Transpose

$$A = \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 0 & -1 \\ 1 & 3 & 2 \\ 2 & 6 & 0 \end{array} \qquad A^T = \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 0 & 3 & 6 \\ 1 & -1 & 2 & 0 \end{array}$$

$$A = \begin{bmatrix} 1 & 3 & 5 \\ 6 & -1 & 0 \end{bmatrix} \qquad A^T = \begin{bmatrix} 1 & 6 \\ 3 & -1 \\ 5 & 0 \end{bmatrix}$$

$$A[0][0] = A^T[0][0]$$
$$A[0][1] = A^T[1][0]$$
$$A[0][2] = A^T[2][0]$$
$$A[1][0] = A^T[0][1]$$
$$A[1][1] = A^T[1][1]$$
$$A[1][2] = A^T[2][1]$$

$$i, j = j^T, i^T$$

```
int[][] getTranspose(int[][] A) {

    int N = A.length;
    int M = A[0].length;
    int[][] B = new int[M][N];
```

```
for(int row=0; row<N; row++){
    for(int col=0; col<M; col++){
        B[col][row] = A[row][col];
    }
}

return B;
}
```

Identity

$x + 0 = x$

0 is additive identity

$x * 1 = x$

1 is multiplicative identity

$x / 1 = x$

1 is divisive identity

$A + 0 = A$

0 is additive identity for matrices

$A * I = A$

I is multiplicative identity for matrix

Properties

⇒ Square matrix [rows = cols]

⇒ All main diagonal values are 1

⇒ All non-main diagonal values are 0

$$I_3 = \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 2 & 0 & 0 & 1 \end{array} \qquad I_2 = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

$$I_1 = [1]$$

```
int[][] identity(int N) {
    // return identity matrix of size N
    int[][] I = new int[N][N];
    for(int row = 0; row < N; row++){
        for(int col = 0; col < N; col++){
            if( row == col){
                I[row][col] = 1;
            } else {
                I[row][col] = 0;
            }
        }
    }
    return I;
}
```

How many cells visited ⇒ $N * N$

```
int[][] identity(int N) {
    // return identity matrix of size N
    int[][] I = new int[N][N];
    for(int row = 0; row < N; row++){
        I[row][row] = 1;
    }
    return I;
}
```

How many cells visited ⇒ N

Break : 10:00 pm

2D ArrayList

↳ Arraylist of array list

```
AL<AL<Integer>> list = new AL<AL<Integer>>();
```

Add

```
AL<Integer> l1 = new AL<Integer>();
l1.add(1);
l1.add(-2);
list.add(l1);
```

list:

l1: 1

: 1, -2

list : [1, -2]

```
AL<Integer> l2 = new AL<Integer>();
  l2.add(10);
  l2 add (-2);
  l2. add (0);
List. add (l2);
```

l2: 10
: 10, -2
: 10, -2, 0

list: [1, -2]
[10, -2, 0]

```
AL<Integer> l3 = new AL<Integer>();
  l3. add(5);
List. add (l3);
```

l3: 5

list: [1, -2]
[10, -2, 0]
[5]

Get

list.get (1); ⇒ [10, -2, 0]

list.get(0).get(1); ⇒ -2

Set

```
AL<Integer> l4 = new AL<>();
  l4.add(-5);
  l4. add(-6);

list. set(2, l4);
```

list: [1, -2]
[10, 3, 0]
[-5, -6]

list.get(1).set(1, 3);

Size

count of lists ⇒ list.size()

length of list at pos 1 ⇒ list.get(1).size()

Code : https://www.interviewbit.com/snippet/d61e440a4776306f1a16/