DAY 30

3 October, 2021

DSA

2-D ARRAY

Agenda for Today:-

1. → 2d Array Introduction → CW
2. → Matrix multiplication → HW
3. → Wave Display → CW
4. → Spiral Display → HW
5. → Exit point of Matrix → HW
6. → Rotate by 90. → HW

2-D array के बारे में 2 तरीको से बात करेंगे

1. Usage of 2-D array.
2. Memory used by 2-D array.

1. Usage of 2-D array.

Matrix की information अगर हमें चाहिए तो हमे 2-Darray बनाना पड़ता है।

int [][] arr ; → ये है 2-d array का नाम

Declare हुआ है 2-Darray.

int [][] varr = new int [3][4] ;

Define भी होगया है अब

फिर no. of columns लिखे जाते हैं।

पहले no. of rows लिखी जाती हैं

```
int[][] arr = new int [3][4];
```

	0	1	2	3
0	[0,0]	[0,1]	[0,2]	[0,3]
1	[1,0]	[1,1]	[1,2]	[1,3]
2	[2,0]	[2,1]	[2,2]	[2,3]

इस line से एक ऐसा array बनगया है।

ये spot है [2][0] मतलब 2nd row का 0th column.

अब इस 2-D array में values डालते हैं फिर loop लगाके print करेंगे।

```
arr[0][0] = 11;
arr[0][1] = 12
arr[0][2] = 13
arr[0][3] = 14

arr[1][0] = 21
arr[1][1] = 22
arr[1][2] = 23
arr[1][3] = 24

arr[2][0] = 31
arr[2][1] = 32
arr[2][2] = 33
arr[2][3] = 34
```

values डालने पर ये बनगया

11	12	13	14
21	22	23	24
31	32	33	34

i	j	input at arr[i][j]
0	0	arr[0][0]
	1	arr[0][1]
	2	arr[0][2]
	3	arr[0][3]
1	0	arr[1][0]
	1	arr[1][1]
	2	arr[1][2]
	3	arr[1][3]
2	0	arr[2][0]
	1	arr[2][1]
	2	arr[2][2]
	3	arr[2][3]

अब इस array को print कैसे कराएँगे ?

No. of rows मिलती है → arr.length से
No. of column मिलते है → arr[0].length से

for no of rows → `for(int i=0; i< arr.length ; i++)`
  `for(int j=0; j< arr[0].length ; j++)`
  `{ System.out.println( arr[i][j] +"\t");`
  `}`

for no. of columns

arr[i][j] : i = row no., j = column no.

```
→ public class Main
→{ public static void main(String args[])
→ { int[][] arr = new int[3][4];
    arr[0][0] = 11;
    arr[0][1] = 12;
    arr[0][2] = 13;
    arr[0][3] = 14;

    arr[1][0] = 21;
    arr[1][1] = 22;
    arr[1][2] = 23;
    arr[1][3] = 24;

    arr[2][0] = 31;
    arr[2][1] = 32;
    arr[2][2] = 33;
    arr[2][3] = 34;
    for(int i=0; i < arr.length; i++)
    {
      for(int j=0; j< arr[0].length ; j++)
      { System.out.print(arr[i][j] + "\t");
      }
      System.out.println();
    }
}
```

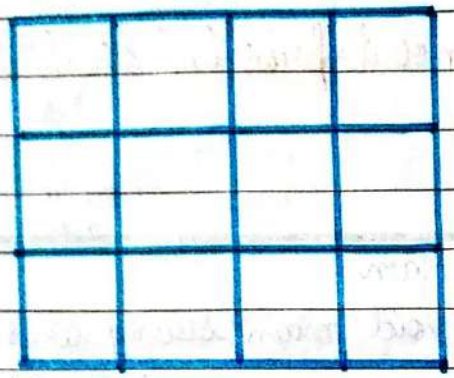
→ इस tab का gap print करने के बाद

Page No.
Date

तो अभी तक हमने देखा Usage of 2-D array. अब समझते हैं Memory Map.

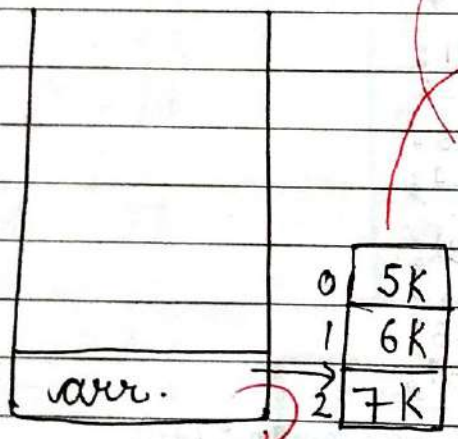
2. Memory Map

```
int [][] arr = new int [3][4];
```



हम सोचते हैं की memory में ऐसे Saved है. lekin ये एक illusion है।

Reality में वैसे store हुआ?



4K = address पर एक arr नाम का variable बना जिसका size है 3 जितनी rows है, उतना ही size होगा 2-d array का। तो जब हम arr.length करते हैं तो हमें no. of rows मिलती है।

तो जब हमें column निकालने हो तब हम array[0].length या array[1].length या array[2].length निकालें, So equal होंगे और columns बताएंगे।

ये वाला array असलियत में सिर्फ Pointers रखता है actual arrays का address रखता है।

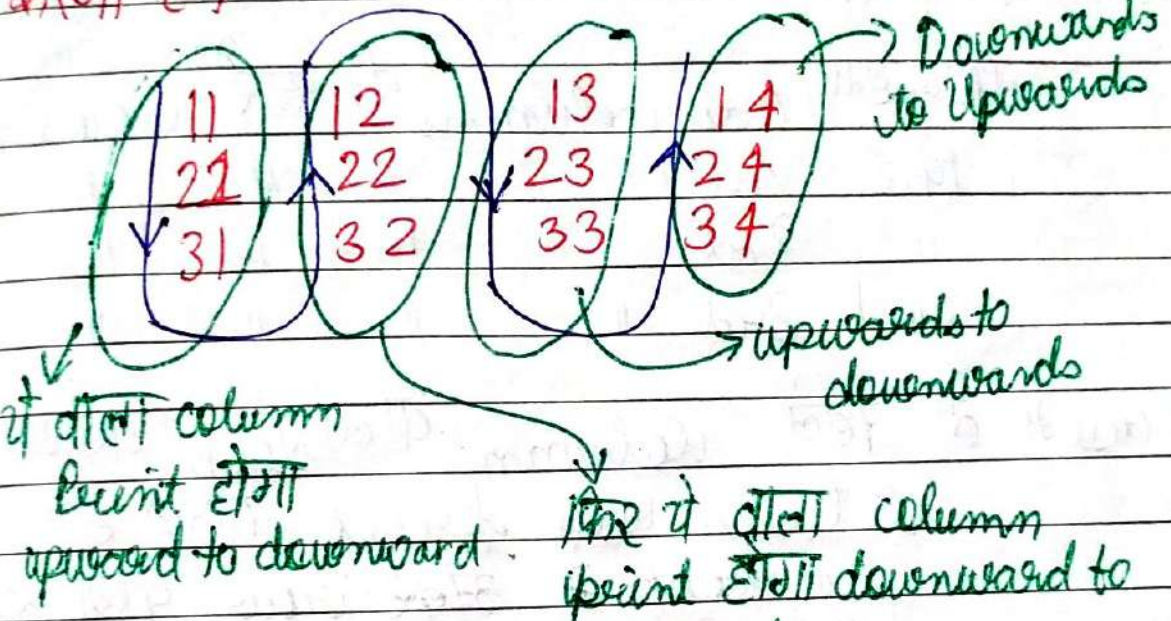
ये arrays होंगे column के size जितने और यहीं पर actual values store होंगी।

Page No.
Date

Ques. 1) WaveDisplay → The State of Wakanda - 1.

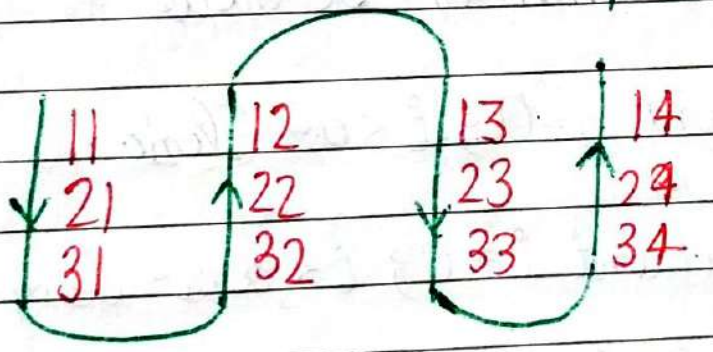
1) Take as Input 2d Array.

