PROBLEMS

# 1.

On matrix read from SI compute the difference of sum of elements of odd columns and sum of elements of even rows. Print the result.

# 2.

Write a program that for a given matrix read from SI will replace the elements from the main diagonal with the difference between the maximum and minimum element from the matrix. Print the result matrix.

# 3.

Write a program that will print on screen if a given matrix is symetric based on the main diagonal. Dimensions and the matrix are read from SI.

# 4.

# 5.

A squared matrix is read from SI. First, the number of rows and columns N is read, and then the N\*N elements of the matrix.

Change the sign of the elements located on the main diagonal (the positive numbers should become negative and reverse).

Print the transformed matrix on the screen (each element is printed with 3 places using %3d).

# 6.

A matrix with m rows and n columns is read. Firstly, the dimensions m and n are read, followed by the matrix elements.

Perform a min-max normalization of each column in the matrix i.e. each element in every column should be replaced with the value   where x is an element in a given column. Max and min are correspondingly the maximum and minimum elements in the corresponding column.

Print the transformed matrix on SO.

Explanation of the example:

The elements on the  0-th column are 1,6,11,16,21. Min is 1, max is 21.

Each element in the column should be transformed in the following way:

(1-1)/(21-1) = 0/20 = 0.00

(6-1)/(21-1) = 5/20 = 0.25

(11-1)/(21-1) = 10/20 = 0.50

(16-1)/(21-1) = 15/20 = 0.75

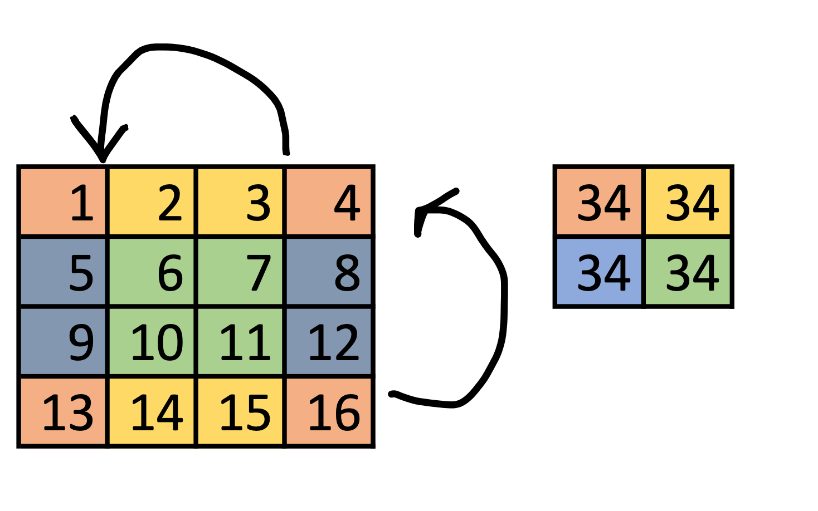
(21-1)/(21-1) = 20/20 = 1.00

# 7.

A squared matrix with dimension n is read from SI. Firstly the number n is read, followed by the matrix elements.

If n is an odd number, the message ERROR should be printed.

Otherwise, you need to **fold** the matrix as demonstrated in the figure below. When folding the matrix the elements in the matching positions are summed.



# 8.

Write a program that reads two integer matrices with dimension m x n, and then print how many columns of the first matrix are in the second matrix.

# 9.

From standard input, a positive number n is read, which gives the dimensions of a square matrix of integers, which is then read. Write a program that for that matrix will print the length of the longest strictly increasing substring, if we look at the matrix row by row.

