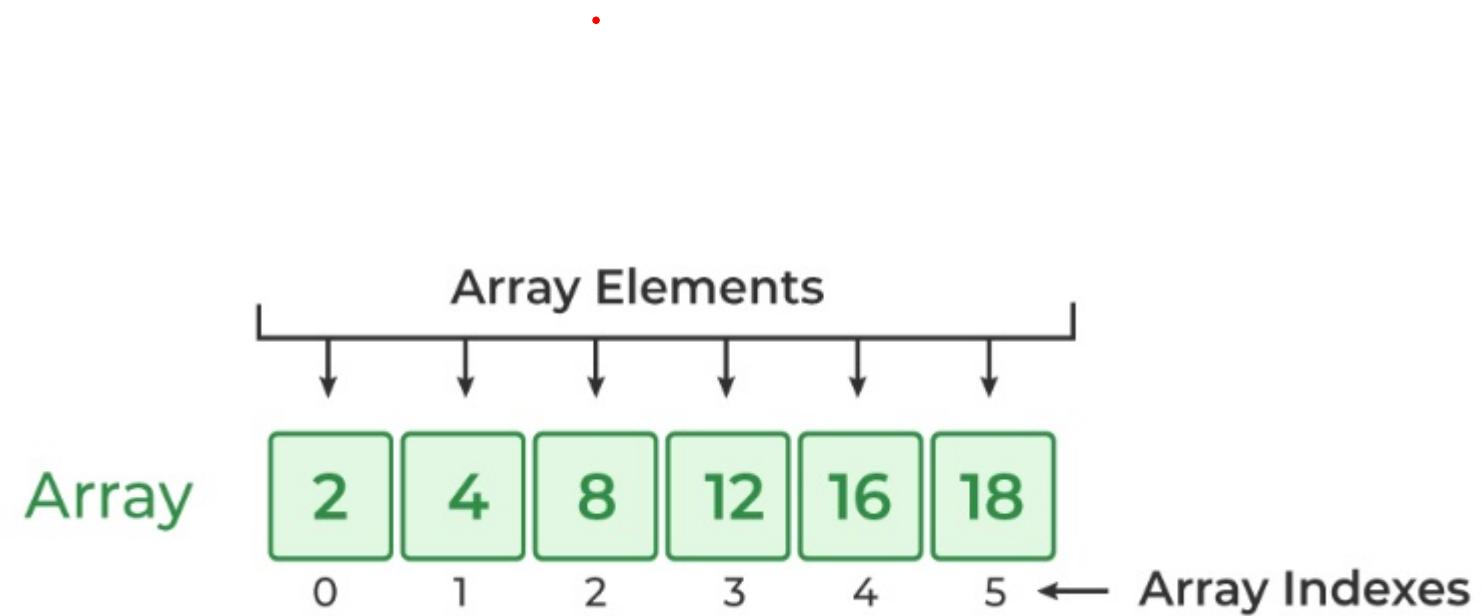


## Implementation of 1D and Multi-Dimension Arrays





## Problem 1: Insert Element at Index

### Problem Statement:

Write a function to insert an element into a given array at a specified index.

cpp

```
void insertAtIndex(int arr[], int &n, int value, int index);
```

### Input:

First line: Integer n (number of elements)

Second line: n integers

Third line: Integer value to insert

Fourth line: Integer index where value is to be inserted

### Output:

Print the array after insertion.

### Sample Input:

```
5
```

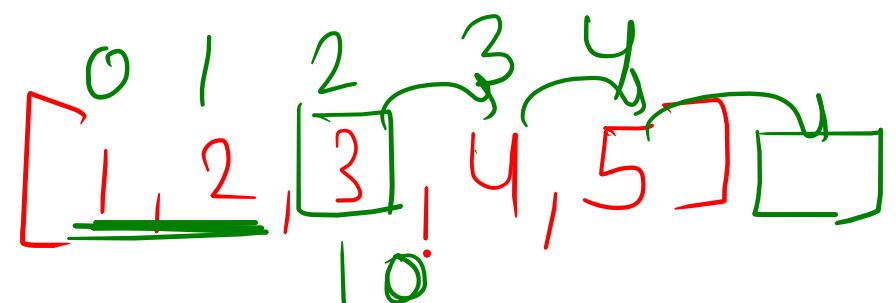
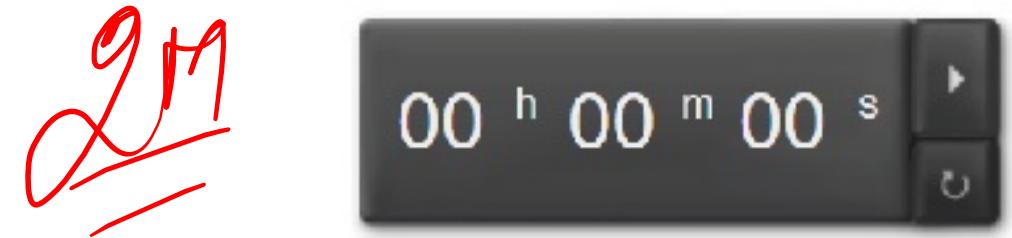
```
1 2 3 4 5
```

```
10
```

```
2
```

### Sample Output:

```
1 2 10 3 4 5
```

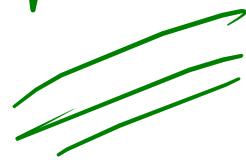


Value = 10

Index = 2

=

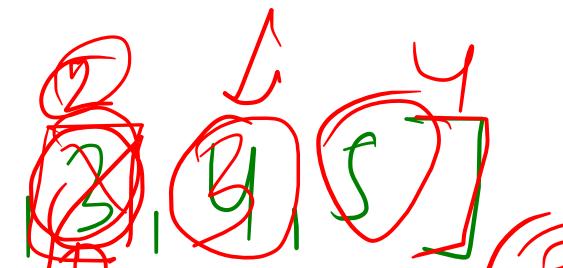
Approach ①



ptr  
Jump

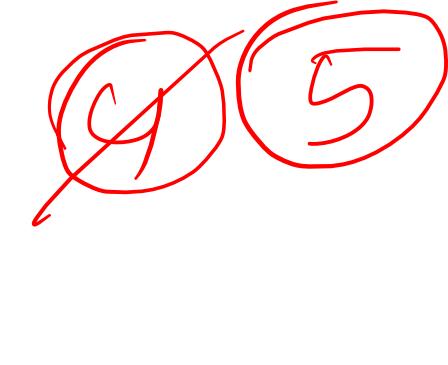


[1, 2]



loop②

arr.push (temp)



Approach  $\text{Arr} =$



Value = 10

Index = 2

Arr.push\_back(0)

array[Index] = value

sum += arr[Index]

## Problem 2: Delete Element at Index

### Problem Statement:

Write a function to delete the element at a specified index from an array.

cpp

```
void deleteAtIndex(int arr[], int &n, int index);
```

### Input:

First line: Integer n (number of elements)

Second line: n integers

Third line: Integer index to delete

### Output:

Print the array after deletion.

### Sample Input:

```
6
10 20 30 40 50 60
3
```

### Sample Output:

```
10 20 30 50 60
```

## Problem 3: Update Negative Values to Positive

### Problem Statement:

Write a function to update all negative elements in a given array by converting them to positive.

cpp

 Copy  Edit

```
void convertNegativesToPositive(int arr[], int n);
```

### Input:

First line: Integer n (number of elements)

Second line: n integers (can include negative values)

### Output:

Print the updated array where all negative values are converted to positive.

### Sample Input:

diff

 Copy  Edit

6

-10 5 -3 8 0 -2

### Sample Output:

10 5 3 8 0 2

## Problem 4: Find Second Largest Element

### Problem Statement:

Write a function to find the second largest element in a given array of distinct integers.

cpp

```
int findSecondLargest(int arr[], int n);
```

### Input:

First line: Integer n (number of elements)

Second line: n distinct integers

### Output:

Print the second largest element.

### Sample Input:

```
5  
25 10 60 45 30
```

### Sample Output:

```
45
```

$H \rightarrow \text{for } (n)$

$\quad ) \rightarrow [1, 2, 60, 90, 10] \rightarrow \underline{\max} \rightarrow \underline{\underline{90}}$

$L_2 \text{ for } ( ) \leq \underline{\underline{90}} \rightarrow \underline{\underline{60}}$

$\underline{\underline{2n}} \Rightarrow n + n \Rightarrow \underline{\underline{2n}} \Rightarrow O(n)$

## Problem 5: Move All Zeros to End

### Problem Statement:

Write a function to move all 0s in the array to the end while maintaining the order of other elements.

cpp

 Copy  Edit

```
void moveZerosToEnd(int arr[], int n);
```

### Input:

First line: Integer n (number of elements)

Second line: n integers

### Output:

Print the array after moving all 0s to the end.

### Sample Input:

```
7  
0 5 0 3 0 1 4
```

 Copy  Edit

### Sample Output:

```
5 3 1 4 0 0 0
```

 Copy  Edit

## Problem 6: Count Frequency of a Given Element

### Problem Statement:

Write a function to count how many times a given number appears in the array.

cpp

```
int countFrequency(int arr[], int n, int key);
```

### Input:

First line: Integer n (number of elements)

Second line: n integers

Third line: Integer key to search for

### Output:

Print the count of how many times the key appears.