2) फिर उस Input array को print करने की तरीके से करना है।



पहले वाला column print होगा upward to downward.

फिर से वाला column print होगा downward to upwards



Output → 11
21
31
32
22
12
13
23
33
34
24
14

पहले सारी rows को upwards to downward print कराते हैं। फिर देखेंगे ques के according

11	12	13	14
21	22	23	24
31	32	33	34

तो हम पहले 1st col print करना है
फिर 2nd " " " "

पहले हम 0th columns की सारी rows print करेंगे
फिर 1st " " " " " "
" 2nd " " " " " "
" 3rd " " " " " "

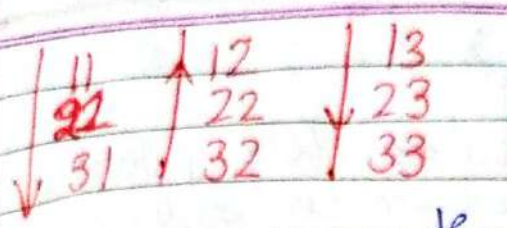
इस case में तो पहले column वाला loop चलेगा। क्योंकि हम column wise print कर रहे हैं। और column वाले loop के अंदर row वाला loop लगेगा ताकि column की हर row को print करा पाए

Output
11
21
31
12
22
32
13
23
33
14
24
34

```
for(int j=0 ; j<arr[0].length ; j++)
{
  for(int i=0 ; i<arr.length ; i++)
  { System.out.println(arr[i][j]) ;
  }
}
```

→ DRY RUN करके जरूर देखना है खुद से जब भी code करेंगे।

अब हम अपने Code को देखते हैं



Even columns हैं तो जायेंगे नीचे की तरफ
Odd columns हैं तो जायेंगे ऊपर की तरफ

last row से first row तक

arr.length - 1 → 0th row

3 - 1 = 2nd row से 0th row तक जायेंगे तो loop में decrement होता जायेगा

```
for(int j=0; j<arr[0].length; j++)
{
  if (j%2==0)
  { for(int i=0; i<arr.length; i++)
    { System.out.println(arr[i][j]);
    }
  }
  else{
    for(int i=arr.length-1; i>=0; i--)
    {
    System.out.println(arr[i][j];
    }
}
```

```
→ import java.util.*;
→ public class Main {
    public static void main (String[]args)
  { Scanner scn= new Scanner (System.in);
    int r = scn.nextInt();
    int c= scn.nextInt();
int[][] arr= new int[r][c];
for(int i=0; i<arr.length; i++)
 { for int j=0; j< arr.length; j++)
   { arr[i][j] = scn.nextInt();
   }
}
for (int j=0; j< arr[0].length; j++)
{   if (j%2==0)
    { for(int i=0; i<arr.length; i++)
      { System.out.println (arr[i][j]);
      }
    }else.
     { for(int i=arr.length; i>=0; i--)
       { System.out.println (arr[i][j]);
     }
}}
}
```

Input ↘

11	12	13	14
21	22	23	24
31	32	33	34

Output ↘

11
21
31
32
22
12
13
23
33
34
24
14

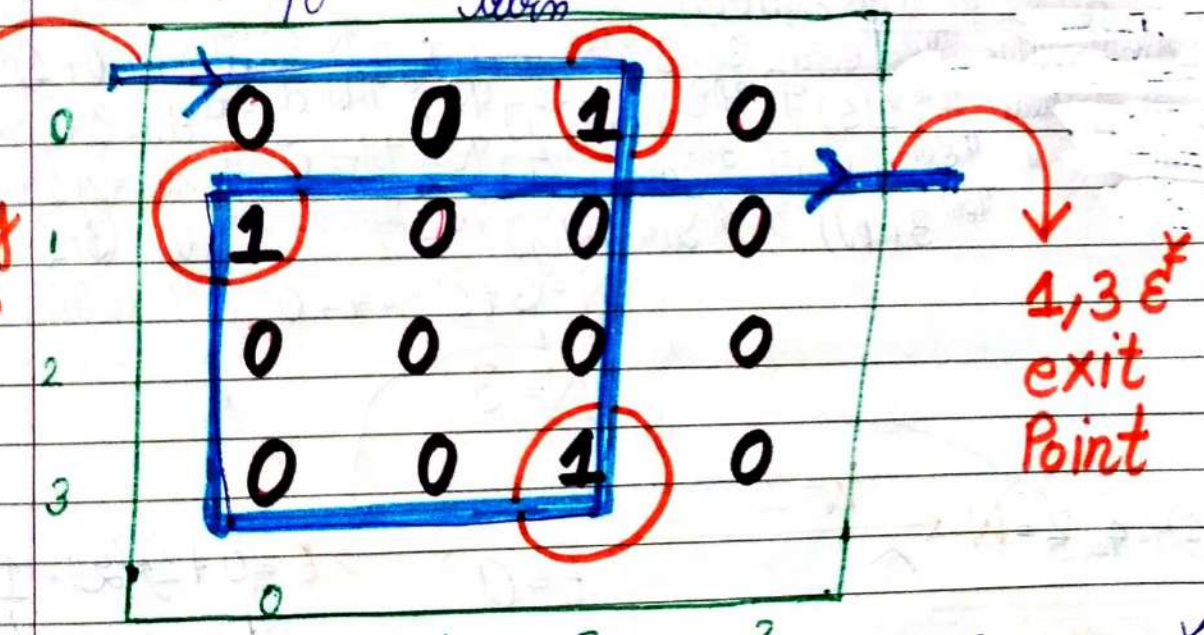
* Java में by default dynamic array ही बनता है।

Ques-2. Exit Point of a Matrix

1. Input लेना है एक 2-D array जिसमे बस 0 और 1 value होगी

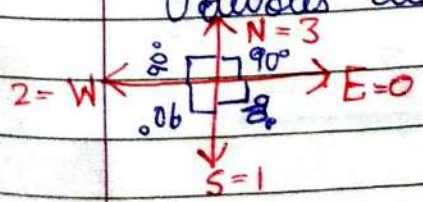
```
0 0 1 0
1 0 0 0
0 0 0 0
1 0 1 0
```

एक बंदा इस array को traverse करता है। सीधा-सीधा चलते रहना है। जैसे ही 1 आएगा तो एक 90° का turn लेलेना है



तो direction नाम का एक variable बनाते है जिसमे east direction है मतलब 0 है value तो हम शुरू करेंगे east direction से।

0 denotes east dir →



हमे ऐसे 1 आते ही 90° rotate करने के लिए केवल इस line को ध्यान से समझना है।

```
dir = (dir + arr[i][j]) %4
```

E = 0
S = 1
W = 2
N = 3

0 → E → East → (0+1) % 4 = 1
1 → S → South → (1+1) % 4 = 2
2 → W → West → (2+1) % 4 = 3
3 → N → North → (3+1) % 4 = 0
0 = East ← & continues

