Day 22 / 100 :

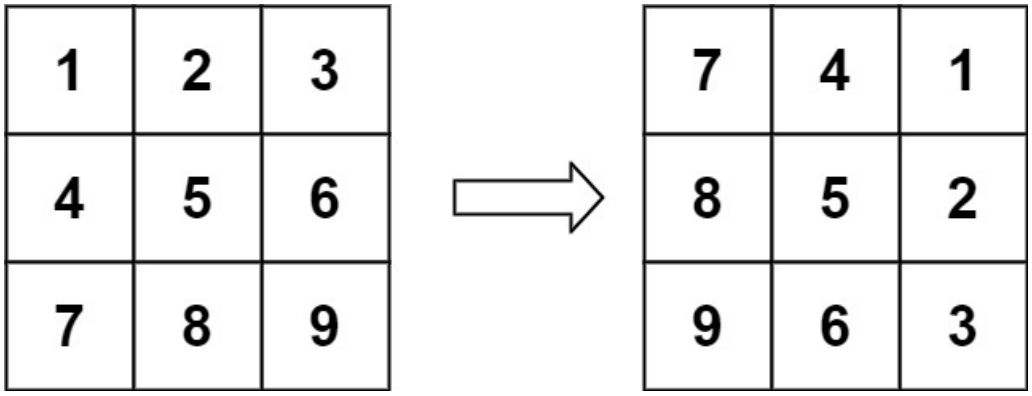
Topic - Matrix, Linked LIst

1 Problem statement: Rotate image (Medium)

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

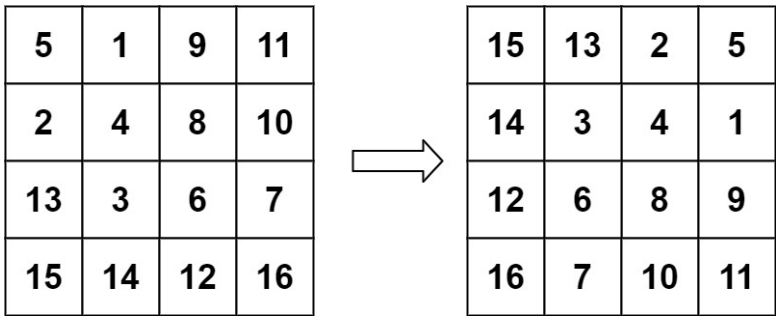
You have to rotate the image in-place, which means you have to modify the input 2D matrix directly. DO NOT allocate another 2D matrix and do the rotation.

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]]

Solutions :
Approach 1 - 2D array

Intuition:
Did a dry run of a test case. Figured out, we have to transpose the matrix and then reverse it.

Approach;
Transpose - swap a[i][j] with a[j][i]
Reverse - swap a[i][j] with a[i][n-j-1]
Created functions and called them

Complexity;
Time complexity:
O(n2) - nxn matrix

Space complexity:
O(1)- no additional space used

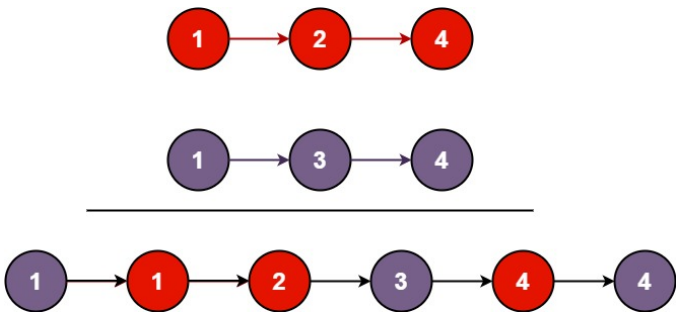
```
class Solution {
public:
    void transpose (vector<vector<int>> &matrix)
    {
        // swap mat[i][j] with mat[j][i]
        int n = matrix.size();
        int m = matrix[0].size();
        // n==m transpose exists only for square matrices, matrix given in
question nxn so square
        for (int i = 0 ; i< n ;i++)
        {
            // j begins from i and not 0
            for (int j = i ; j<n ; j++)
            {
                int temp = matrix[i][j];
                matrix[i][j]=matrix[j][i];
                matrix[j][i]=temp;
            }
        }
    }

    void reverse ( vector<vector<int>> &matrix)
    {
        int n = matrix.size();
```

```
        for (int i = 0 ; i<n ; i++)
        {
            // j goes till n/2 only and not till n
            for (int j = 0 ; j<n/2 ; j++)
            {
                int temp = matrix[i][j];
                matrix[i][j] = matrix[i][n-j-1];
                matrix[i][n-j-1]= temp;
            }
        }
    }

    void rotate(vector<vector<int>>& matrix) {

        // transpose and then reverse
        transpose(matrix);
        reverse(matrix);
    }
};
```