### Sample Input:

```
7  
1 2 3 2 4 2 5  
2
```

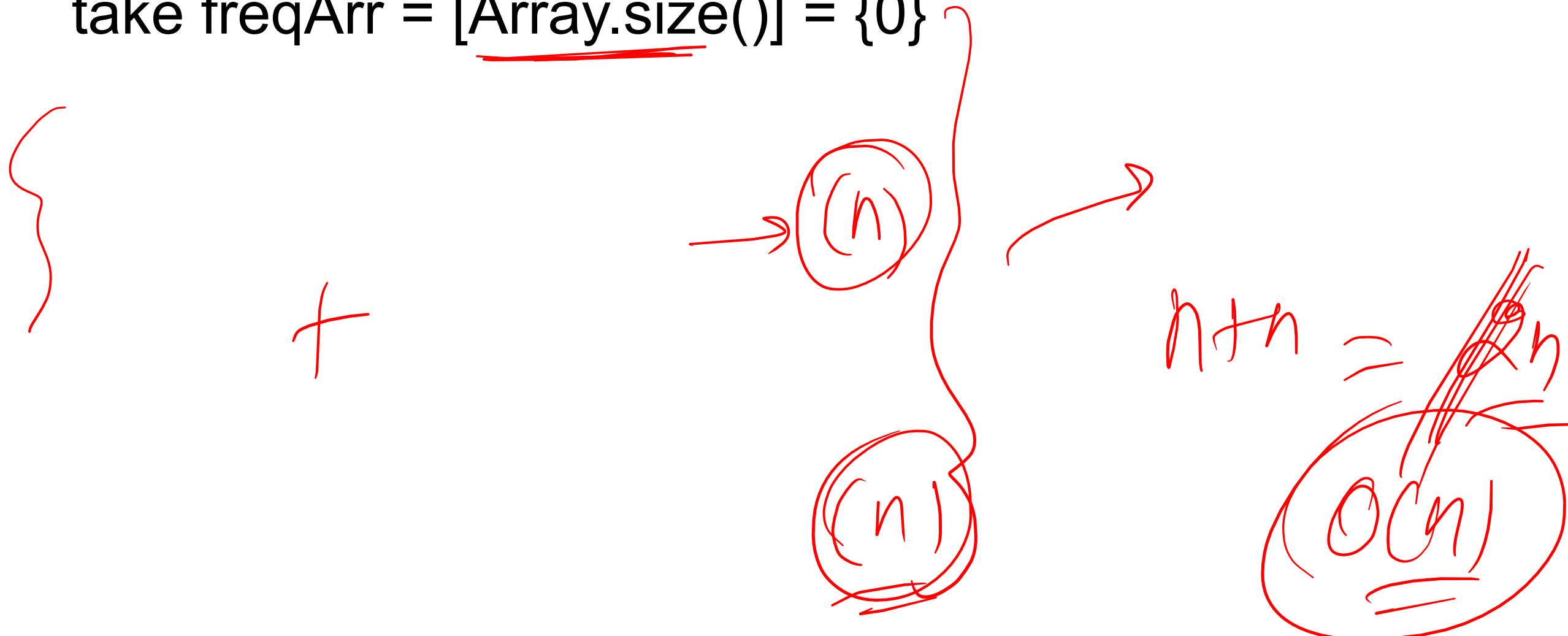
### Sample Output:

```
3
```

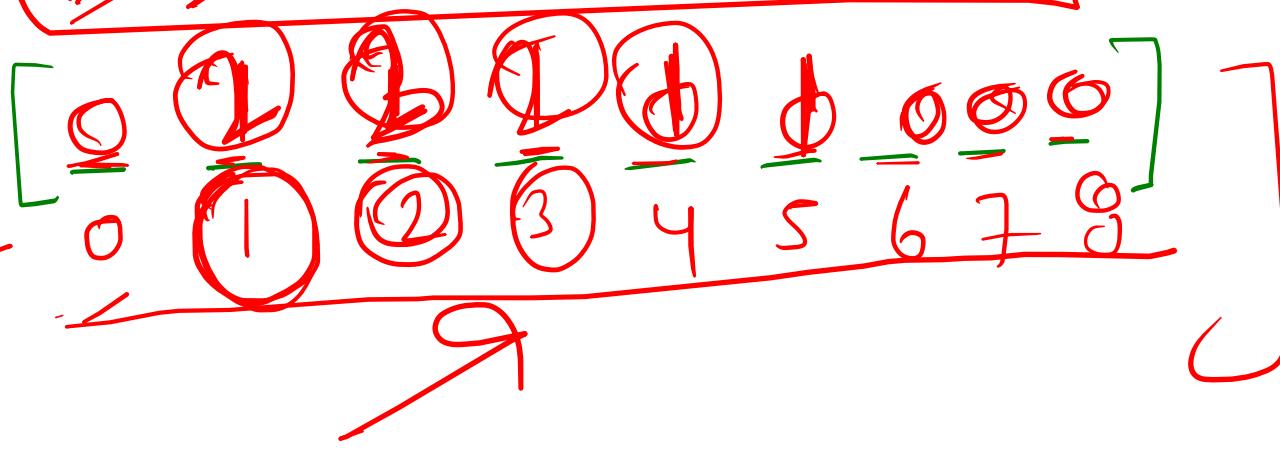
**Problem: Find the frequency of array elements**

Array = [1, 2, 3, 2, 1, 3, 4, 5, 2]

take freqArr = [Array.size()] = {0}



Array =  $[1, 2, 3, 2, 1, 3, 4, 5, 2]$  → 9 element

temp = [  ] →

Number

Approach





## Problem 7: Check if Array is Sorted

### Problem Statement:

Write a function to check if the given array is sorted in non-decreasing order.

cpp

```
bool isSorted(int arr[], int n);
```

### Input:

First line: Integer n (number of elements)

Second line: n integers

### Output:

Print "YES" if the array is sorted, otherwise print "NO".

### Sample Input:

```
5  
1 2 3 4 5
```

### Sample Output:

```
objectivec  
YES
```

Non-decreasing order  
Ascending order

## Problem 8: Rotate Array to the Right by One

### Problem Statement:

Write a function to rotate the array elements to the right by one position.

cpp

```
void rotateRightByOne(int arr[], int n);
```

### Input:

First line: Integer n (number of elements)

Second line: n integers

### Output:

Print the array after rotating it right by one position.

### Sample Input:

```
5  
10 20 30 40 50
```

### Sample Output:

```
50 10 20 30 40
```

## Problem 9: Find First Repeating Element

### Problem Statement:

Write a function to find and print the first repeating element in the array. If no element repeats, print -1.

cpp

 Copy  Edit

```
int findFirstRepeating(int arr[], int n);
```

### Input:

First line: Integer n (number of elements)

Second line: n integers

### Output:

Print the first element that appears more than once. If no element repeats, print -1.

### Sample Input:

```
6  
4 5 1 2 1 3
```

 Copy  Edit

### Sample Output:

```
1
```

 Copy  Edit

## Problem 10: Find Number of Positive, Negative, and Zero Elements

### Problem Statement:

Write a function to count how many positive, negative, and zero elements are in the array.

cpp

 Copy  Edit

```
void countPosNegZero(int arr[], int n, int &pos, int &neg, int &zero);
```

### Input:

First line: Integer n (number of elements)

Second line: n integers

### Output:

Print three space-separated integers: count of positive, negative, and zero values.

### Sample Input:

diff

 Copy  Edit

7

-3 0 2 -1 0 4 -2

### Sample Output:

 Copy  Edit

2 3 2

## Problem 11: Frequency of Each Element

### Problem Statement:

Write a function to find and print the frequency of each unique element in the given array.

Each element and its frequency should be printed only once.

cpp

 Copy  Edit

```
void printFrequencies(int arr[], int n);
```

### Input:

First line: Integer `n` (number of elements)

Second line: `n` space-separated integers

### Output:

Print each element and its frequency in the format:

`element: frequency`

Each element should be printed only once, in the order of their first appearance.

### Sample Input:

 Copy  Edit

```
7  
1 3 5 3 2 3 1
```

### Sample Output:

makefile

 Copy  Edit

```
1: 2  
3: 3  
5: 1  
2: 1
```

## Problem 12: Rotate Array by K Positions

### Problem Statement:

Write a function to rotate the array to the right by  $k$  positions. The rotation should wrap around the array.

cpp

 Copy  Edit

```
void rotateRightByK(int arr[], int n, int k);
```

### Input:

First line: Integer  $n$  (number of elements)

Second line:  $n$  integers

Third line: Integer  $k$  (number of positions to rotate)

### Output:

Print the array after rotating it to the right by  $k$  positions.

### Sample Input:

```
6  
1 2 3 4 5 6  
2
```

 Copy  Edit

### Sample Output:

```
5 6 1 2 3 4
```

 Copy  Edit

## Problem 13: Find Pair with Given Sum

### Problem Statement:

Write a function to find any one pair of elements in the array whose sum is equal to a given target value.

cpp

Copy Edit

```
bool findPairWithSum(int arr[], int n, int target);
```

### Input:

First line: Integer n (number of elements)

Second line: n integers

Third line: Integer target (sum to find)

### Output:

If a pair exists, print the two numbers separated by space.