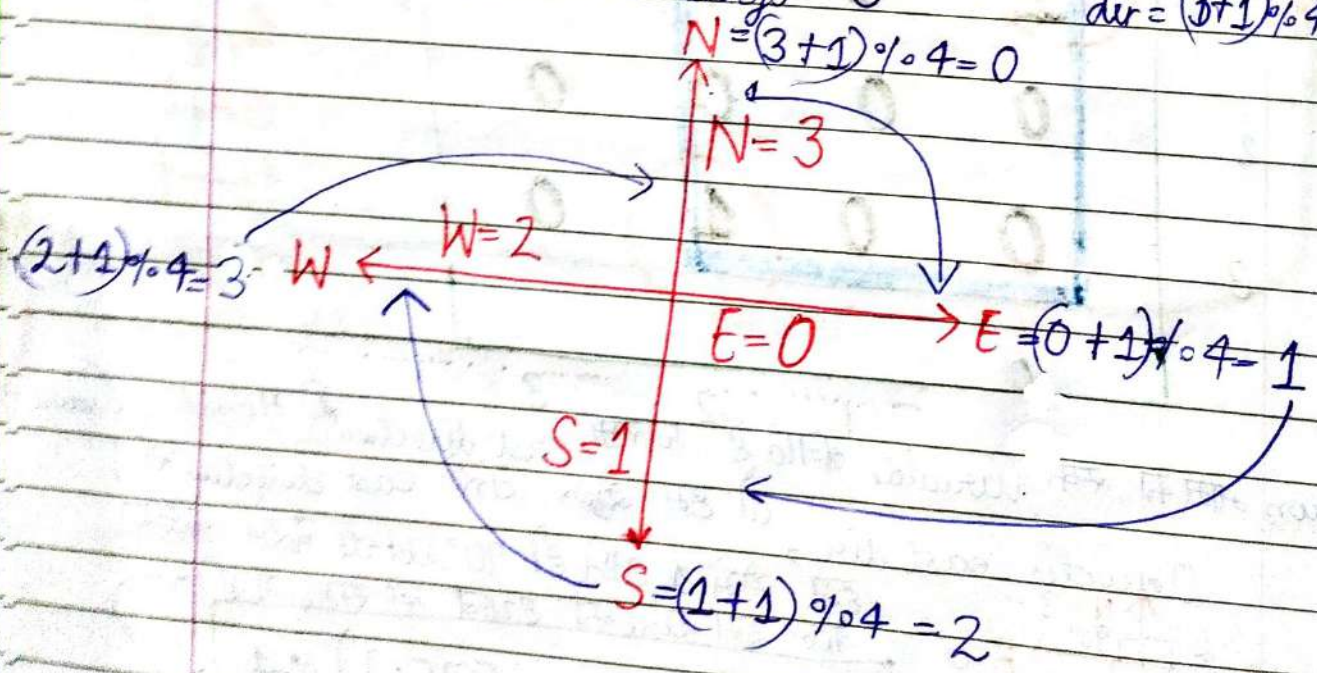
Page No.	
Date	

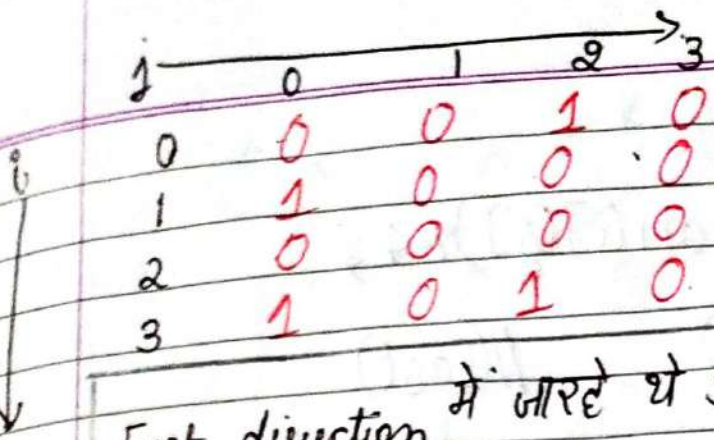
$$dir = (dir + arr[i][j]) \% 4$$

हमें जब जब arr[i][j] पर value 0 मिलेगी तब तब तो सीधा चलेगा और dir में कुछ add भी नहीं होगा तो अगर पहले dir की value 0 थी तो अब भी 0 ही रहेगी।

जैसी ही arr[i][j] पर हमें value 1 मिलेगी तो वो dir में add हो जाएगी

अगर dir की value पहले 0(E) थी और arr[i][j] पर 1 मिला तो अब dir = (0+1) % 4 = 1 = S
" " " " पहले 1(S) थी और arr[i][j] पर 1 मिला तो अब dir = (1+1) % 4 = 2 = W
" " " " पहले 2(W) थी और arr[i][j] पर 2 मिला तो अब dir = (2+1) % 4 = 3 = N
" " " " पहले 3(N) थी और arr[i][j] पर 3 मिला तो अब dir = (3+1) % 4 = 0 = E





East direction में जारहे थे अब 90° turn लेके South में जाना है तो i बढ़ेजाएगा.

South → i++

South में जारहे थे अब 90° turn लेके West में जाना है तो j-- होरहा है मतलब j की value कम होरही है।

West → j--

West में जारहे थे अब 90° turn लेके North में जाना है तो i-- होरहा है मतलब i की value कम होती जारही है।

North → i--

North में जारहे थे 90° turn लेके East में जाना है तो j++ होरहा है मतलब j की value increase होरही है.

East → j++

```
while (true)
{
  dir = (dir + arr[i][j]) % 4;
  if (dir == 0)          //(East)
  {
    j++;
  }
  else if (dir == 1)     //(South)
  {
    i++;
  }
  else if (dir == 2)     //west.
  {
    j--;
  }
  else                   // North
  {
    i--;
  }
}
```

0	0	1	0
1	0	0	0
0	0	0	0
1	0	1	0

j++ →

j की value होगयी 4 > arr[0].length तो ये होगा exit point और वही output देना है।

तो arr[i][j] की value जिसमे last row या column 1 subtract या add करते ही या तो i की value < 0 हो जाए या i की value > arr.length हो जाए या फिर j की value < 0 हो जाए या j की value > arr[0].length हो जाए तो होगा exit point

```
int i=0;
int j=0;
int dir=0;
while(true)
{dir=(dir+arr[i][j])%4;
 if(dir==0)
 {j++;
  if(j==arr[0].length)
    {j--;
     break;
    }
 else if(dir==1)
 {i++;
  if(i==arr.length)
 {i--;
  break;
 }
}else if(dir==2)
{j--;
  if(j==-1)
  {j++;
   break;
  }
}
else{i--;
  if(i==-1){i++;
  break;
```

ExitPointOfAMatrix.java

```
import java.io.*;
import java.util.*;

public class Main
{
public class void main(String[] args)
{Scanner scan= new Scanner(System.in);
int r= scan.nextInt();
int c= scan.nextInt();

int[][]arr = new int[r][c];
for(int i=0; i<arr.length; i++)
{for(int j=0; j<arr[0].length; j++)
{
arr[i][j] = scn.nextInt();
}
}
int i=0;
int j=0;
int dir = 0;

// 0 is East
// 1 is South
// 2 is West
// 3 is north.
```

→ while (true)
{ dir = (dir + arr[i][j]) % 4;
if (dir == 0) // East
{
j++;
if (j == arr[0].length)
{
j--;
break;
}
}else if (dir == 1) // South
{
i++;
if (i == arr.length)
{
i--;
break;
}
}else if (dir == 2) // West
{
j--;
if (j == -1)
{
j++;
break;
}}
}

वापस आए 1 बार से break करदो क्योंकि हमें तो exit point print करना है।

```
                                    // dir==3 north
else
  {  i--;
     if(i==-1)
     {  i++;
        break;
     }
  }
}
System.out.println(i);
System.out.println(j);

  }
}
```

2d array > is better than 2-d list

Head First Java.
Head First C.
Head First OOAD
Head First Design Pattern

} ये 4 Books हर बच्चे को पढ़नी चाहिए

HW

Page No.
Date

2D ARRAY

1. Article — 2D Arrays Demo.
2. Code — 2D Arrays Demo. —] Done in CW
3. Article — Matrix Multiplication
4. Code — Matrix Multiplication
5. Article — The state of Wakanda 1.
6. Code — The state of Wakanda 1. —] Done in CW
7. Article — Spiral Display
8. Code — Spiral Display.
9. Article — Exit point of Matrix.
10. Code — Exit point of Matrix. —] Done.
11. Article — Rotate by 90 Degrees.
12. Code — Rotate by 90 Degrees.