1 Problem statement: [Merge 2 sorted list](#) (Medium)

You are given the heads of two sorted linked lists list1 and list2.
Merge the two lists in a one sorted list. The list should be made by splicing together the nodes of the first two lists.
Return the head of the merged linked list.

Example 1:



Input: list1 = [1,2,4], list2 = [1,3,4]
Output: [1,1,2,3,4,4]

Example 2:
Input: list1 = [], list2 = []
Output: []

Example 3:
Input: list1 = [], list2 = [0]
Output: [0]

Solutions :
Approach 1 - Brute Force

1. Copy both lists to a vector.
2. Sort the vector.
3. Copy sorted vector to a linkedlist to get the sorted merged linkedlist.

Complexity:
Where length of list 1 and list 2 is n and m respectively,
Time complexity: O(nm*lognm)
Space complexity: O(n+m)

```
#include<vector>
#include<algorithm>
using namespace std

int copyList_toVector(vector<int> &v, ListNode *list){
    while(list){
        v.push_back(list->val);
        list=list->next;
    }
    return 0;
}

ListNode* copyVector_toList(vector<int> v){
    //we will need two ListNode pointers that point to the first node
    //Reason: one will be used to iterate over the list

    ListNode *head = NULL;
    ListNode *tail = NULL;

    //DONT FORGET TO USE NEW!
    //iterating over the vector
    for(auto it = v.begin(); it!=v.end(); it++){
        ListNode *temp = new ListNode(*it); //temp now contains the value
```

```
of current element in vector

        if(head==NULL){ //if the list is EMPTY
            head = temp;
            tail = temp;
        }
        else{ //list is NOT EMPTY
            tail->next = temp;
            tail = tail->next;
        }
    }

    return head;
}

class Solution {
public:
    ListNode* mergeTwoLists(ListNode* list1, ListNode* list2) {
        vector<int> v;
        //copy both lists to vector
        copyList_toVector(v,list1);
        copyList_toVector(v,list2);

        //sort the vector
        sort(v.begin(),v.end());

        //copy sorted vector to a list to get sorted list
        ListNode* sortedList = copyVector_toList(v);

        //return sorted list
        return sortedList;
    }
};
```

Solutions :
Approach 2 - optimized Approach (modify original array)

Step 1: choose a small element list first.
Step 2: loop throught both the loops and change their next pointer.
Time: O(n+m), Space: O(1).

```
if(list1 == NULL){
            return list2;
        }else if(list2 == NULL){
            return list1;
        }

        ListNode *head, *p;
        if(list1->val<= list2->val){
            head = list1;
            p = head;
            list1 = list1->next;
        }else{
            head = list2;
            p = head;
            list2 = list2->next;
        }

        while(list1 != NULL && list2 != NULL){
            if(list1->val <= list2->val){
                p->next= list1;
                p=p->next;
                list1 = list1->next;

            }else{
                p->next=list2;
                p=p->next;
                list2= list2->next;
            }
        }
        if(list1 != NULL){
            p->next = list1;
            p=p->next;
            list1 = list1->next;
        }else{
            p->next=list2;
            p=p->next;
            list2= list2->next;
        }

        return head;
```