If no such pair exists, print -1.

### Sample Input:

Copy Edit

```
6  
2 4 7 11 5 3  
9
```

### Sample Output:

Copy Edit

```
4 5
```

## Problem 14: Find Difference Between Max and Min

### Problem Statement:

Write a function to find the difference between the maximum and minimum element in the array.

cpp

 Copy  Edit

```
int maxMinDifference(int arr[], int n);
```

### Input:

First line: Integer n (number of elements)

Second line: n integers

### Output:

Print the difference (maximum - minimum)

### Sample Input:

```
5  
10 20 5 8 30
```

 Copy  Edit

### Sample Output:

```
25
```

 Copy  Edit

## Problem 15: Find Unique Elements (Appear Only Once)

### Problem Statement:

Write a function to print all elements that appear only once in the array.

cpp

 Copy  Edit

```
void printUniqueElements(int arr[], int n);
```

### Input:

First line: Integer n (number of elements)

Second line: n integers

### Output:

Print all unique elements (elements that appear only once), separated by space.

Order of output should be the same as original input.

### Sample Input:

 Copy  Edit

```
7  
1 2 2 3 4 4 5
```

### Sample Output:

 Copy  Edit

```
1 3 5
```

# 2D Dimension Array Impelmentation

## Problem 1: Print 2D Vector Elements

### Problem Statement:

Write a function to print all elements of a 2D vector in row-wise order.

cpp

```
void print2DVector(const vector<vector<int>> &matrix);
```

### Input:

First line: Two integers `rows` and `cols`

Next `rows` lines: Each line contains `cols` integers

### Output:

Print the elements row-wise, each row on a new line.

### Sample Input:

```
2 3
```

```
1 2 3
```

```
4 5 6
```

### Sample Output:

```
1 2 3
```

```
4 5 6
```

## Problem 2: Sum of All Elements in a 2D Vector

### Problem Statement:

Write a function to return the sum of all elements in a 2D vector.

cpp

```
int sum2DVector(const vector<vector<int>> &matrix);
```

### Input:

First line: Two integers `rows` and `cols`

Next `rows` lines: Each line contains `cols` integers

### Output:

Print the sum of all elements.

### Sample Input:

```
2 2
1 2
3 4
```

### Sample Output:

```
10
```

### Problem 3: Sum of Two Matrices

#### Problem Statement:

Write a function to add two 2D matrices of the same size and store the result in a third matrix. Then print the resulting matrix.

cpp

Copy Edit

```
vector<vector<int>> addMatrices(const vector<vector<int>> &matrix1, const vector<vector<int>> &matrix2)
```

#### Input:

First line: Two integers `rows` and `cols`

Next `rows` lines: Each line contains `cols` integers of `matrix1`

Next `rows` lines: Each line contains `cols` integers of `matrix2`

#### Output:

Print the resulting matrix after addition, row by row.

#### Sample Input:

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

matrix1  
matrix2

Copy Edit

#### Sample Output:

```
8 10 12  
5 6 7
```

## Problem 4: Row-wise and Column-wise Sum

### Problem Statement:

Write a function to calculate and print the sum of each row and each column in a 2D matrix.

cpp

Copy Edit

```
void rowColWiseSum(const vector<vector<int>> &matrix);
```

### Input:

First line: Two integers `rows` and `cols`

Next `rows` lines: Each line contains `cols` integers

### Output:

First, print the sum of each row in separate lines.

Then, print the sum of each column in one line, space-separated.

### Sample Input:

```
3 3
```

```
1 2 3
```

```
4 5 6
```

```
7 8 9
```

### Sample Output:

sql

Copy Edit

Row Sums:

6

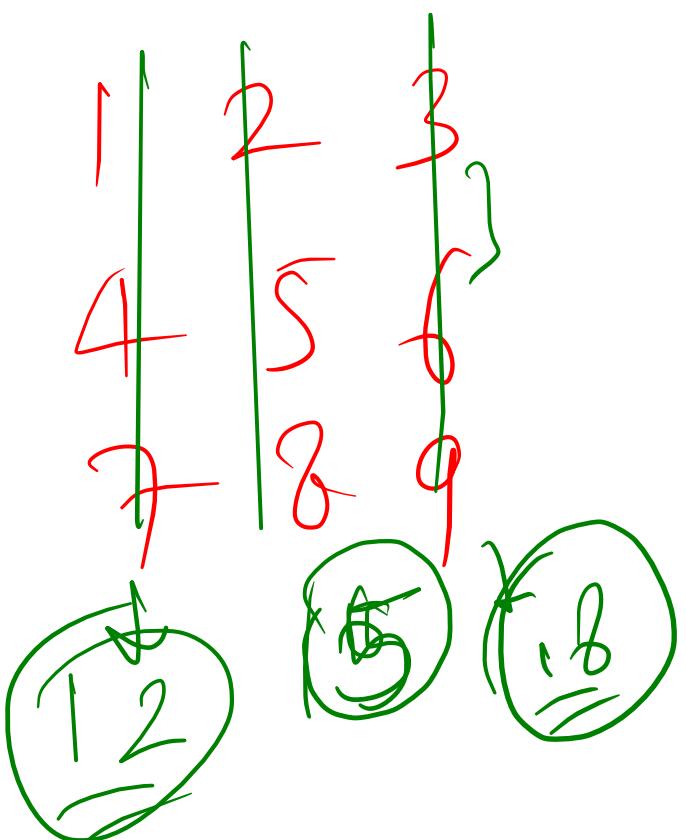
15

24

Column Sums:

12 15 18

1 2 3 → 6  
4 5 6 → 15  
7 8 9 → 24



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

Row wise

Column wise

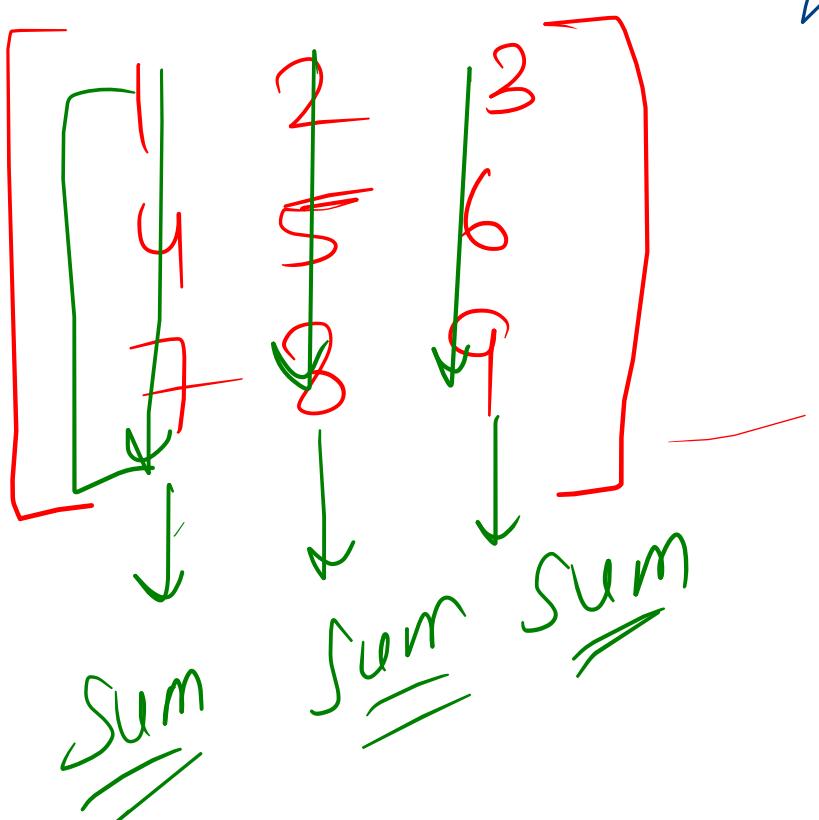
for Row to Row

for Column to Column

rowsum += Ans[i][j]