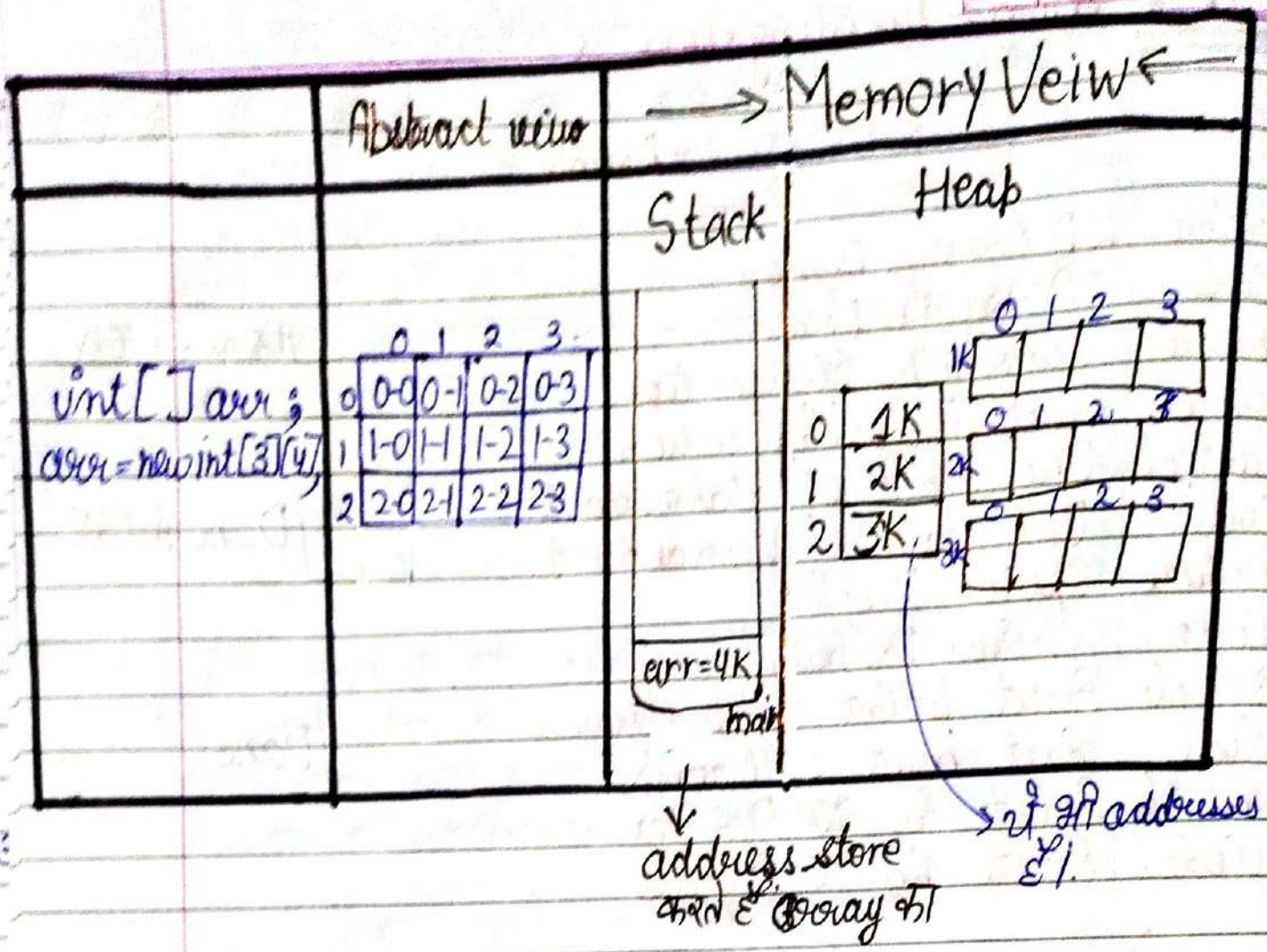
1) 2-D Array Article

1-D Array ? → 1-D array are the ... us that represent many elements and are one-dimensional.

Similarly, 2D arrays are 2 dimensional arrays which are used to represent matrices.

* We are going to meditate on 2 aspects of 2-D array :- Abstract View & Memory View
  - Abstract View → Usage
  - Memory View → Memory used by 2d memory

for most of our discussion, the abstract view is sufficient However, Memory view allows us to understand 2D array with deep understanding.



1. * If we declare arr[][]
int arr[][];
then a stack is created in the memory which has a variable arr which stores a null value (i.e it doesn't point to anything).

2. * When we define this array as arr = new int[3][4] then the first dimension = 3 denotes the rows & the second dimension = 4 denotes the column.

Abstract View

	0	1	2	3
0	0-0	0-1	0-2	0-3
1	1-0	1-1	1-2	1-3
2	2-0	2-1	2-2	2-3

Here we see a grid/matrix with rows and a column, each starting from index 0.

↳ Inside each cell, the position of that cell is written in the format "row - column". Foreg → 2-1 denotes the cell at 2nd row & 1st column.

Memory View

If we actually consider arr = new int[3][4], then we realize that, it denotes that we have 3 arrays (1-D) of size 4 each. As we can see in the Memory View, that the variable arr is stored at address 4K. This 4K address is storing an array of size 3 which consists of 3 addresses and at each of these 3 addresses, an array of size 4 is present.

Do we even realize when we define the given array, actually 4 arrays are formed.

How? → 3 are the arrays with 4 Sizes each which will contain integers and the 4th array is of Size 3 which contains the addresses (1K, 2K, 3K) as shown in the Memory view

Hence, arr is an array of arrays.

3. * The Programming Implementation in 2-D array is done using 2 loops to iterate over the arrays and by storing elements in the cell. Similar to the input process, the output process is also carried out using 2 loops.

4. * Time Complexity :- $O(n^2)$
This time complexity is quadratic due to the use of nested for loop.

* Space Complexity :-
As a 2D array is used to store input values, therefore space complexity is Quadratic

MATRIX MULTIPLICATION

Page No.
Date

Ques-3) This question takes 2 matrices as the Input & returns the product of those matrices on the multiplication

Solution

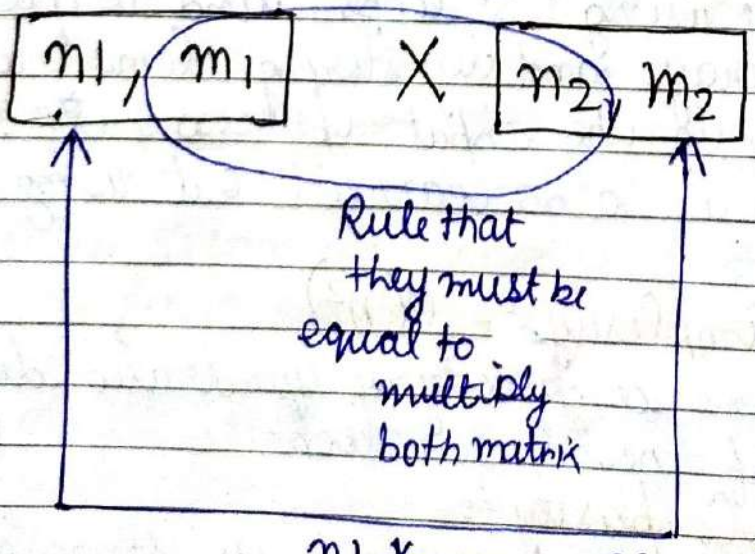
* Here Let $n1$ = no. of rows of Matrix 1.
  Let $m1$ = no of column of Matrix 1.

  Let $n2$ = no. of rows of Matrix 2.
  Let $m2$ = no. of column of Matrix 2.

We must understand that, not all the matrices can be multiplied with each other.
They can be multiplied iff
no. of column of Matrix 1 = no. of rows of Matrix 2

i.e $m1 == n2$

Only if $(m1 == n2)$ then a product matrix can be formed of size $(n1 * m2)$.



Eg → Lets assume, we are given a matrix A[2,3] and another matrix B[3,4]

Can Matrix A be multiplied to Matrix B?

Yes, Both of these Matrix can be multiplied to each other & this is because no. of column of Matrix A is equal to no. of rows of Matrix B.

A new matrix will be formed of size $(m_1 * m_2)$ i.e P[2][4].

↓

P is the ~~name of product matrix~~

	0	1	2
0	a_{00}	a_{01}	a_{02}
1	a_{10}	a_{11}	a_{12}

A

×

	0	1	2	3
0	b_{00}	b_{01}	b_{02}	b_{03}
1	b_{10}	b_{11}	b_{12}	b_{13}
2	b_{20}	b_{21}	b_{22}	b_{23}

B.

	0	1	2	3
0	P_{00}	P_{01}	P_{02}	P_{03}
1	P_{10}	P_{11}	P_{12}	P_{13}

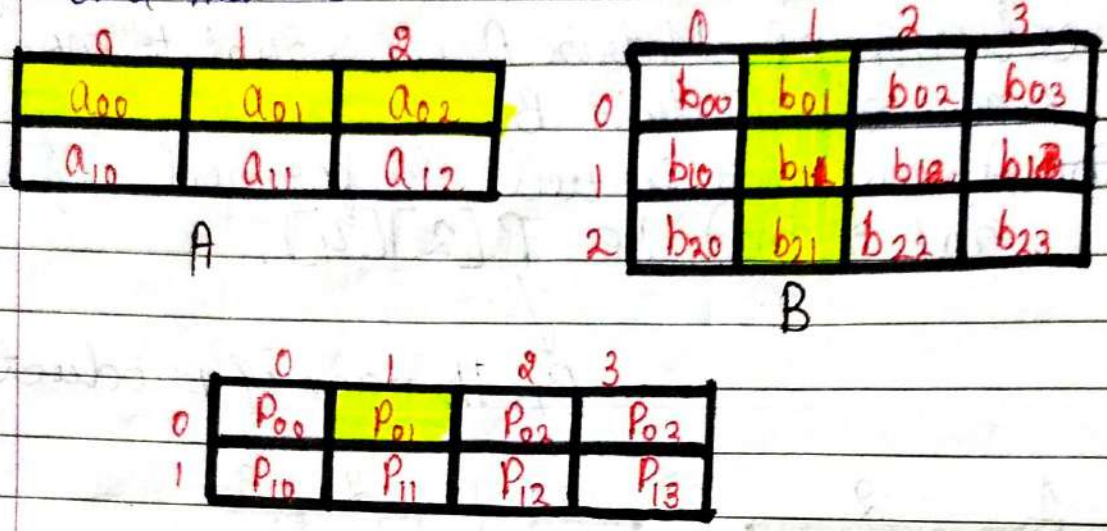
P.

