# Rotate Image

You are given an n x n 2D matrix representing an image, rotate the image by **90** degrees (clockwise).

You have to rotate the image [**in-place**](https://en.wikipedia.org/wiki/In-place_algorithm), which means you have to modify the input 2D matrix directly. **DO NOT** allocate another 2D matrix and do the rotation.

**Example 1:**



**Input:** matrix = [[1,2,3],[4,5,6],[7,8,9]]

**Output:** [[7,4,1],[8,5,2],[9,6,3]]

**Example 2:**



**Input:** matrix = [[5,1,9,11],[2,4,8,10],[13,3,6,7],[15,14,12,16]]

**Output:** [[15,13,2,5],[14,3,4,1],[12,6,8,9],[16,7,10,11]]

**Constraints:**

* n == matrix.length == matrix[i].length
* 1 <= n <= 20
* -1000 <= matrix[i][j] <= 1000

Logic:

1. Get the transpose.
2. Reverse each row.

#include <iostream>

#include <vector>

using namespace *std*;

void display(*vector*<*vector*<int>>& matrix)

{

for (auto i : matrix)

{

for (auto j : i)

{

*cout* << j << " ";

}

*cout* << *endl*;

}

*cout* << *endl*;

}

void transpose(*vector*<*vector*<int>>& matrix)

{

for (*size\_t* i = 0; i < matrix.*size*(); i++)

{

// SWAP elements of upper triangle with lower tirangle

for (***size\_t* j = i**; j < matrix[0].*size*(); j++)

{

auto temp = matrix[i][j];

matrix[i][j] = matrix[j][i];

matrix[j][i] = temp;

}

}

}

void reverse(*vector*<*vector*<int>>& matrix)

{

for (*size\_t* i = 0; i < matrix.*size*(); i++)

{

int li = 0; int ri = matrix[i].*size*() - 1;

while (li < ri)

{

auto temp = matrix[i][li];

matrix[i][li] = matrix[i][ri];

matrix[i][ri] = temp;

li++;

ri--;

}

}

}

int main(void)

{

//vector<vector<int>> matrix{ {1, 2, 3},{4, 5, 6},{7, 8, 9} };

*vector*<*vector*<int>> matrix{ {5, 1, 9, 11}, {2, 4, 8, 10}, {13, 3, 6, 7}, {15, 14, 12, 16} };

display(matrix);

transpose(matrix);

display(matrix);

reverse(matrix);

display(matrix);

return 0;

}