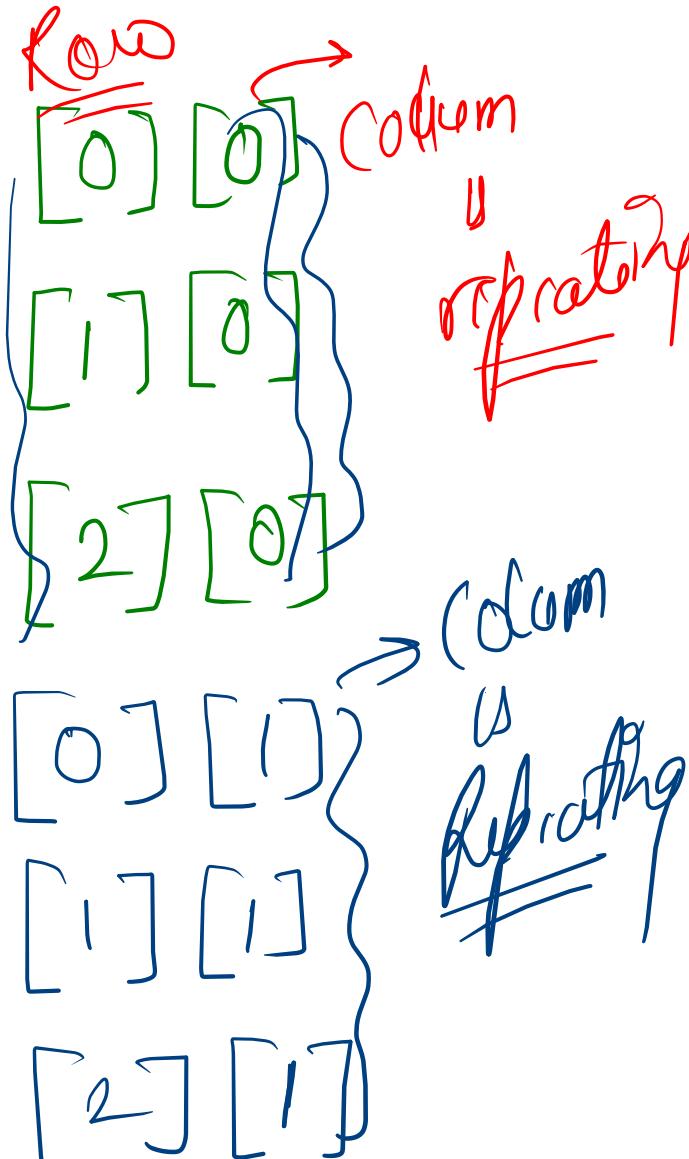
Column-wise sum

2D



for (Column to  $n$ )  
for (Row to  $m$ )

2  
5  
8



(Column)  
U  
Repeating

## Problem 5: Search an Element in 2D Vector

### Problem Statement:

Write a function to search for a target element in a 2D vector and print its position if found. If not found, print `-1 -1`.

cpp

Copy Edit

```
void searchElement(const vector<vector<int>> &matrix, int target);
```

### Input:

First line: Two integers `rows` and `cols`

Next `rows` lines: Each line contains `cols` integers

Next line: Integer `target`

### Output:

Print the row and column index of the first occurrence of the target.

If not found, print `-1 -1`.

### Sample Input:

Copy Edit

```
2 3
5 7 9
1 3 4
3
```

### Sample Output:

Copy Edit

```
1 1
```

## Problem 6: Normal Traversal of 2D Matrix

### Problem Statement:

Write a function to traverse a 2D matrix **in row-major order** and return all elements as a 1D list.

cpp

 Copy  Edit

```
vector<int> normalTraversal(const vector<vector<int>> &matrix);
```

### Input:

First line: Two integers `rows` and `cols`

Next `rows` lines: Each line contains `cols` integers

### Output:

Print all elements in row-wise order in a single line, space-separated.

### Sample Input:

 Copy  Edit

```
2 3  
1 2 3  
4 5 6
```

### Sample Output:

 Copy  Edit

```
1 2 3 4 5 6
```

Matrix

①

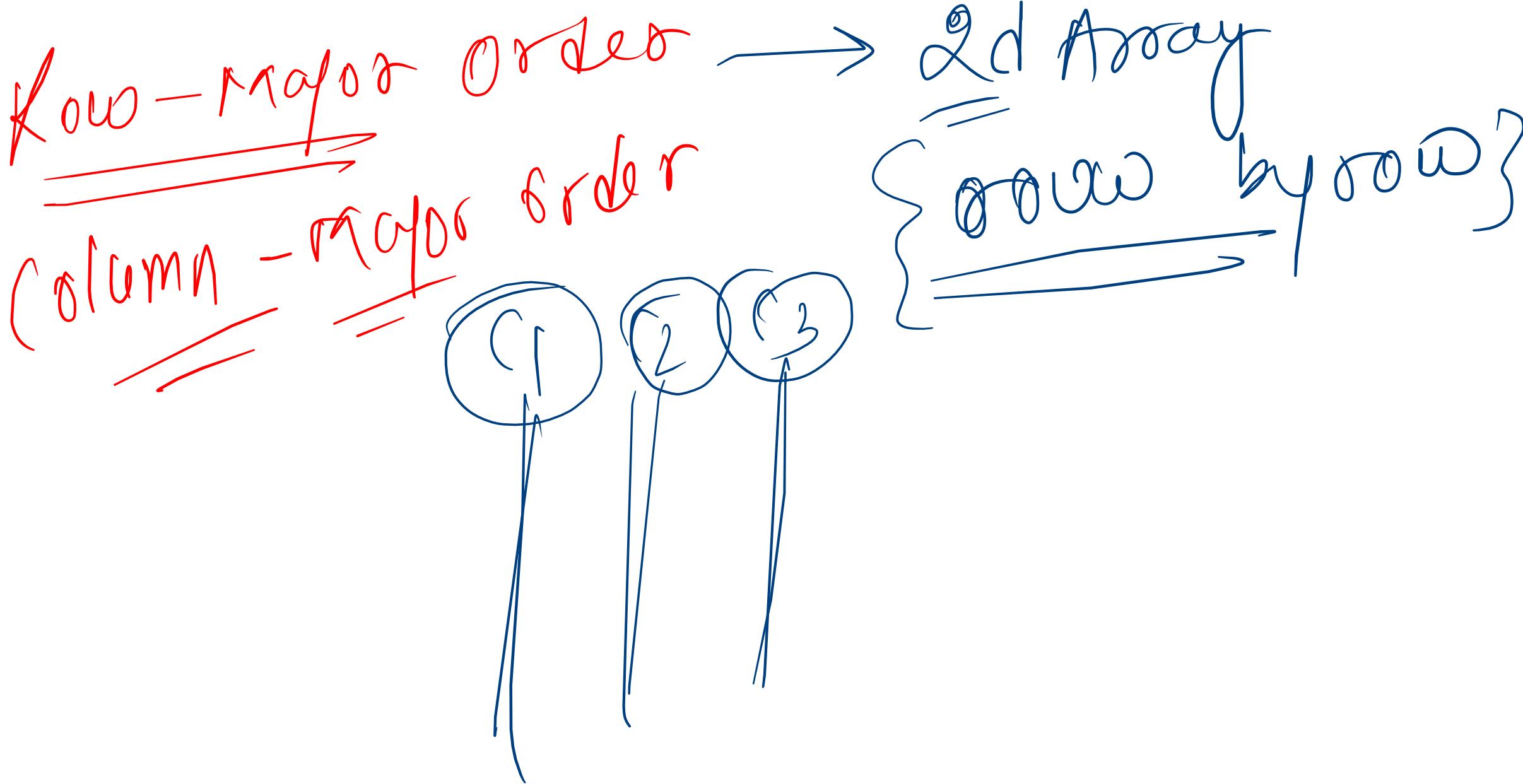
Row-major Order

Row by Row

②

Column-major order

Column by Column



## Problem 7: Transpose of a Matrix

### Problem Statement:

Write a function to return the transpose of a given 2D matrix.

cpp

Copy Edit

```
vector<vector<int>> transposeMatrix(const vector<vector<int>> &matrix);
```

### Input:

First line: Two integers `rows` and `cols`

Next `rows` lines: Each line contains `cols` integers

### Output:

Print the transposed matrix.

### Sample Input:

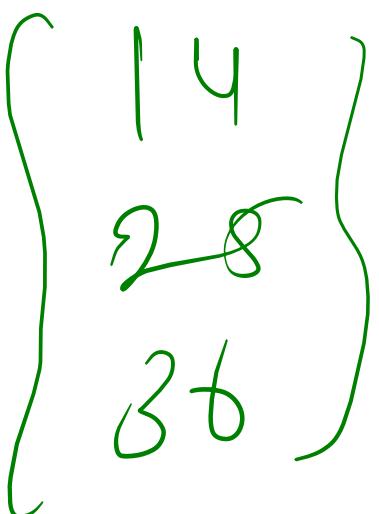
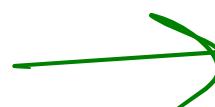
```
2 3  
1 2 3  
4 5 6
```