Now, we need to understand how to calculate the elements in the product matrix P.

$$P_{ij} = a_{i0} b_{0j} + a_{i1} b_{1j} + \cdots + a_{i(n-1)} b_{(n-1)j}$$

$$= \sum_{k=0}^{n-1} a_{ik} b_{kj}$$

The value at P01 is the sum of the product of elements of the 0th row of the 1st matrix with the elements at the 1st column of the 2nd matrix.



* Time Complexity $\Rightarrow O(n^3)$

The time complexity will be cubic i.e n^3 because 3 nested for loops are used.

* Space Complexity $\Rightarrow O(n^2)$

As 2D arrays are used to store numbers, therefore space complexity is quadratic.

Now, lets code it:-

```
→ import java.util.*;
→ public class Main{
→ public static void main(String[]args).
{ Scanner scan = new Scanner (System.in);
int row1 = scan.nextInt();
int column1 = scan.nextInt();
int arr1[][] = new int[row1][column1];
for(int i=0; i<row1; i++)
{for(int j=0; j<column1; j++)
{ arr1[i][j] = scan.nextInt();
}
}
int row2 = scan.nextInt();
int column2 = scan.nextInt();
int arr2[][] = new int[row2][column2];
for(int i=0; i<row2; i++)
{for(int j=0; j<column2; j++)
{ arr[i][j] = scan.nextInt();
}
}
if(column1==row2)
{int product[][] = new int[row1][column2]
for(int i=0; i<row1; i++)
{for(int j=0; j<column2; j++)
{ int sum=0;
for(int k=0; k<column1; k++)
{ sum = sum + (arr1[i][k] * arr2[k][j]);
}
product[i][j] = sum;
}}
for(int i=0; i<row1; i++)
{for(int j=0; j<column2; j++)
{ System.out.print(product[i][j] + " ");
}
System.out.println();
}
else{System.out.println("Invalid Input");}
}}
```

Arr1 Input

Array2 Input.

Check if Arr1 & Arr2 can be multiplied

Multiplication

k is inc with each loop.

column no of arr1 is inc with each loop so k is written here

row no. of arr2 is increasing with each loop. so k is here

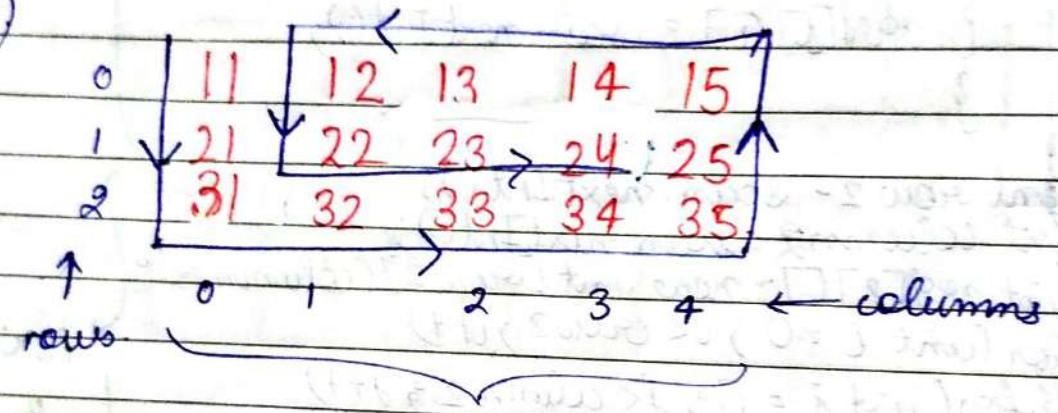
Print output

(Ques 4) # SPIRAL DISPLAY

Page No.
Date

* Remember that algorithm for success in coding is
while (! success)
try again ();

eg-1)



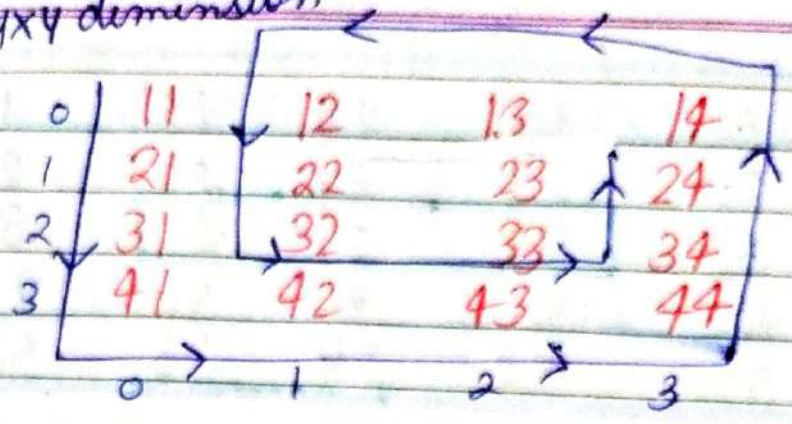
Output

11
21
31
32
33
34
35
25
15
14
13
12
22
23
24

Page No.
Date

2-darray

Eg 2) 4x4 dimension

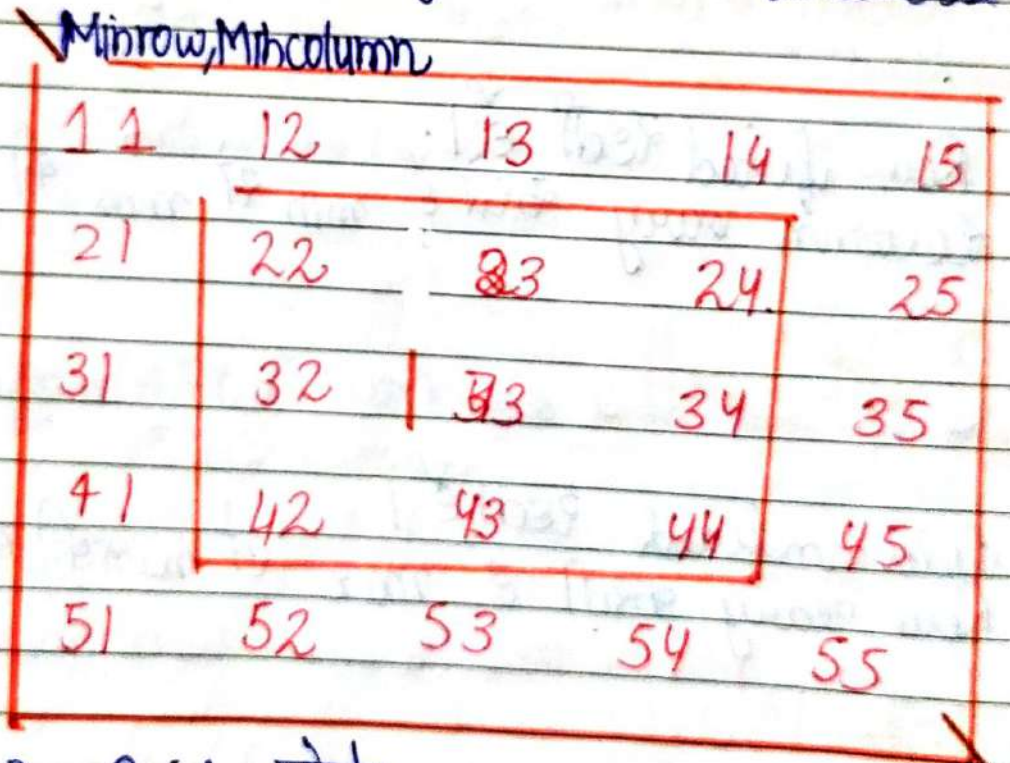


Output

11 21 31 41 42 43 44 34 24 14 13
12 22 32 33 23

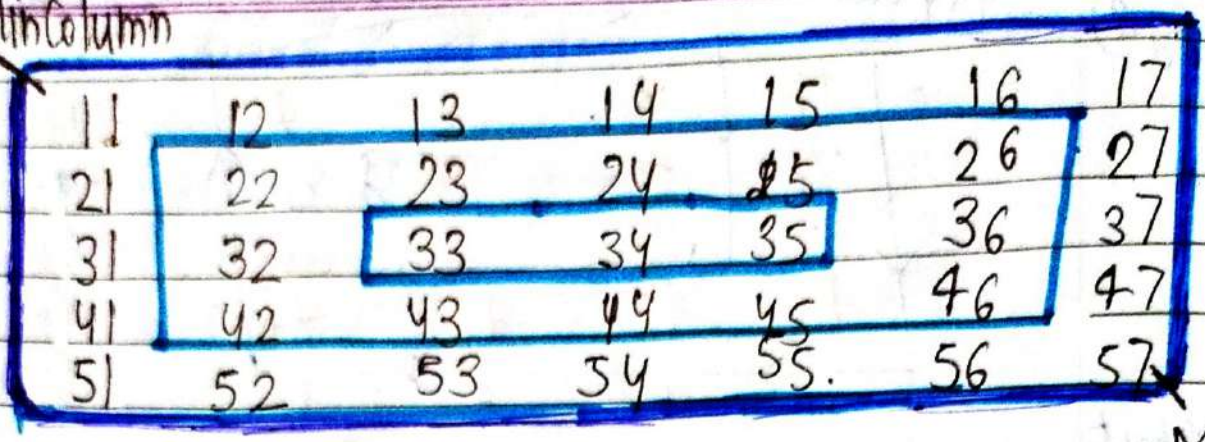
Spiral Way traverse करने से अच्छा होगा कि हम pattern को box by box wise traverse करें।

To identify box we need min row, min col. & max row, max col.



