**Week 7: Programming Assignment 1**

**Magic Square**

A magic square of order **n** is a square arrangement of **n2** numbers, typically distinct integers, where the sum of the **n** numbers in each row, each column, and both main diagonals is the same constant. It consists of the integers from 1 to .

The constant sum of each row, column, and diagonal in a magic square is known as the magic constant or magic sum, **M**. For a normal magic square, this constant depends solely on n and is given by the formula:  
  
M=(n(n2+1))/2M=(n(n2+1))/2

For normal magic squares of order n = 3,4,5,...,

the magic constants are: 15,34,65,111,175,260,...

Given a matrix, check whether it’s Magic Square or not.

**Input Format:**

An integer value for **n (n>2)**

An **nXn** matrix

**Output Format:**

If the given matrix is magic square then print “Magic Matrix”, otherwise “Not a Magic Matrix

**Example :**

**Input**

3

2 7 6

9 5 1

4 3 8

**Output**

Magic Matrix

def is\_magic\_square(matrix, n):

# Calculate the magic constant M

M = (n \* (n\*\*2 + 1)) // 2

# Check sum of each row

for i in range(n):

if sum(matrix[i]) != M:

return "Not a Magic Matrix"

# Check sum of each column

for j in range(n):

column\_sum = 0

for i in range(n):

column\_sum += matrix[i][j]

if column\_sum != M:

return "Not a Magic Matrix"

# Check the main diagonal (top-left to bottom-right)

diag1\_sum = 0

for i in range(n):

diag1\_sum += matrix[i][i]

if diag1\_sum != M:

return "Not a Magic Matrix"

# Check the secondary diagonal (top-right to bottom-left)

diag2\_sum = 0

for i in range(n):

diag2\_sum += matrix[i][n - 1 - i]

if diag2\_sum != M:

return "Not a Magic Matrix"

return "Magic Matrix"

# Input reading

n = int(input()) # Size of the matrix

matrix = [list(map(int, input().split())) for \_ in range(n)]

# Check and print result

result = is\_magic\_square(matrix, n)

print(result,end="")

**def is\_magic\_square(matrix, n):**

**M = (n \* (n\*\*2 + 1)) // 2**

**for i in range(n):**

**if sum(matrix[i]) != M:**

**return "Not a Magic Matrix"**

**for j in range(n):**

**column\_sum = 0**

**for i in range(n):**

**column\_sum += matrix[i][j]**

**if column\_sum != M:**

**return "Not a Magic Matrix"**

**diag1\_sum = 0**

**for i in range(n):**

**diag1\_sum += matrix[i][i]**

**if diag1\_sum != M:**

**return "Not a Magic Matrix"**

**diag2\_sum = 0**

**for i in range(n):**

**diag2\_sum += matrix[i][n - 1 - i]**

**if diag2\_sum != M:**

**return "Not a Magic Matrix"**

**return "Magic Matrix"**

**n = int(input())**

**matrix = [list(map(int, input().split())) for \_ in range(n)]**

**result = is\_magic\_square(matrix, n)**

**print(result,end="")**

2 Week 7: Programming Assignment 2

**Median of Matrix**

You are given a row-wise sorted matrix of size **n×m**, where both the number of rows and the number of columns are always odd. Your task is to find the median of the matrix.

The median is the middle number in a sorted list of numbers. In case the list has an even number of elements, the median is the leftmost of the two middle elements.

The matrix is sorted row-wise, but the entire matrix is not necessarily sorted.

You need to find the median from the matrix when all the elements are sorted in ascending order.

**Input Format:**

The first line contains two integers **n** and **m**, representing the number of rows and columns in the matrix.

The next **n** lines each contain m space-separated integers representing a row of the matrix.

**Output Format:**

Print the median of the matrix.

**Constraints:**

* 3 ≤ n, m ≤ 100
* n and m are always odd.

Example :

Input:

3 3

1 3 5

2 6 9

3 6 9

Output:

5

Explanation:

The elements in sorted order: [1,2,3,3,5,6,6,9,9].

The total number of elements is 9, so the median is the element at 5th location, which is 5.

def find\_median(matrix, n, m):

# Flatten the matrix into a single list

flattened\_matrix = []

for row in matrix:

flattened\_matrix.extend(row)

# Sort the flattened list

flattened\_matrix.sort()

# The median is the middle element in the sorted list

median\_index = (n \* m) // 2 # 0-based index for the median

return flattened\_matrix[median\_index]

# Input reading

n, m = map(int, input().split()) # Reading the dimensions of the matrix

matrix = [list(map(int, input().split())) for \_ in range(n)] # Reading the matrix rows

# Find and print the median

median = find\_median(matrix, n, m)

print(median,end="")

**def find\_median(matrix, n, m):**

**flattened\_matrix = []**

**for row in matrix:**

**flattened\_matrix.extend(row)**

**flattened\_matrix.sort()**

**median\_index = (n \* m) // 2**

**return flattened\_matrix[median\_index]**

**n, m = map(int, input().split())**

**matrix = [list(map(int, input().split())) for \_ in range(n)]**

**median = find\_median(matrix, n, m)**

**print(median,end="")**

3 Week 7: Programming Assignment 3

**Rotate a Rectangular Image by 90 Degrees Clockwise**

You are given an image represented by an **m x n** matrix. Your task is to rotate the image by 90 degrees in the clockwise direction. After rotation, the dimensions of the result matrix will be **n x m**, as the number of rows and columns will be swapped.