Copy Edit

### Sample Output:

```
1 4  
2 5  
3 6
```

Copy Edit

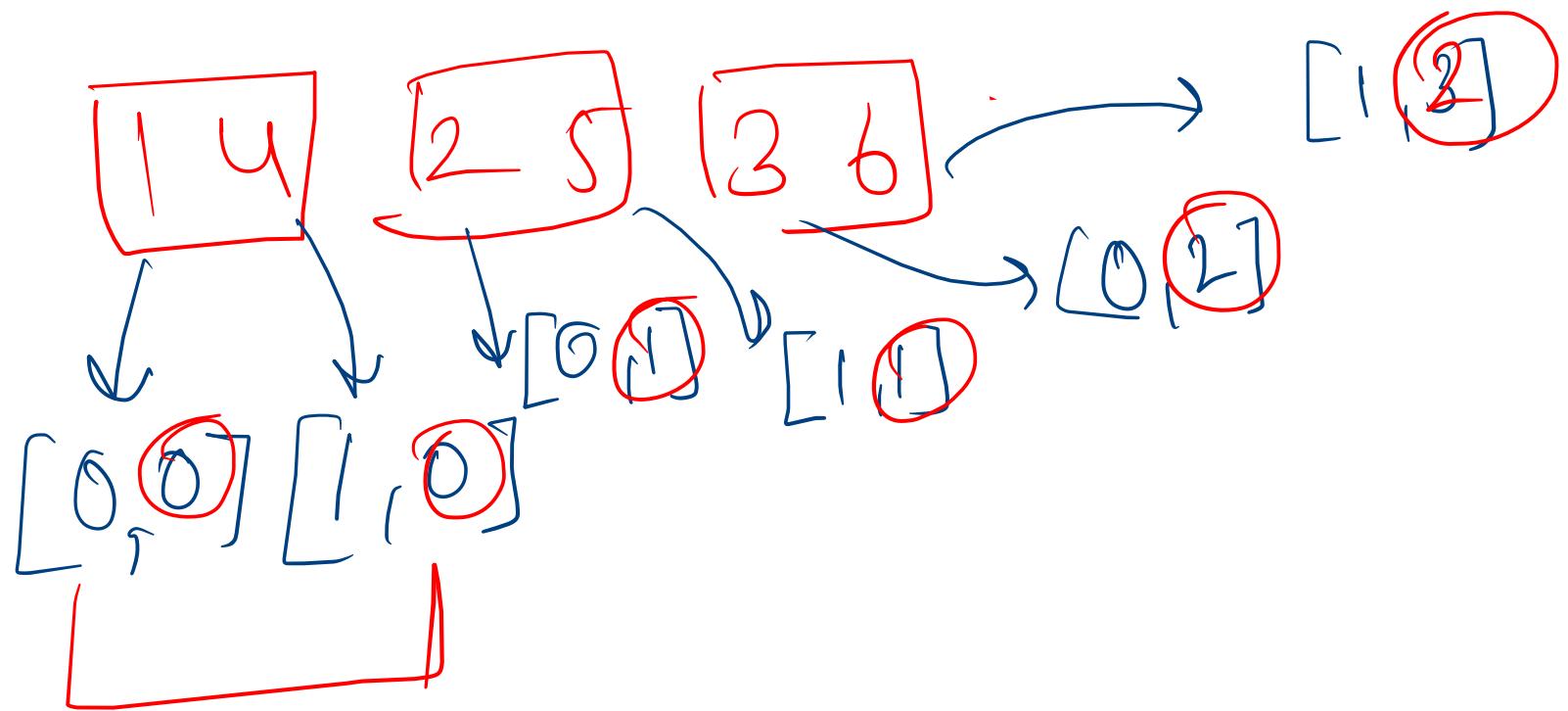


for (column)  
for (row)

1 2 3  
4 5 6

→ Column to Row  
ao

Row to column



## Problem 8: Rotate Matrix by 90 Degrees Clockwise

### Problem Statement:

Write a function to rotate a square matrix ( $n \times n$ ) by 90 degrees clockwise.

cpp

Copy Edit

```
void rotate90Clockwise(vector<vector<int>> &matrix);
```

### Input:

First line: Integer  $n$  (rows and columns, square matrix)

Next  $n$  lines: Each line contains  $n$  integers

### Output:

Print the matrix after rotating it 90 degrees clockwise.

### Sample Input:

Copy Edit

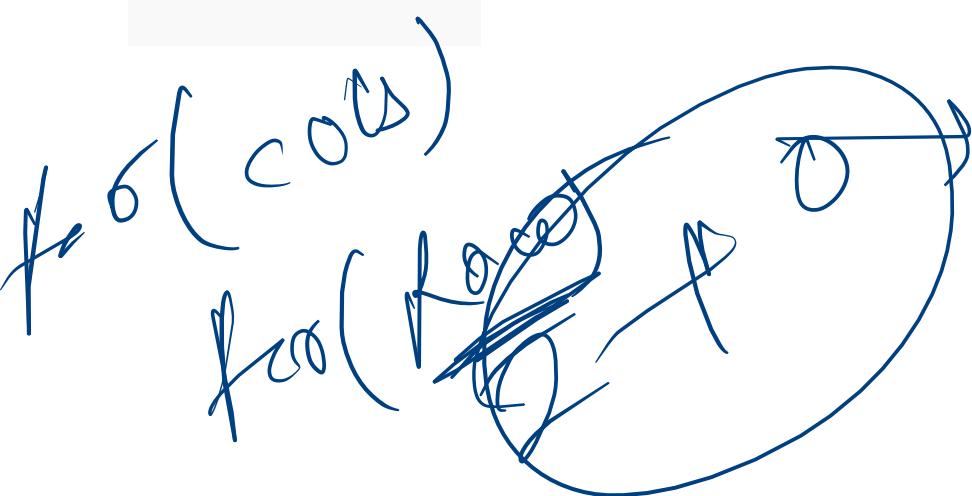
```
3  
1 2 3  
4 5 6  
7 8 9
```

### Sample Output:

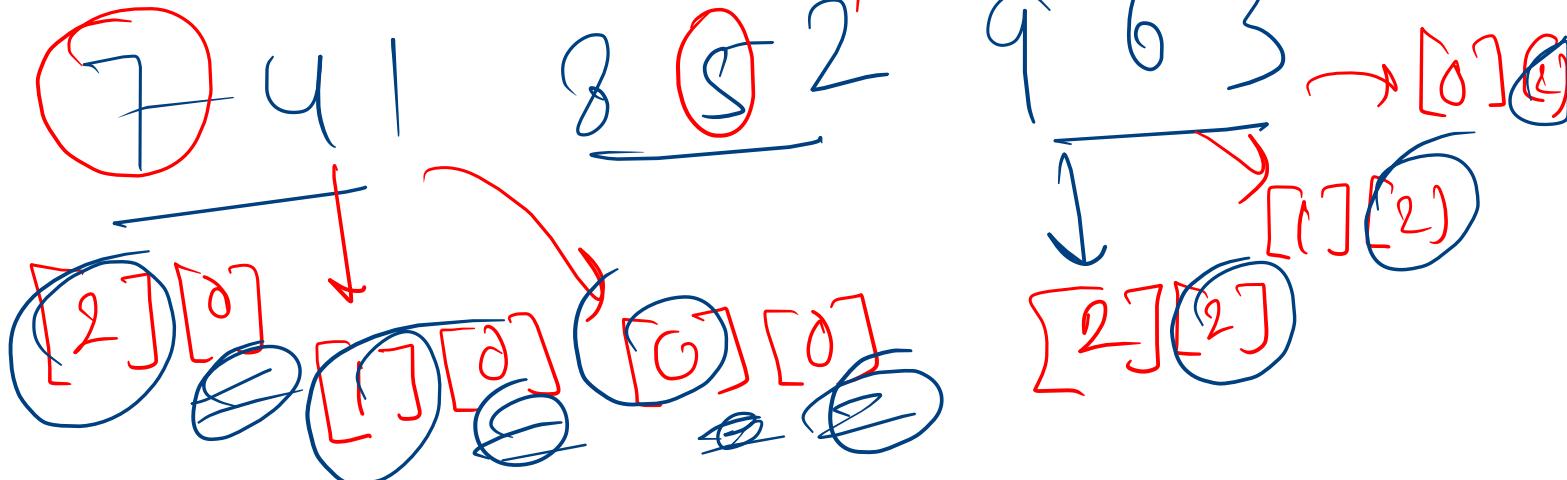
Copy Edit

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

0	1	2	3
1	4	5	6
2	7	8	9



}  
 {  
 7 4 1  
 8 5 2  
 9 6 3



## Problem 9: Print Diagonal Elements

### Problem Statement:

Write a function to print the **main diagonal** and **secondary diagonal** elements of a square matrix.

cpp

 Copy  Edit

```
void printDiagonals(const vector<vector<int>> &matrix);
```

### Input:

First line: Integer `n` (for  $n \times n$  matrix)