1) Box के अंदर Box Print करेंगे
2) एक Box Print करने के 4 wall Print करनी होगी
3) 4 wall Print करने के लिए 4 loops लगाने पड़ेंगे चारों walls के
4) सबसे पहले left wall print करेंगे फिर bottom वाली फिर Right और फिर Top wall.

Page No.
Date



1.* **Left Wall**
1) Column fixed रहता है।
2) Row vary करती है Min से Max की तरफ

2. **Bottom Wall**
1) Row fixed रहती है। i.e.
2) Column vary करते है min से max की तरफ

3) **Right Wall**
1) Column fixed रहता है।
2) Row vary करती है max से min की तरफ

4) **Top Wall**
1) Row fixed रहती है।
2) Column vary करते है Max से Min की तरफ

पहले 1 Box को print करने का try करते हैं।
और अभी हम corners को 2 बार print होने देंगे।
बस Box की values print कराके देखना है।

```
int minr = 0;
int minc = 0;
int maxr = arr.length - 1;
int m    = arr[0].length - 1;
while(true){
// left wall.
    for (int i = minr , j = minc; i<=maxr;i++)
    { System.out.println (arr[i][j];
    }
// bottom wall
    for (int i = maxr , j= minc ; j<=maxc;j++)
    {System.out.println (arr[i][j];
    }
// right wall.
    for (int i=maxr, j= maxc; i>=minr ; i--)
    {
    System.out.println (arr[i][j];
    }
// top wall.
    for(int i=minc, j=maxc; j>=minc ;j--)
    {System.out.println (arr[i][j];
    }
```

// Box तो Print हो रहा है लेकिन Corners repeat हो रहे हैं।

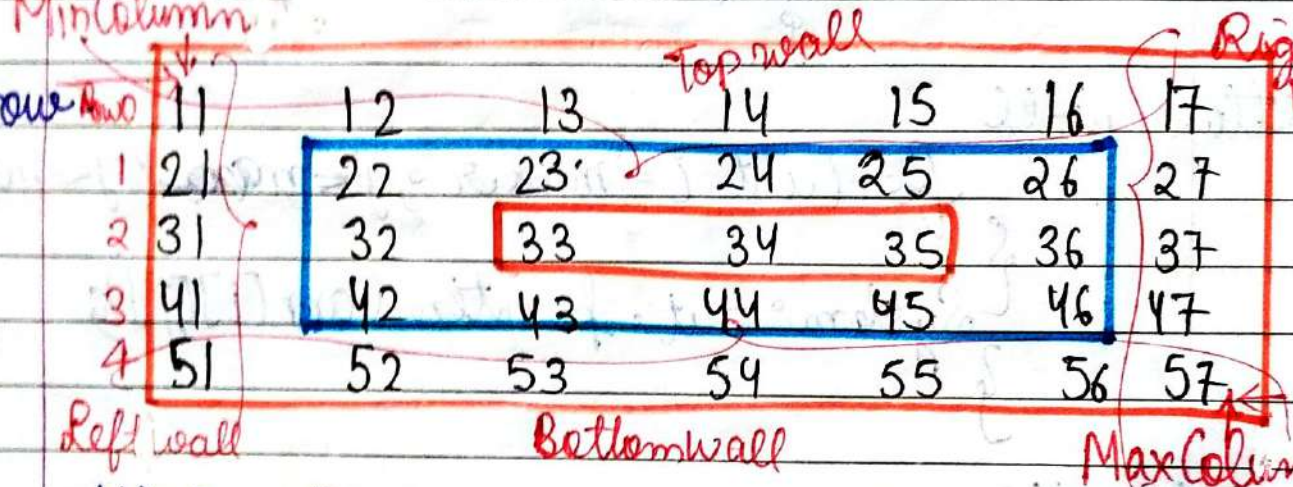
अब हमें repeated corners के लिए कुछ करना पड़ेगा।

left wall → को indicate करता है → min column

bottom wall → को indicate करता है → max row

right wall → को indicate करता है → max column

top wall → को indicate करता है → min row



जैसे ही हमने left wall print करदी, तभी हमें min column को बढ़ा देना है।

1. जैसे ही Left Wall print की min column ++ करदो Left Wall के बाद Bottom wall print होता है। और Bottom wall में columns change होते हैं: min column से max column तक। अब अगर हम Bottom wall print करेंगे, तो min column start होगा 2nd column से और 2 बार 5 Print नहीं होगा।

while (true)
{
// left wall
for (int i = minr, j = minc ; i <= maxr; i++)
{
System.out.println (arr[i][j]);
}
minc++;

2. जैसे ही Bottom wall print हो जाए, अब max row को -- (decrease) कर दीजिए। अब जब max row decrease होगया है तो वो Row भी नहीं 3 को point करेगा और अब Bottom wall के बाद तो Right wall होगी print और right wall को indicate करता है max column और max column (Right wall print होगी max row से min row तक

for (int i = maxr, j = minc ; j <= maxc ; j++)
{
System.out.println (arr[i][j]);
}
maxr--;

3. जैसे ही Right wall print हो max column -- करदो bcz max column represent करता है right wall को। अब Top wall में col change होते हैं max column से min column तक। अब अगर हम Top wall print करेंगे तो max column को 1 से 2 बार print नहीं होगा।

