatrix}$$

```
import java.util.*;
class LowerTriangularMatrix
{
   public static void main(String args[])throws Exception
   {
      Scanner sc=new Scanner(System.in);
      System.out.print("Enter the size of the matrix : ");
      int m=sc.nextInt();
      int A[][]=new int[m][m];

      /* Inputting the matrix */
      for(int i=0;i<m;i++)
      {
         for(int j=0;j<m;j++)
         {
            System.out.print("Enter an element : ");
            A[i][j]=sc.nextInt();
         }
      }

      /* Printing the matrix */
      System.out.println("**************************");
      System.out.println("The Matrix is : ");
      for(int i=0;i<m;i++)
      {
         for(int j=0;j<m;j++)
         {
            System.out.print(A[i][j]+"\t");
         }
         System.out.println();
      }
      System.out.println("**************************");

      int p=0;

      for(int i=0;i<m;i++)
      {
         for(int j=i+1;j<m;j++)
         {
```

```
            /* Checking that the matrix is Lower Triangular or not */
            if(A[i][j]!=0) // All elements above the diagonal must be zero
            {
               p=1;
               break;
            }
         }
      }

      if(p==0)
         System.out.println("The matrix is Lower Triangular");
      else
         System.out.println("The matrix is not Lower Triangular");
   }
}
```

Upper Triangular Matrix or not

Write a Program in Java to input a 2-D square matrix and check whether it is an Upper Triangular Matrix or not.

Upper Triangular Matrix : An Upper Triangular matrix is a square matrix in which all the entries below the main diagonal (↘) are zero. The entries above or on the main diagonal themselves may or may not be zero.

Example:

$$\begin{bmatrix} 5 & 3 & 0 & 7 \\ 0 & 1 & 9 & 8 \\ 0 & 0 & 4 & 6 \\ 0 & 0 & 0 & 2 \end{bmatrix}$$

```
import java.util.*;
class UpperTriangularMatrix
{
   public static void main(String args[])throws Exception
   {
      Scanner sc=new Scanner(System.in);
      System.out.print("Enter the size of the matrix : ");
      int m=sc.nextInt();
      int A[][]=new int[m][m];

      /* Inputting the matrix */
      for(int i=0;i<m;i++)
      {
         for(int j=0;j<m;j++)
         {
            System.out.print("Enter an element : ");
            A[i][j]=sc.nextInt();
         }
      }
```

```
        /* Printing the matrix */
        System.out.println("**************************");
        System.out.println("The Matrix is : ");
        for(int i=0;i<m;i++)
        {
           for(int j=0;j<m;j++)
           {
              System.out.print(A[i][j]+"\t");
           }
           System.out.println();
        }
        System.out.println("**************************");

        int p=0;

        for(int i=0;i<m;i++)
        {
           for(int j=0;j<i;j++)
           {
              /* Checking that the matrix is Upper Triangular or not */
              if(A[i][j]!=0) // All elements below the diagonal must be zero
              {
                 p=1;
                 break;
              }
           }
        }

        if(p==0)
           System.out.println("The matrix is Upper Triangular");
        else
           System.out.println("The matrix is not Upper Triangular");
     }
}
```

Scalar Matrix or not

Write a Program in Java to input a 2-D square matrix and check whether it is a Scalar Matrix or not.

Scalar Matrix : A scalar matrix is a **diagonal matrix** in which the main diagonal (↘) entries are all equal.

Example:

$$\begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 5 & 0 & 0 \\ 0 & 0 & 5 & 0 \\ 0 & 0 & 0 & 5 \end{bmatrix}$$

```
import java.util.*;
class ScalarMatrix
{
   public static void main(String args[])throws Exception
   {
      Scanner sc=new Scanner(System.in);
      System.out.print("Enter the size of the matrix : ");
      int m=sc.nextInt();
      int A[][]=new int[m][m];

      /* Inputting the matrix */
      for(int i=0;i<m;i++)
      {
         for(int j=0;j<m;j++)
         {
            System.out.print("Enter an element : ");
            A[i][j]=sc.nextInt();
         }
      }

      /* Printing the matrix */
      System.out.println("**************************");
      System.out.println("The Matrix is : ");
      for(int i=0;i<m;i++)
      {
         for(int j=0;j<m;j++)
         {
            System.out.print(A[i][j]+"\t");
         }
         System.out.println();
      }
      System.out.println("**************************");

      int p = 0, q = 0, x = A[0][0]; // 'x' is storing the 1st main diagonal element

      for(int i=0;i<m;i++)
      {
         for(int j=0;j<m;j++)
         {
            /* Checking that the matrix is diagonal or not */
            if(i!=j && A[i][j]!=0) // All non-diagonal elements must be zero
            {
               p=1;
               break;
            }
            /* Checking the matrix for scalarity */
            if(i==j && (A[i][j]==0 || A[i][j]!=x))              {
               q=1;
               break;
            }
         }
      }

      if(p==0 && q==0)
         System.out.println("The matrix is scalar");
```

```
        else
            System.out.println("The matrix is not scalar");
    }
}
```

Diagonal Matrix or not

Write a Program in Java to input a 2-D square matrix and check whether it is a Diagonal Matrix or not.

Diagonal Matrix : A diagonal matrix is a matrix (usually a square matrix) in which the entries outside the main diagonal (↘) are all zero. The diagonal entries themselves may or may not be zero (but all diagonal entries cannot be zero).

Example:

$$\begin{bmatrix} 5 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 7 \end{bmatrix}$$

```
import java.util.*;
class DiagonalMatrix
{
    public static void main(String args[])throws Exception
    {
        Scanner sc=new Scanner(System.in);
        System.out.print("Enter the size of the matrix : ");
        int m=sc.nextInt();
        int A[][]=new int[m][m];

        /* Inputting the matrix */
        for(int i=0;i<m;i++)
        {
            for(int j=0;j<m;j++)
            {
                System.out.print("Enter an element : ");
                A[i][j]=sc.nextInt();
            }
        }

        /* Printing the matrix */
        System.out.println("**************************");
        System.out.println("The Matrix is : ");
        for(int i=0;i<m;i++)
        {
            for(int j=0;j<m;j++)
            {
                System.out.print(A[i][j]+"\t");
            }
            System.out.println();
        }
        System.out.println("**************************");
```

```
        int p=0, q=0;

        for(int i=0;i<m;i++)
        {
           for(int j=0;j<m;j++)
           {
              if(i!=j && A[i][j]!=0) // Checking non-diagonal elements
              {
                 p=1;
                 break;
              }
              if(i==j && A[i][j]==0) // Checking diagonal elements
              {
                 q++;
                 break;
              }
           }
        }

        if(p==0 && q<m)
           System.out.println("The matrix is Diagonal");
        else
           System.out.println("The matrix is not Diagonal");
     }
}
```

Program to print the possible consecutive number combinations

A positive natural number, (for e.g. 27), can be represented as follows:
2+3+4+5+6+7
8+9+10
13+14
where every row represents a combination of consecutive natural numbers, which add up to 27.

Write a program which inputs a positive natural number N and prints the possible consecutive number combinations, which when added give N.

Test your program for the following data and some random data.

SAMPLE DATA

INPUT:
N = 9

OUTPUT:
4 + 5
2 + 3+ 4