Next `n` lines: Each line contains `n` integers

### Output:

First line: Main diagonal elements

Second line: Secondary diagonal elements

### Sample Input:

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

 Copy  Edit

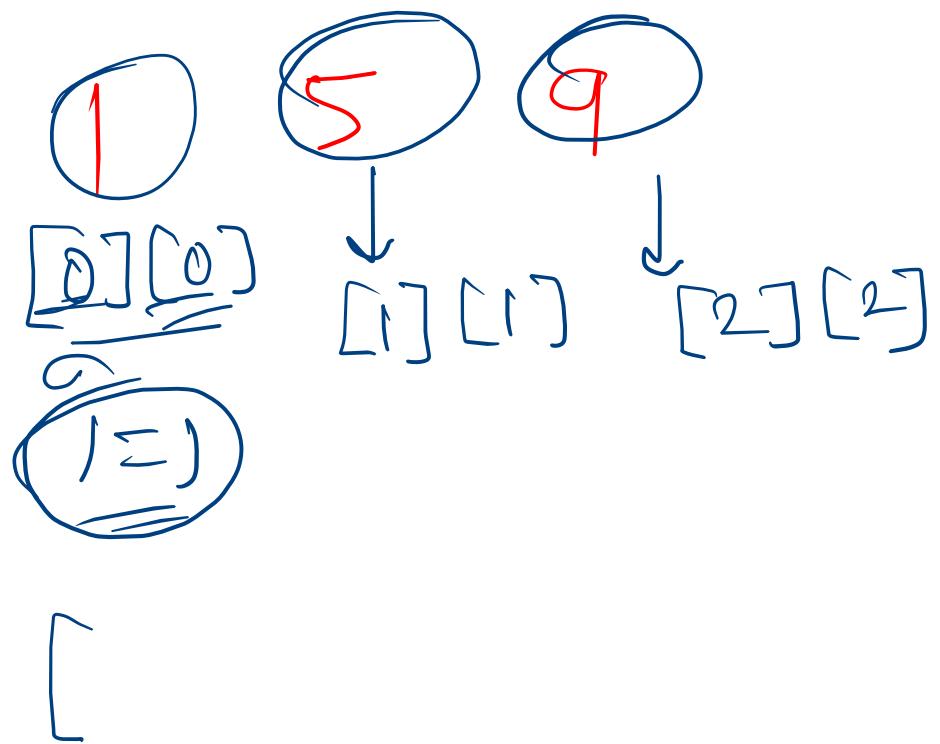
### Sample Output:

```
1 5 9
3 5 7
```

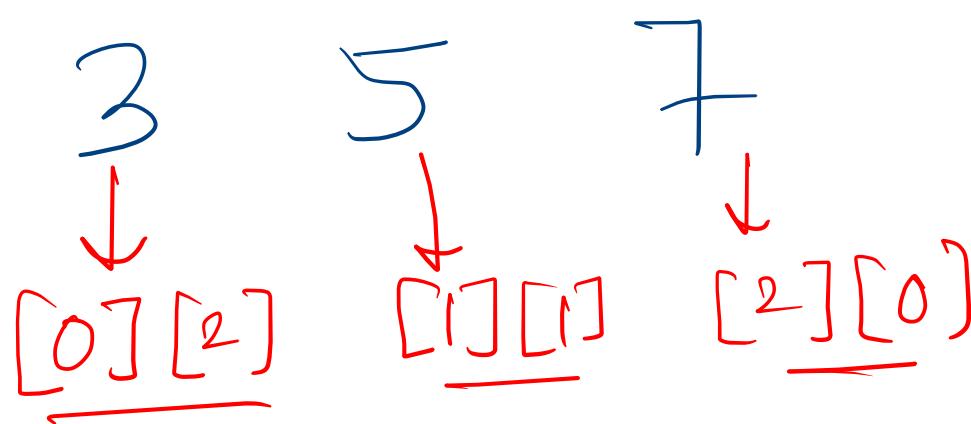
 Copy  Edit

1 2 3  
4 5 6  
7 8 9

right →

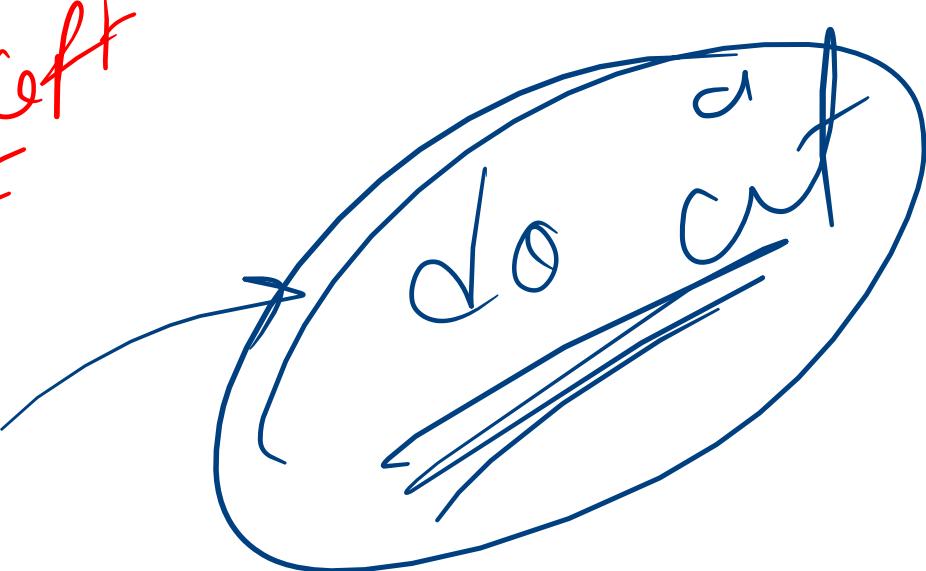


Left →



1	2	3
4	5	6
7	8	9

*cross left*



## Problem 10: N Traversal of Matrix

### Problem Statement:

Write a function to print the elements of a square matrix in an **N** pattern traversal:

- Print the first column from top to bottom
- Then print the diagonal from bottom-left to top-right
- Then print the last column from top to bottom

cpp

Copy Edit

```
void nTraversal(const vector<vector<int>> &matrix);
```

### Input:

First line: Integer `n` (for an `n x n` matrix)

Next `n` lines: Each line contains `n` integers

### Output:

Print the elements in N pattern, space-separated.

### Sample Input:

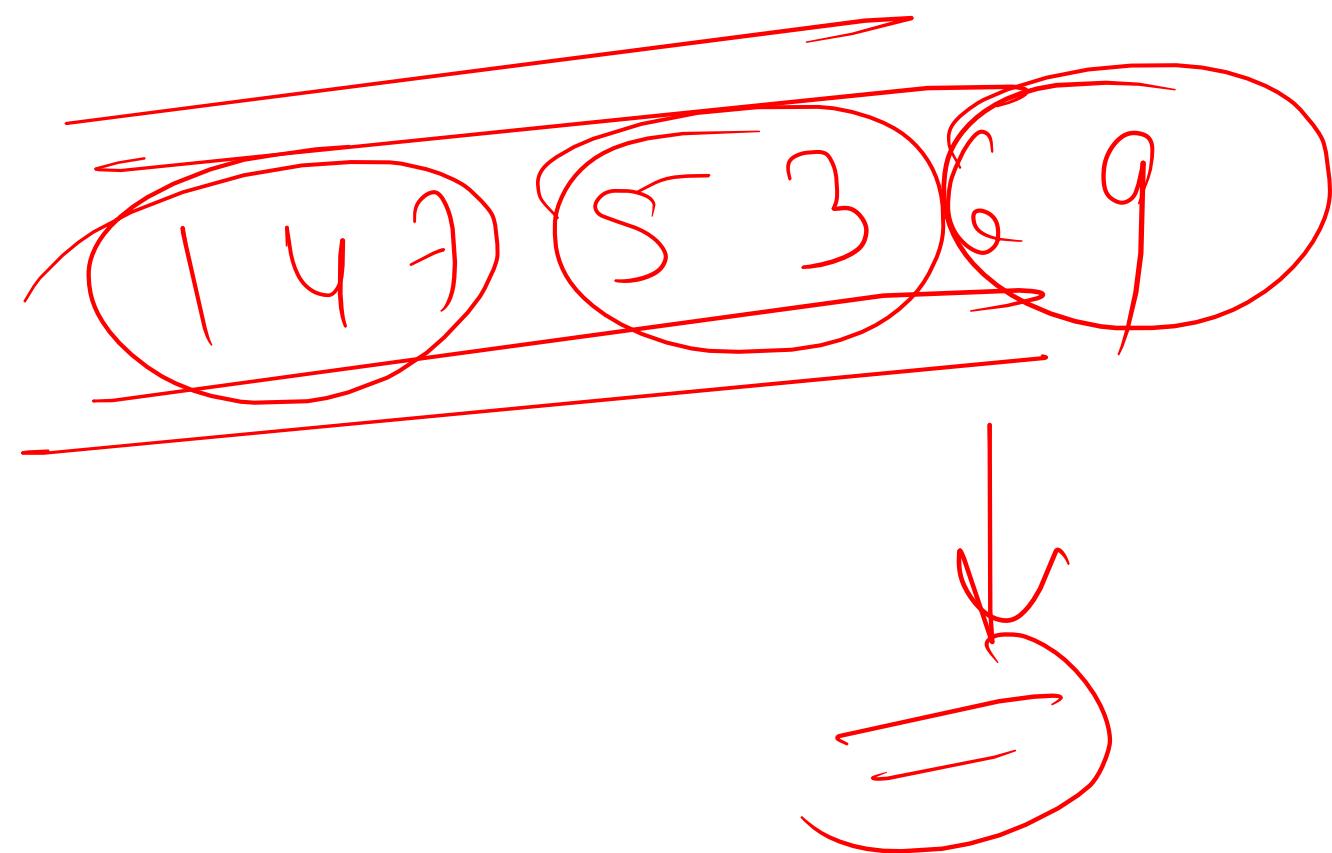
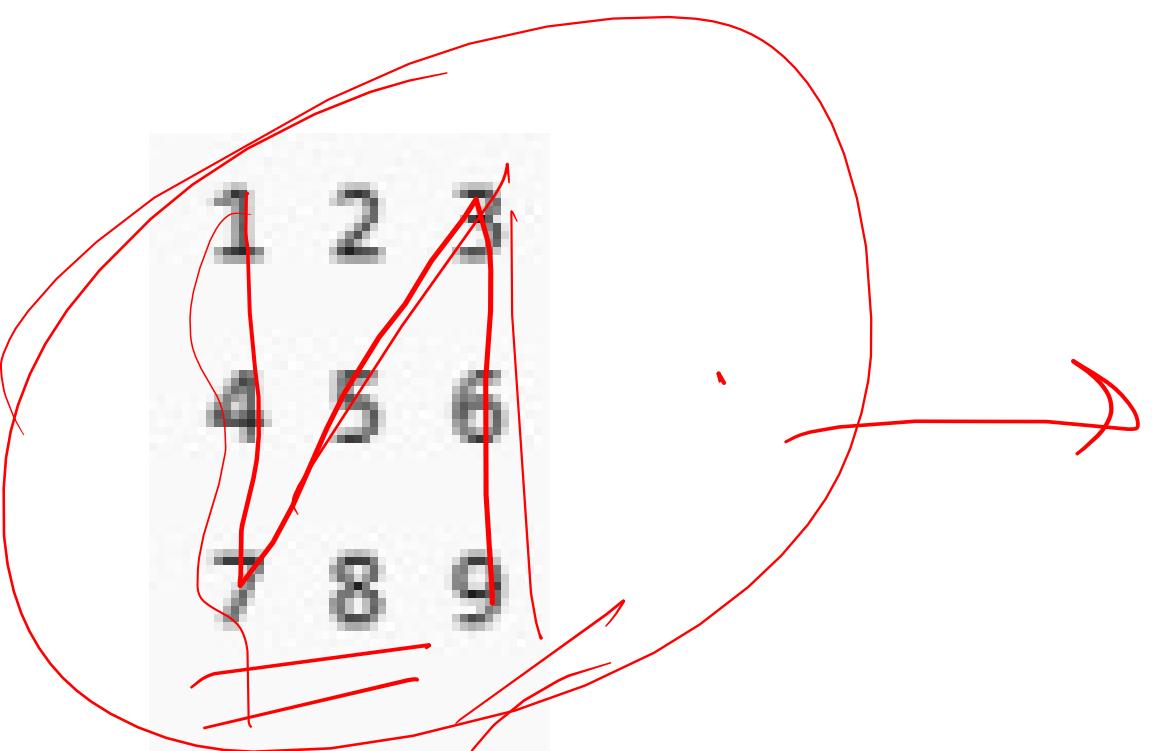
```
3
1 2 3
4 5 6
7 8 9
```