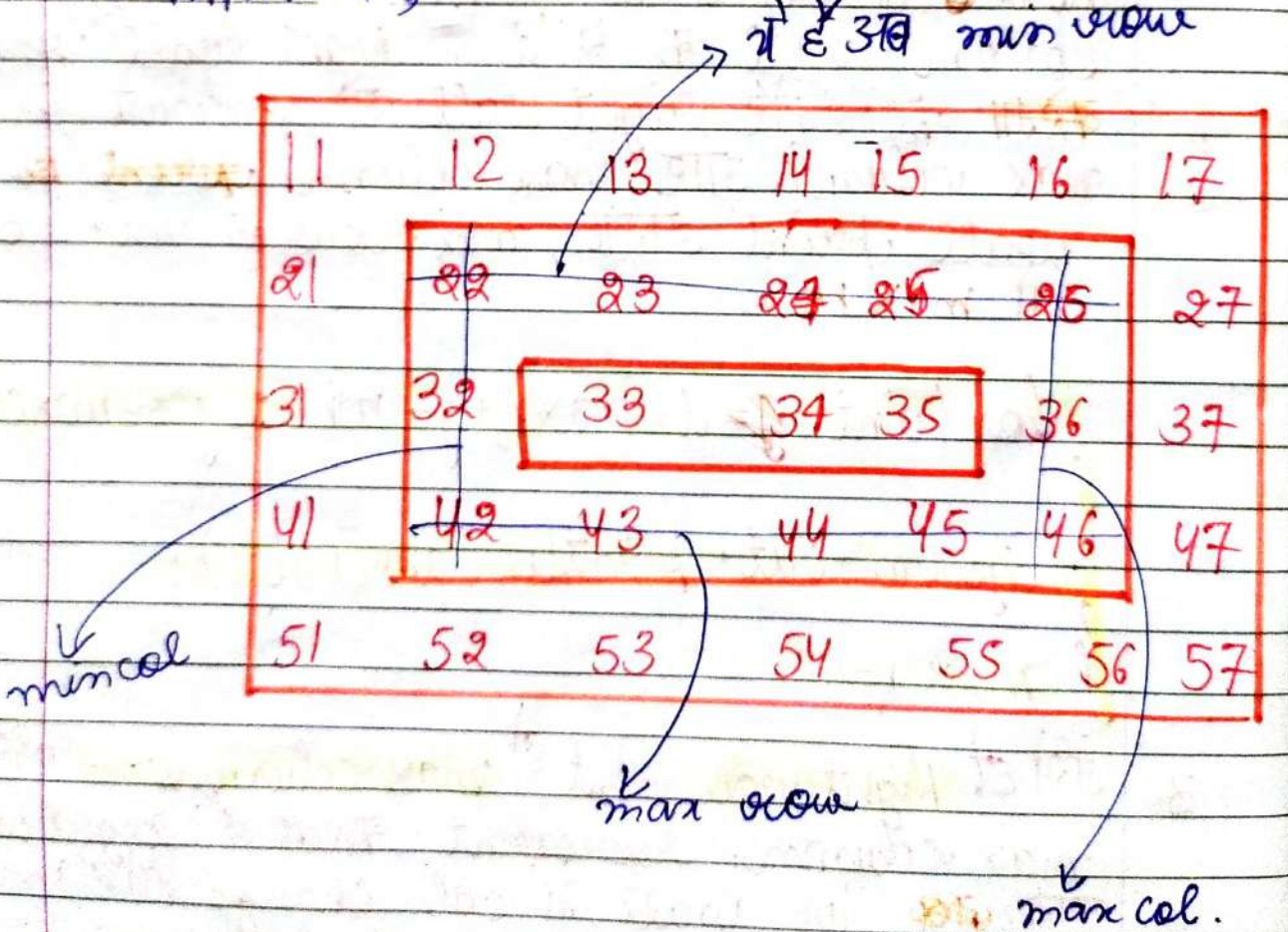
Page No.
Date

// right wall

```
for(int i = maxr , j = maxc ; j > minr ; j--)
{ System.out.println(arr[i][j]);
}
   maxc--;
```

↳ जैसे ही Top wall print होगया, min row ++ करदो ताकि जो अगला Box है वो 1st (अगले) row से शुरू हो।

```
minr++;
```



कबतक Loop चलता रहेगा? जब तक no. of digit count in array, no. of digit printed नहीं हो जाती।

Page No. | Date

last box में जरूरी नहीं है कि चारो walls हो हमारे पास तो एक count variable को check जरूर ही रखेंगे (no. of digit count)

```
import java.util.*;
import java.io.*;
public class Main{
public static void main(String[]args)
{ Scanner scan=new Scanner(System.in);
int n= scn.nextInt();
int m= scn.nextInt();
int arr[][]=new Int[n][m];
for(int i=0; i<arr.length; i++)
{for(int j=0; j<arr[0].length; j++)
  { arr[i][j]=scn.nextInt();
  }
}
int minrow=0;
int mincol=0;
int maxrow=arr.length-1;
int maxcol=arr[0].length-1;
int tne=n*m  //total number of elements
int count=0;
while(count<tne)
{ //left wall
  if(count<tne)
  {for(int i=minrow; i<=maxrow; i++)
    {System.out.println(arr[i][mincol]);
    count++;
    }
  }mincol++;
```

```
// bottom wall
  if (count < itne)
{
{ for(int i = mincol ; i <= maxcol ; i++)
{
{System.out.println(arr[maxrow][i]);
}count++;
}
}
  maxrow--;
// right wall
  if (count< itne)
{ for(int i = maxrow ; i >= minrow ; i--)
  { System.out.println(arr[i][maxcol]);
    count++;
  }
}
  maxcol--;
// top wall;
  if (count < itne)
  {
  { for(int i = maxcol; i >= mincol; i--)
    {
      {System.out.println( arr[minrow][i]);
       count++;
    }
  }
  minrow++;
}
}
}
```

Page No.
Date

Time Complexity :-

$O(n^2)$

As there is nested for loop & the outer for loop runs n times. There are 2 inner for loops. Either one will run in each iteration.

Making the time complexity : $O(n) * O(m)$

$= O(n * m)$.

Space Complexity

$O(1)$.

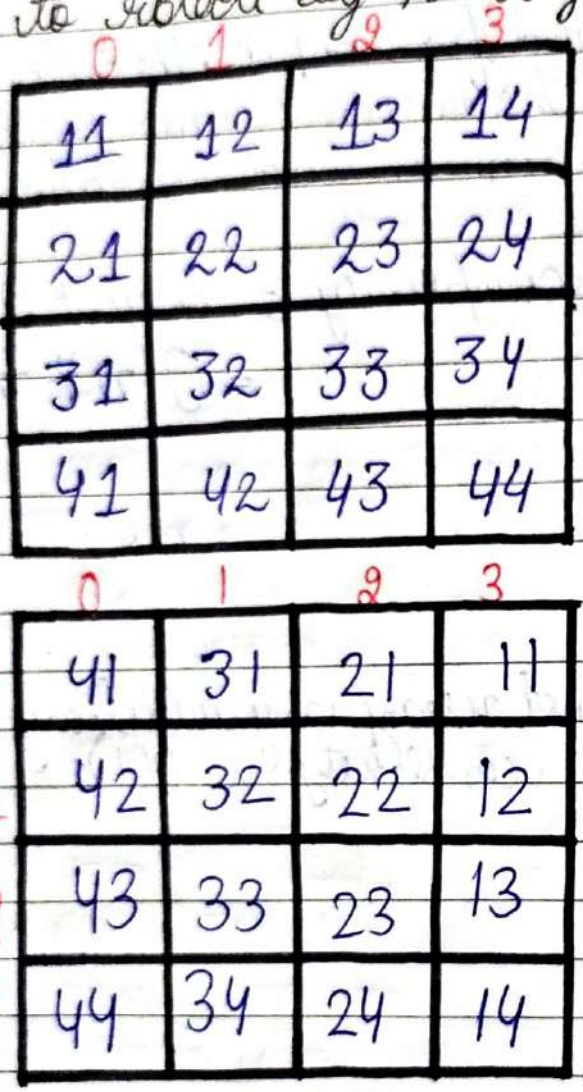
Since we are not using any auxiliary space & hence the space complexity is $O(1)$.

Ques. 5).

Rotate by 90° Degrees

Page No.
Date

→ We are given a square matrix and we have to rotate by 90° degrees.



: Rotate Matrix 90° by 90° degree

So this is a 90° turn. This turn can be clockwise also, & this turn can be counter clockwise also, it only Depend on what question wants. Our question want 90° turn to be clockwise

After 90° rotation

Extra Space नहीं लेना और n^2 complexity में ही करनी है।

Here, the first row become the last column. 0th row will be 3rd column.

* For rotating a matrix to 90° degrees clockwise. We need to transform each row of a Matrix to a column in rotated matrix

* For rotating matrix to 90°, we need to transform the rows into columns & the columns to rows in a result matrix.
So, the number of rows in rotated matrix will be equal to no. of columns of Original matrix and number of columns in rotated matrix will be equal to no. of rows. no (m) row * (n) no. column — original; (n) no. of col * (m) no. of row

```
int [][] rotatedMatrix = new int
        [cols of Original Matrix][rows of Original Matrix]
```

→ Size of Rotated Matrix will be n * m

In 90° Clockwise Rotation:
So, to summarise:- If size of original was m * n.

1. First row of Original Matrix will be last column of the Rotated Matrix.
2. Second row of original Matrix will be second last column of the Rotated Matrix
3. Third row of Original Matrix will be the last third column of the Rotated matrix & so on.

& So on

* We have a way of converting the rows of a matrix in the column of a matrix and that way is called Transpose of a Matrix.

So, What is Transpose of a Matrix?
The transpose of a Matrix is nothing but a flipped version of the matrix when we exchange the rows & columns of a matrix such that each element A(i, j) gets swapped with A(j, i), then the matrix obtained is called the Transpose of Matrix.

इसे counter clock wise rotation बोलते हैं.

Page No.
Date

Anticlockwise 90° rotation → TRANSPOSE

11	12	13	14
21	22	23	24
31	32	33	34
41	42	43	44

⟹

11	21	31	41
12	22	32	42
13	23	33	43
14	24	34	44

Transpose में Anti-clockwise / counter clockwise 90° rotation होती है।

Transpose

↓

90° Anti Clockwise Rotation

→ 1) पहली row पहला column बन जाता है।
→ 2) दूसरा row दूसरा column बनजाता है।
→ 3) Third row → Third बन जाता है।
& so on

While Doing Transpose
element at arr[i][j] बन जाएगा
element at arr[j][i].

90° Clockwise Rotation

1) पहली row last column बनजाती है। 2) दूसरी row last 2nd column बन जाती है।
3) Third row, last 3rd col बन जाती है
...

