/\*

2D array

datatype arr\_name[row\_size][col\_size];

int arr[3][3];

0 1 2

0 1000 1004 1008

1 1012 1016 1020

2 1024 1028 1032 arr[1][1];

\*/

#include <stdio.h>

int main()

{

int arr[3][3], row, col;

for(row = 0; row < 3; row++)

{

for(col = 0; col < 3; col++)

{

scanf("%d", &arr[row][col]);

}

}

for(row = 0; row < 3; row++, printf("\n"))

{

for(col = 0; col < 3; col++)

{

printf("%d ", &arr[row][col]);

}

}

return 0;

}

/\*

Check whether the given matrix is NULL matrix or not.

3 3 3 3

0 0 0 0 0 0

0 0 0 0 1 0

0 0 0 0 0 0

Null Matrix Not Null Matrix

\*/

#include <stdio.h>

int main()

{

int NROW, NCOl, row, col;

scanf("%d %d", &NROW, &NCOl);

int MAT[NROW][NCOl];

for(row = 0; row < NROW; row++)

for(col = 0; col < NCOl; col++)

scanf("%d", &MAT[row][col]);

for(row = 0; row < NROW; row++)

{

for(col = 0; col < NCOl; col++)

{

if( MAT[row][col] != 0 )

{

printf("Not Null Matrix");

return 0;

}

}

}

printf("Null Matrix");

return 0;

}

/\*

Check whether the given matrix is Identity matrix or not.

3 3

1 0 0 1 0 1

0 1 0 0 1 0

0 0 1 0 0 1

Identity Matrix Not Identity Matrix

\*/

#include <stdio.h>

int main()

{

int NRC, row, col, flag = 0;

scanf("%d", &NRC);

int MAT[NRC][NRC];

for(row = 0; row < NRC; row++)

for(col = 0; col < NRC; col++)

scanf("%d", &MAT[row][col]);

for(row = 0; row < NRC; row++)

{

for(col = 0; col < NRC; col++)

{

if( row == col && MAT[row][col] != 1)

flag++;

if( row != col && MAT[row][col] != 0)

flag++;

}

}

(flag == 0) ? printf("Identity Matrix") : printf("Not Identity Matrix");

return 0;

}

/\*

Check whether the given matrix is Upper Triangular matrix or not.

3 4

1 2 3 4

1 2 3 0 5 6 7

0 4 5 0 0 8 9

0 0 6 1 0 0 1

Upper Triangular Matrix Not Upper Triangular Matrix

\*/

#include <stdio.h>

int main()

{

int NRC, row, col, flag = 0;

scanf("%d", &NRC);

int MAT[NRC][NRC];

for(row = 0; row < NRC; row++)

for(col = 0; col < NRC; col++)

scanf("%d", &MAT[row][col]);

for(row = 0; row < NRC; row++)

{

for(col = 0; col < NRC; col++)

{

if( col < row && MAT[row][col] != 0 )

flag++;

}

}

( flag == 0 ) ? printf("Upper Triangular Matrix") : printf("Not Upper Triangular Matrix");

return 0;

}

/\*

Check whether the given matrix is lower Triangular matrix or not.

3 4

1 0 0 0

1 0 0 2 5 0 0

2 4 0 4 5 8 0

3 4 6 1 3 6 1

Lower Triangular Matrix Not Lower Triangular Matrix

\*/

#include <stdio.h>

int main()

{

int NRC, row, col, flag = 0;

scanf("%d", &NRC);

int MAT[NRC][NRC];

for(row = 0; row < NRC; row++)

for(col = 0; col < NRC; col++)

scanf("%d", &MAT[row][col]);

for(row = 0; row < NRC; row++)

{

for(col = 0; col < NRC; col++)

{

if( col > row && MAT[row][col] != 0 )

flag++;

}

}

( flag == 0 ) ? printf("Lower Triangular Matrix") : printf("Not Lower Triangular Matrix");

return 0;

}

/\*

Check whether the given matrix is Toeplitz matrix or not.