Input Format

Two integers **m** and **n**

A matrix of integers of dimension **m X n**

Output Format

A matrix of integers of dimension **n X m**

**Example :**

**Input:**

4 4

1  2  3  4

5  6  7  8

9  10 11 12

13 14 15 16

**Output:**

13   9   5   1

14  10   6   2

15  11   7   3

16  12   8   4

def rotate\_image(matrix, m, n):

# Step 1: Transpose the matrix

transposed\_matrix = [[matrix[i][j] for i in range(m)] for j in range(n)]

# Step 2: Reverse each row of the transposed matrix

for i in range(n):

transposed\_matrix[i] = transposed\_matrix[i][::-1]

return transposed\_matrix

# Input reading

m, n = map(int, input().split()) # Read the dimensions m and n

matrix = [list(map(int, input().split())) for \_ in range(m)] # Read the matrix

# Get the rotated matrix

rotated\_matrix = rotate\_image(matrix, m, n)

# Output the rotated matrix

for i in range(n):

# Check if this is the last row to avoid adding an extra newline

if i < n - 1:

print(" ".join(map(str, rotated\_matrix[i]))) # Print row with spaces

else:

print(" ".join(map(str, rotated\_matrix[i])), end="") # No extra newline after last row

**def rotate\_image(matrix, m, n):**

**transposed\_matrix = [[matrix[i][j] for i in range(m)] for j in range(n)]**

**for i in range(n):**

**transposed\_matrix[i] = transposed\_matrix[i][::-1]**

**return transposed\_matrix**

**m, n = map(int, input().split())**

**matrix = [list(map(int, input().split())) for \_ in range(m)]**

**rotated\_matrix = rotate\_image(matrix, m, n)**

**for i in range(n):**

**if i < n - 1:**

**print(" ".join(map(str, rotated\_matrix[i])))**

**else:**

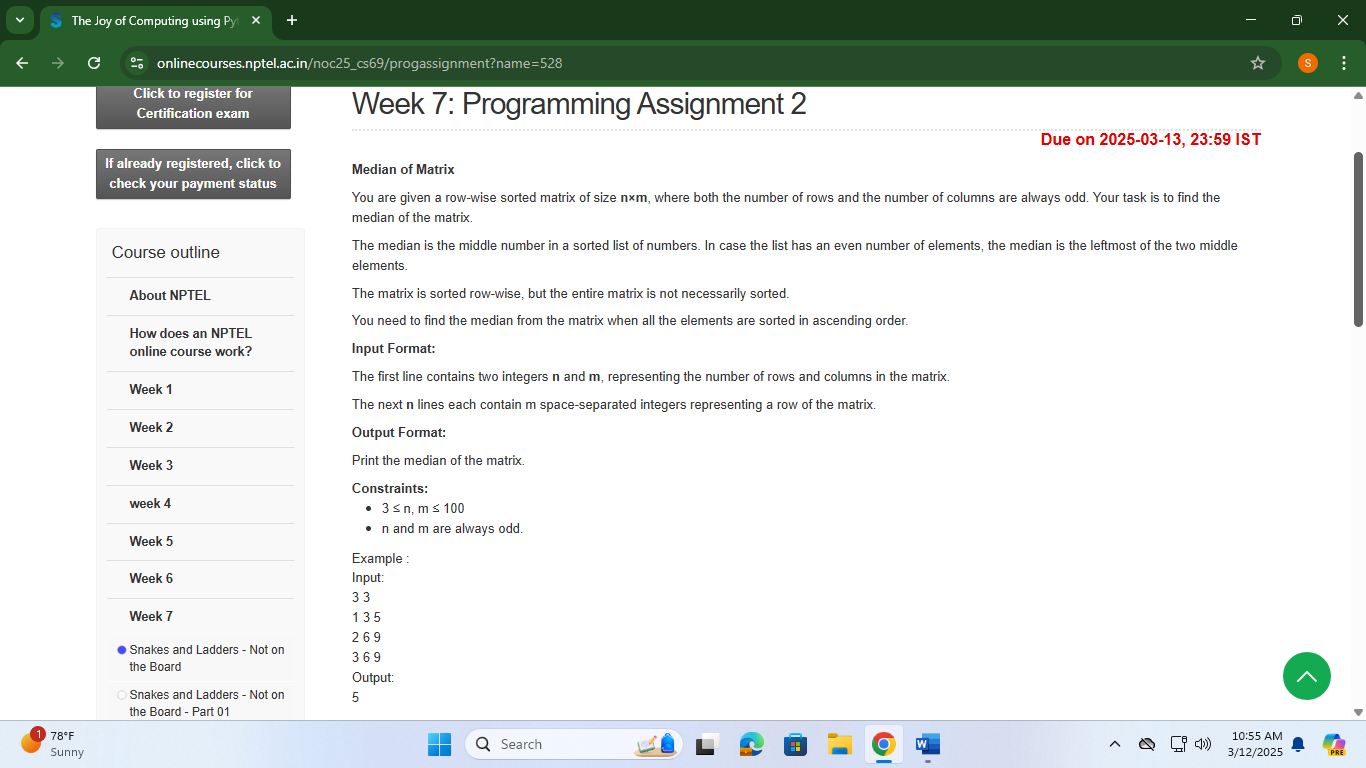
**print(" ".join(map(str, rotated\_matrix[i])), end="")**