Clockwise 90° Rotation

11	12	13	14
21	22	23	24
31	32	33	34
41	42	43	44

⟹

41	31	21	11
42	32	22	12
43	33	23	13
44	34	24	14

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

90° CLOCK WISE ROTATION →

m	i	e	a
n	j	f	b
o	k	g	c
p	l	h	d

90° ANTI CLOCK - WISE ROTATION → TRANSPOSE

a	e	i	m
b	f	j	n
c	g	k	o
d	h	l	p

EXCHANGE

REVERSE COLUMNS →

1) first row first column करदो
2) 2nd row → 2nd column & so on
3) arr[i][j] को बनादो arr[j][i]

1) Transpose के बाद Obtained array के columns reverse करदो इसका मतलब 1st column को last बनादो, last column को first
2) Second column को last से second column बनादो
3) Third Column को last Third column बनादो & so on

→ TO TRANSPOSE or
→ To rotate 90° Clockwise

Page No.
Date

* तो पहले हमारा काम होगा Transpose करना
* फिर Transposed Matrix के Columns को reverse करना है।

a	b	c	d
e	f	g	h
i	j	k	l
m	n	o	p

arr[i][j] को arr[j][i] से exchange करना है या swap करना है एक ही बात है और बस एक ही बार करना है ये बात main है।

हमें Transpose करते Time ये गलती नहीं करनी है।

```
for(int i=0; i<arr.length; i++)
    for(int j=0; j<arr[0].length; j++)
    {
        int temp = arr[i][j];
        arr[i][j] = arr[j][i];
        arr[j][i] = temp;
    }
```

Don't Do like this → row के हर column के लिए ये loop चलेगा

Don't Do this Mistake

Since we know that Transpose is a matrix in which arr[i][j] gets swapped or interchanged with arr[j][i], and this is exactly what we are going to do. If we traverse the entire matrix and swap arr[i][j] with arr[j][i], it is not going to work, and we will get the input matrix only as the result. ऐसा इसलिए होता है क्योंकि हर पूरा matrix ही Traverse कर रहे हैं तो सारी row की सारी columns को Traverse किया जा रहा है तो जो values हम पहले ही swap कर चुके होते हैं, उन्हें हम वापस से swap कर देते हैं।

ऐसा करने पर हमें अपना Input Matrix ही मिलेगा as Output और Transpose नहीं होगा। तो अगर हमें Transpose करना है Matrix को, तो या तो हम Upper Triangle को ही Swap करेंगे या फिर सिर्फ Lower Triangle को।

11	12	13	14
21	22	23	24
31	32	33	34
41	42	43	44

→ These 3 lines are making an Upper Triangle.

We will traverse only this part for a Transpose.

After Traversing Only the Upper Part or only the Lower Part of the 2-d array & then doing the transpose, we get this 2-d array

11	21	31	41
12	22	32	42
13	23	33	43
14	24	34	44.

TRANSPOSED MATRIX

अगर हमें Transpose of Lower Triangle करना है।

Transpose of Lower Triangle के लिए loop

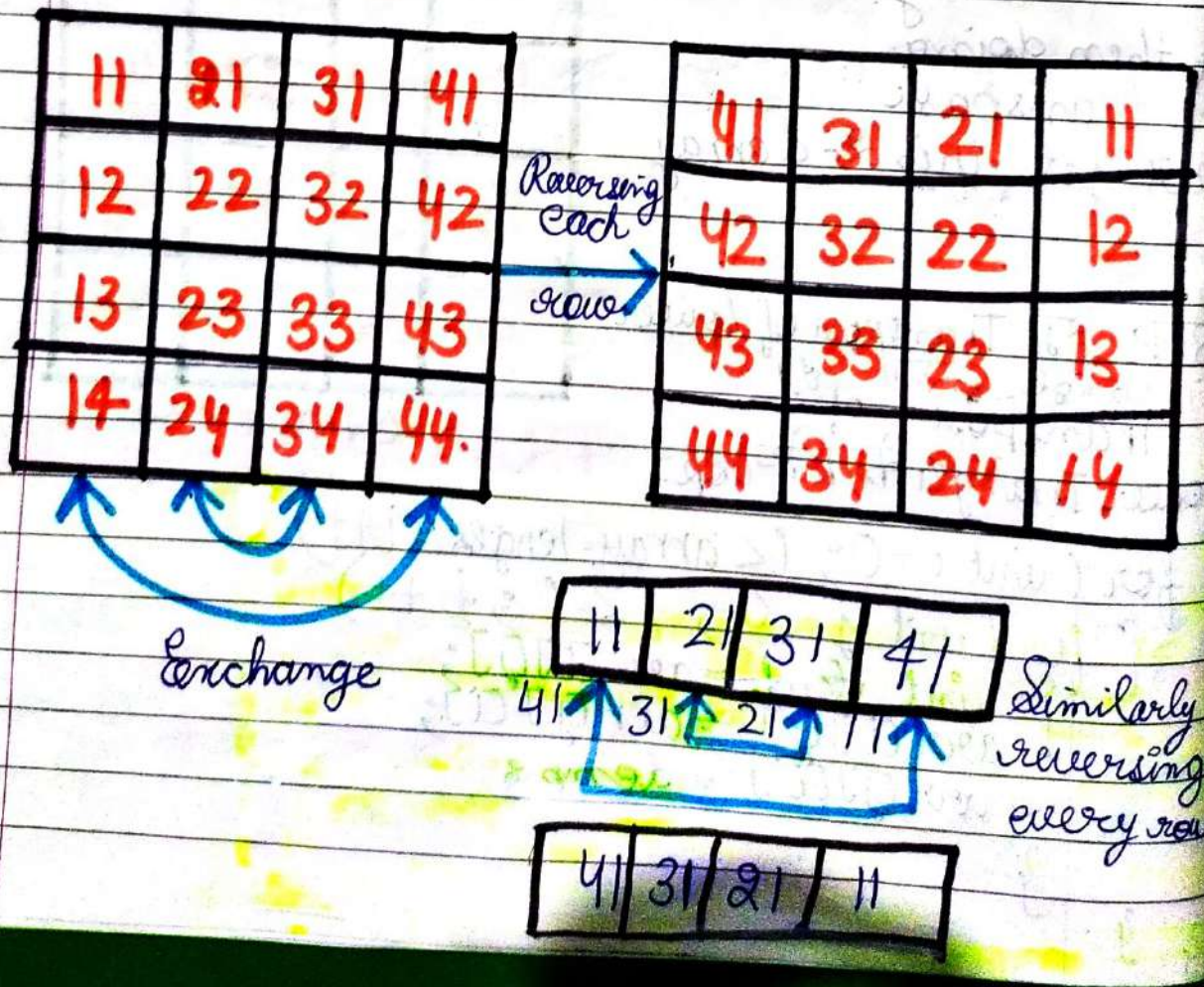
```
{ for (int i = 0; i < arr.length; i++)
    { for (int j = 0; j <= i; j++)
        { int temp = arr[i][j];
          arr[i][j] = arr[j][i];
          arr[j][i] = temp;
        }
    }
}
```

अगर हमें Transpose of Upper Triangle करना है।

तो Transpose of Upper Triangle के लिए Loop

```
for (int i=0; i< arr.length; i++)
{
  for (int j=i; j< arr[0].length; j++)
  {
    int temp = arr[i][j];
    arr[i][j] = arr[j][i];
    arr[j][i] = temp;
  }
}
```

→ To reverse the Columns of Transposed Matrix



Row by Row फिर line से column reverse करो

Reverse column row by row

Page No.
Date

```
for(int i=0; i<arr.length; i++)
{
  int li = 0;                          ---> min col
  int ri = arr[0].length - 1;          ---> max col
  while ( li < ri)                     ---> right index from col
                                       ---> left index of col
  {
    int temp = arr[i][li];
    arr[i][li] = arr[i][ri];
    arr[i][ri] = temp;
    li++;
    ri--;
  }
}
```

> जब तक left index of column right index से कम हो तभी तक ही करना है

```
public class Main
public static void main (String[] args){
    Scanner scan = new Scanner (System.in);
    int n = scn.nextInt();

    int[][] arr = new int[n][n];
    for(int i=0; i<arr.length; i++)
    {
        for(int j=0; j<arr[0].length; j++)
        {
            arr[i][j] = scn.nextInt();
        }
    }

    //transpose
    for(int i=0; i<arr.length; i++)
    {   for(int j=0; j<=i; j++)
        {
            int temp = arr[i][j];
            arr[i][j] = arr[j][i];
            arr[j][i] = temp;
        }
    }
    // reverse columns
    for(int i=0; i<arr.length; i++)
    {   int li=0;
        int ri = arr[i].length-1;
        while(li<ri)
        {int temp = arr[i][li];
         arr[i][li] = arr[i][ri];
         arr[i][ri] = temp;
            li++;
            ri--;
        }
    }
```

```
for(int i=0; i<arr.length; i++)
{
  for (int j=0; j<arr[0].length; j++)
  {
    System.out.print(arr[i][j]+" ");
  }
  System.out.println();
}
}
```