4 5 3

1 2 3 4 1 2 3 4 7 1 2 3

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

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

7 6 5 1 4 3 2 1 2

5 4 3 2 1

Toeplitz Toeplitz Not Toeplitz

\*/

#include <stdio.h>

int main()

{

int NRC, row, col, flag = 0;

scanf("%d", &NRC);

int MAT[NRC][NRC];

for(row = 0; row < NRC; row++)

for(col = 0; col < NRC; col++)

scanf("%d", &MAT[row][col]);

for(row = 0; row < NRC - 1; row++)

for(col = 0; col < NRC - 1; col++)

if( MAT[row][col] != MAT[row+1][col+1] )

flag++;

( flag == 0 ) ? printf("Toeplitz") : printf("Not");

return 0;

}

/\*

Addition two matrix

3 3

1 1 1 1 1 1

2 2 2 2 2 2

3 3 3 3 3 3

2 2 2

4 4 4

6 6 6

\*/

#include <stdio.h>

int main()

{

int NROW, NCOL, row, col;

scanf("%d %d", &NROW, &NCOL);

int MAT1[NROW][NCOL], MAT2[NROW][NCOL] , RES[NROW][NCOL];

for(row = 0; row < NROW; row++)

for(col = 0; col < NCOL; col++)

scanf("%d", &MAT1[row][col]);

for(row = 0; row < NROW; row++)

for(col = 0; col < NCOL; col++)

scanf("%d", &MAT2[row][col]);

for(row = 0; row < NROW; row++)

for(col = 0; col < NCOL; col++)

RES[row][col] = MAT1[row][col] + MAT2[row][col];

for(row = 0; row < NROW; row++, printf("\n"))

for(col = 0; col < NCOL; col++)

printf("%d ", RES[row][col]);

return 0;

}

/\*

Matrix Multiplication

3 3

3 3

1 1 1 1 1 1 (1+2+3) (1+2+3) (1+2+3) 6 6 6

2 2 2 2 2 2 (2+4+6) (2+4+6) (2+4+6) 12 12 12

3 3 3 3 3 3 (3+6+9) (3+6+9) (3+6+9) 18 18 18

6 6 6

12 12 12

18 18 18

\*/

#include <stdio.h>

int main()

{

int R1, C1, R2, C2, row, col, itr;

printf("Enter the first matrix row and col\n");

scanf("%d %d", &R1, &C1);

printf("Enter the second matrix row and col\n");

scanf("%d %d", &R2, &C2);

if( C1 != R2 )

{

printf("Cannot perform Multiplication");

return 0;

}

int MAT1[R1][C1], MAT2[R2][C2], RES[R1][C2];

printf("Enter MAT1 elements\n");

for(row = 0; row < R1; row++)

for(col = 0; col < C1; col++)

scanf("%d", &MAT1[row][col]);

printf("Enter MAT2 elements\n");

for(row = 0; row < R2; row++)

for(col = 0; col < C2; col++)

scanf("%d", &MAT2[row][col]);

int sum;

for(row = 0; row < R1; row++)

{

for(col = 0; col < C2; col++)

{

for(itr = 0, sum = 0; itr < R2; itr++)

{

sum = sum + ( MAT1[row][itr] \* MAT2[itr][col]);

}

RES[row][col] = sum;

}

}

printf("THE RESULTANT IS\n");

for(row = 0; row < R1; row++, printf("\n"))

for(col = 0; col < C2; col++)

printf("%d ", RES[row][col]);

return 0;

}

/\*

Transpose of a given matrix.

3 3

1 2 3

4 5 6

7 8 9

1 4 7

2 5 8

3 6 9

\*/

#include <stdio.h>

int main()

{

int NROW, NCOL, row, col;

scanf("%d %d", &NROW, &NCOL);

int MAT[NROW][NCOL], TRANS[NCOL][NROW];

for(row = 0; row < NROW; row++)

{

for(col = 0; col< NCOL; col++)

{

scanf("%d", &MAT[row][col]);

TRANS[col][row] = MAT[row][col];

}

}

for(row = 0; row < NCOL; row++, printf("\n"))

for(col = 0; col < NROW; col++)

printf("%d ", TRANS[row][col]);

return 0;

}

/\*

Transpose of a given matrix. (without using Additional array)(only for square matrix)

3 3

1 2 3

4 5 6

7 8 9

1 4 7

2 5 8

3 6 9

\*/

#include <stdio.h>

int main()

{

int NRC, row, col;

scanf("%d", &NRC);

int MAT[NRC][NRC];

for(row = 0; row < NRC; row++)

for(col = 0; col < NRC; col++)

scanf("%d", &MAT[row][col]);

for(row = 0; row < NRC; row++)

{

for(col = row+1; col < NRC; col++)

{

int temp = MAT[row][col];

MAT[row][col] = MAT[col][row];

MAT[col][row] = temp;

}

}

for(row = 0; row < NRC; row++, printf("\n"))

for(col = 0; col < NRC; col++)

printf("%d ", MAT[row][col]);

return 0;

}