A screenshot of a computer

AI-generated content may be incorrect.  
  
A screenshot of a computer

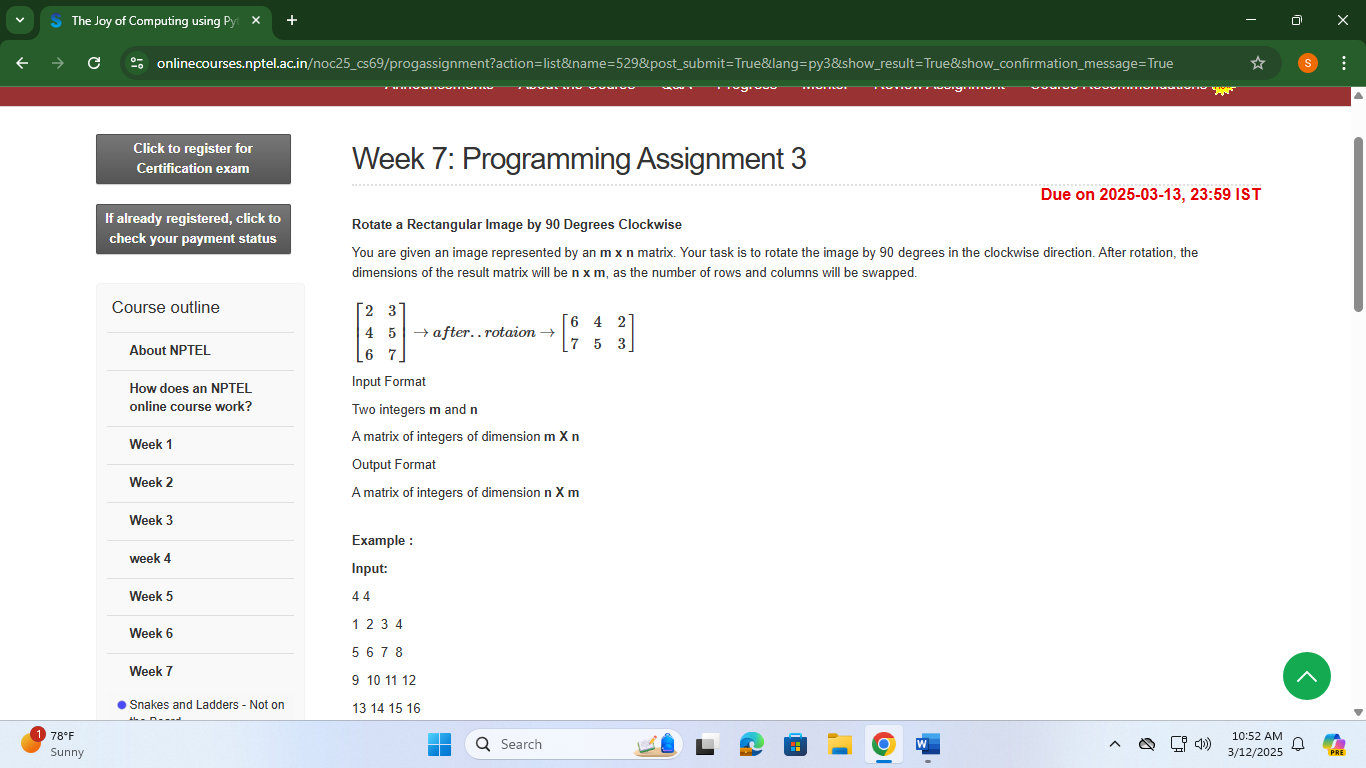
AI-generated content may be incorrect.  
  
A screenshot of a computer

AI-generated content may be incorrect.

  
  
A screenshot of a computer

AI-generated content may be incorrect.  
  
A screenshot of a computer

AI-generated content may be incorrect.

  
  
A screenshot of a computer

AI-generated content may be incorrect.  
  
A screenshot of a computer

AI-generated content may be incorrect.