Copy Edit

### Sample Output:

```
1 4 7 5 3 6 9
```

Copy Edit



## Updated Problem 11: Spiral Traversal (Anti-clockwise)

### Problem Statement:

Write a function to traverse a 2D matrix in **anti-clockwise spiral order**, starting from the top-left corner and moving **down first**.

cpp

Copy Edit

```
vector<int> spiralTraversalAntiClockwise(const vector<vector<int>> &matrix);
```

### Input:

First line: Two integers `rows` and `cols`

Next `rows` lines: Each line contains `cols` integers

### Output:

Print the elements in anti-clockwise spiral order, space-separated.

---

### Sample Input:

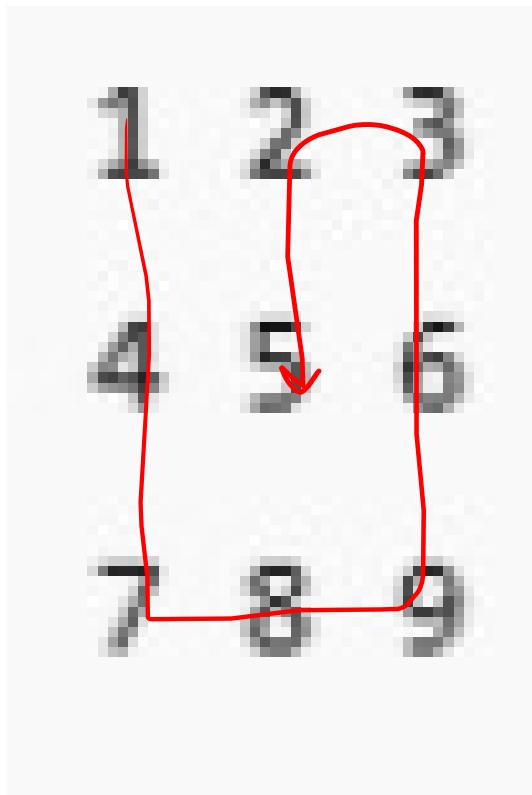
Copy Edit

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

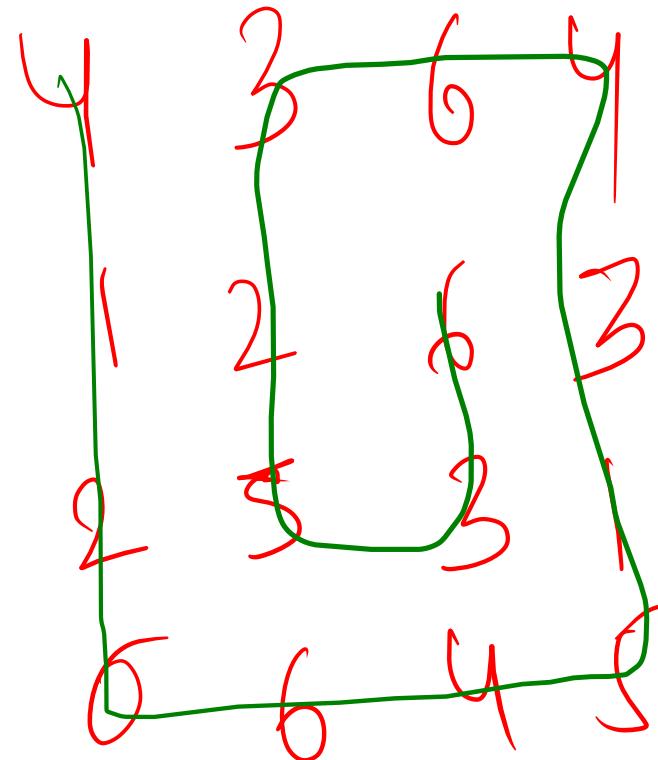
### Sample Output:

