Transpose

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

$$A = \begin{array}{c|ccc} & 0 & 1 & 2 \\ \hline 0 & 1 & 3 & 5 \\ 1 & 6 & -1 & 0 \end{array} \qquad A^T = \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 1 & 6 \\ 1 & 3 & -1 \\ 2 & 5 & 0 \end{array}$$

$$\begin{array}{ll} \quad i \quad j & \quad i^T \quad j^T \\ A[0][0] = & A^T[0][0] \\ A[0][1] = & A^T[1][0] \\ A[0][2] = & A^T[2][0] \\ A[1][0] = & A^T[0][1] \\ A[1][1] = & A^T[1][1] \\ A[1][2] = & A^T[2][1] \end{array}$$

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

```
int[][] getTransposе (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 → Additive identity for numbers

$x * 1 = x$

1 → Multiplicative identity

$x / 1 = x$

$A + O = A$

$A * I = A$

Properties

- Square matrix [rows = cols]
- All diagonal values are 1
- All non-diagonal values are 0

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

$$I_2 = \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 1 & 0 \\ 1 & 0 & 1 \end{array}$$

$$I_1 = \begin{array}{c|c} & 0 \\ \hline 0 & 1 \end{array}$$

```
int [][]   identity ( int  N) {

    int[][]  mat =  new  int [N][N];
    for(int row=0; row<N; row++){
        for(int col=0; col<N; col++) {
            if (row == col) {
                mat[row][col]= 1;
            }
            else {
                mat [row][col] = 0;
            }
        }
    }
    return mat;
}
```

N * N cells visited

```
int [][] identity ( int N) {

    int[][] mat = new int [N][N];
    for(int row=0; row<N; row++){
        mat[row][row] = 1;
    }
    return mat;
}
```

N cells visited

Break : 9:55pm

2D ArrayList

$\hookrightarrow$ ArrayList of Arraylists

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

Add

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

list :

list.add (l1);

l1: 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, -2, 0

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

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

l3: 5

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

Get

list.get(1) ⇒ [10, -2, 0]

list.get(0) .get(1) ⇒ -2

Set

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

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

```
l4: -5, 6
```

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

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

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

Size

```
list.size() => 3
list.get(1).size() => 3
```