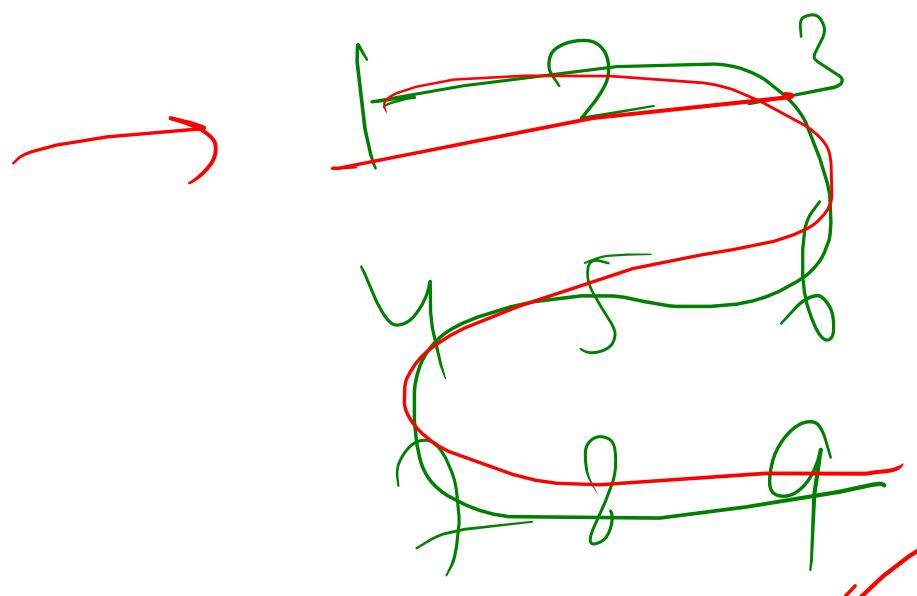
Copy Edit

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

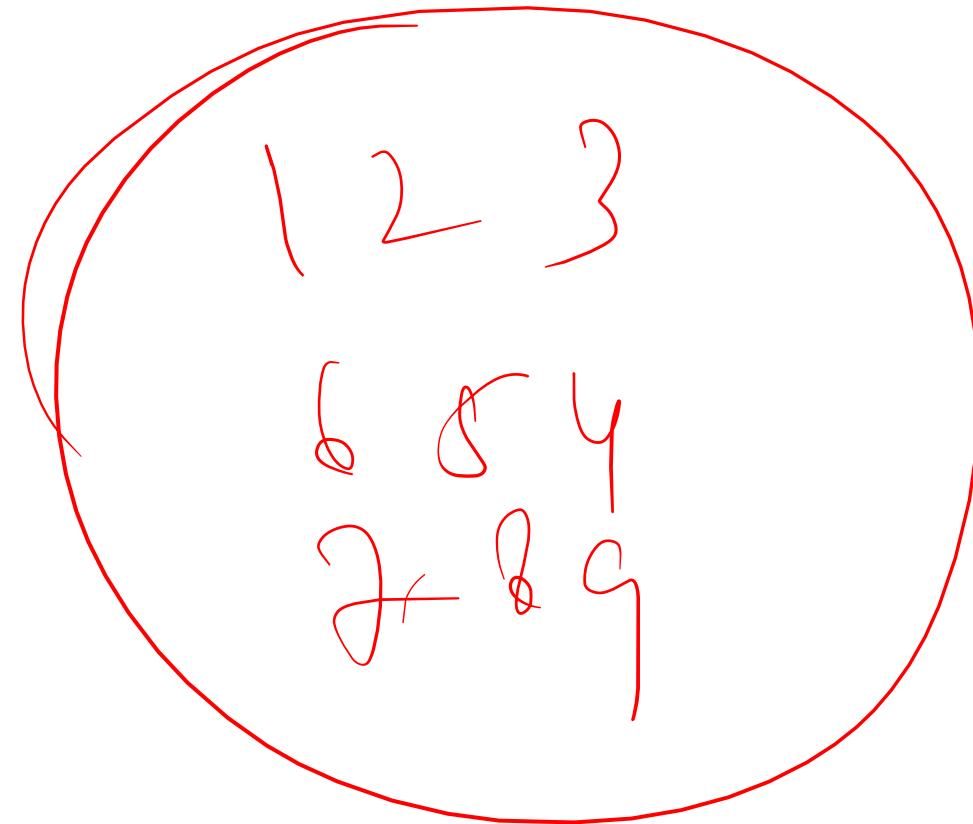
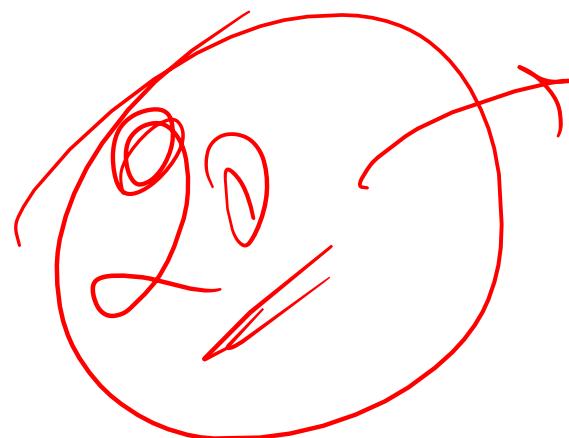


3x3





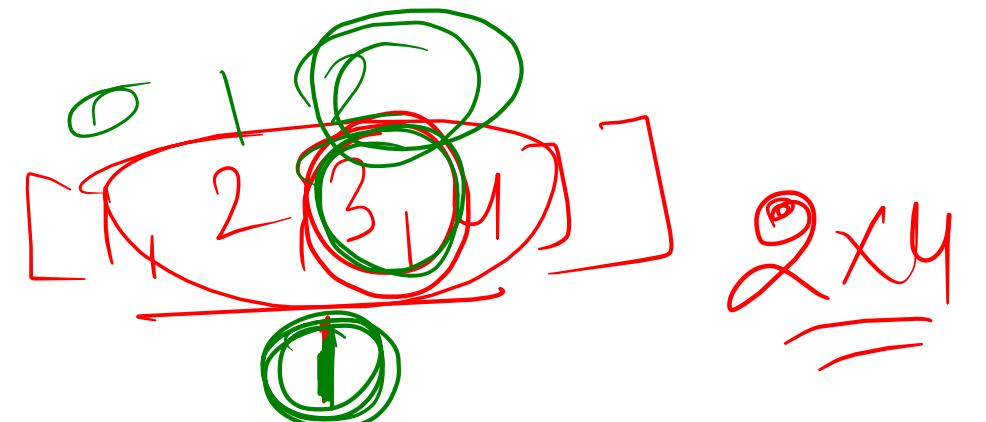
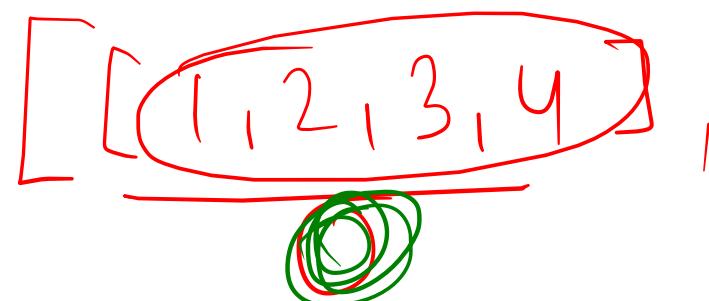
4 days



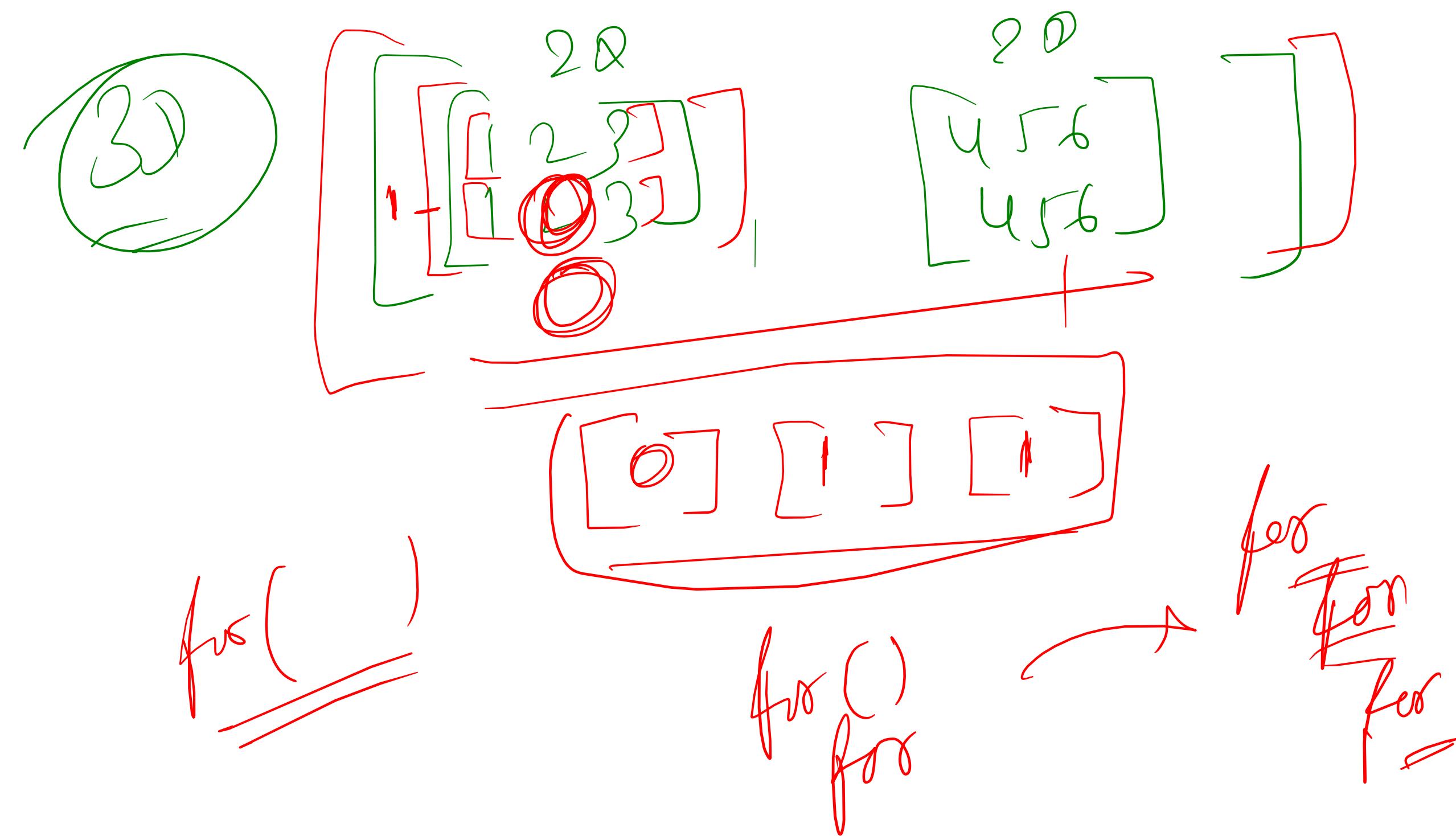
10



20



$$\frac{[feu] [cui]}{[1] (3)}$$



## Problem 12: Print Matrix in Zig-Zag Order

Problem Statement:

Write a function to print a 2D matrix in zig-zag row-wise order:

- Even-indexed rows (0, 2, ...) → left to right
- Odd-indexed rows (1, 3, ...) → right to left

cpp

Copy Edit

```
void zigZagTraversal(const vector<vector<int>> &matrix);
```

Input:

First line: Two integers `rows` and `cols`

Next `rows` lines: Each line contains `cols` integers

Output:

Print the zig-zag traversal elements in a single line.

Sample Input:

```
3 4
1 2 3 4
5 6 7 8
9 10 11 12
```

Sample Output:

```
1 2 3 4 8 7 6 5 9 10 11 12
```

## Problem 13: Boundary Traversal of Matrix

### Problem Statement:

Write a function to print the **boundary elements** of a matrix in a clockwise manner, starting from the top-left corner.

cpp

Copy Edit

```
void boundaryTraversal(const vector<vector<int>> &matrix);
```

### Input:

First line: Two integers `rows` and `cols`

Next `rows` lines: Each line contains `cols` integers

### Output:

Print the boundary elements in clockwise order.

### Sample Input:

```
4 4
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
```

Copy Edit

### Sample Output:

```
1 2 3 4 8 12 16 15 14 13 9 5
```

Copy Edit





**End**