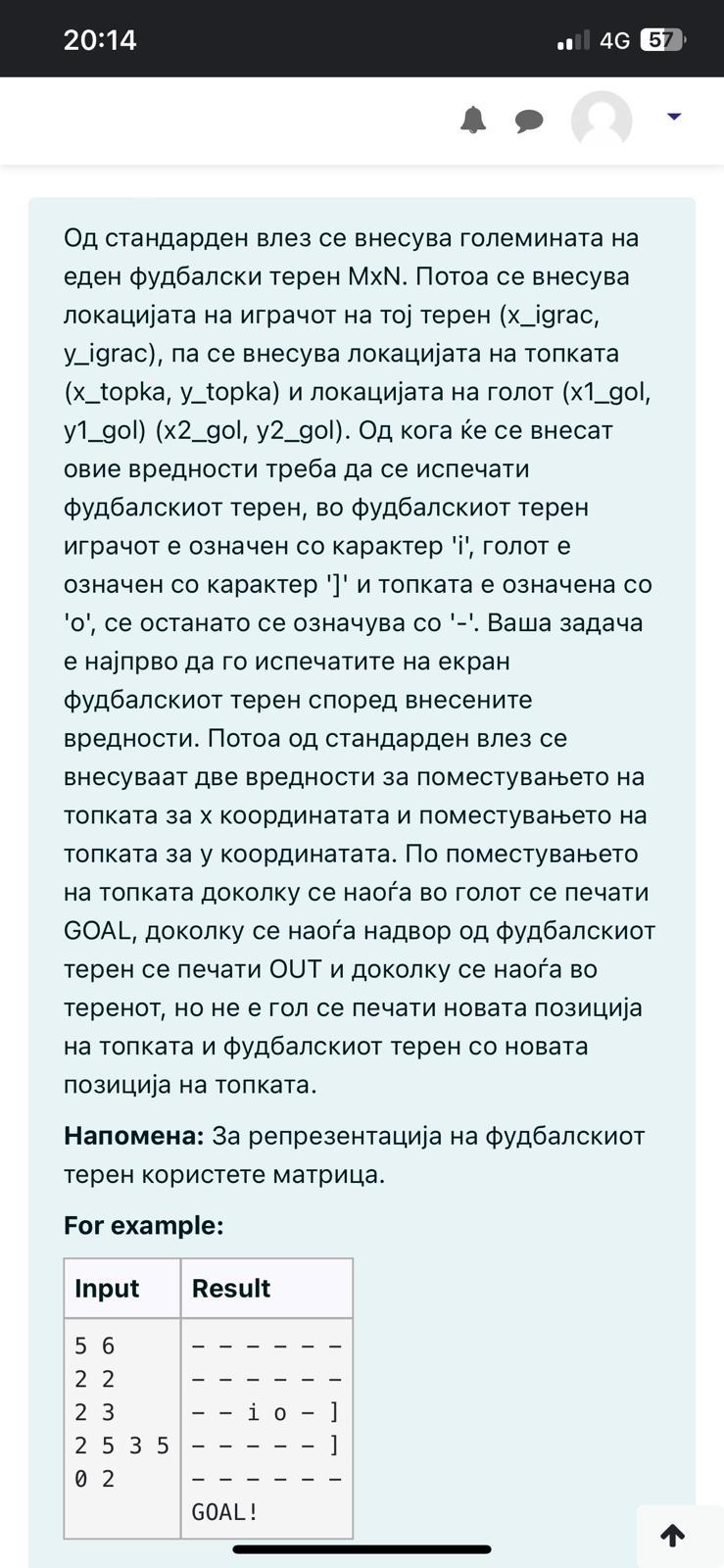
# 10.

From SI, the size of one football field M x N is entered. Then the location of the player on that field is entered (x\_player, y\_player), then the location of the ball is entered(x\_ball, y\_ball) and the location of the goal is entered (x1\_goal, y1\_goal)(x2\_goal, y2\_goal).

When these values are entered, the football field should be printed, in the football field the player is marked with the character 'p', the goal is marked with the character ']' and the ball is marked with 'o', everything else is marked with '- '. Your task is to first print the football field on the screen according to the entered values.

Then two values are entered from the standard input for the displacement of the ball for the x coordinate and the displacement of the ball for the y coordinate. After moving the ball, if it is in the goal, GOAL is printed, if it is outside the football field, OUT is printed, and if it is in the field, but it is not a goal, the position of the ball and the football field with the new position of the ball are printed.

Note: For the presentation of the football field, use a matrix.



SOLUTIONS

# 1.

*/\*On matrix read from SI compute the difference of sum of elements of odd columns and sum of elements of even rows. Print the result.\*/*#include **<stdio.h>  
  
int** main(){  
 **int** m,n;  
 scanf(**"%d%d"**,&m,&n);  
 **int** M[100][100];  
 **for**(**int** i=0 ; i<m ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 scanf(**"%d"**,&M[i][j]);  
 }  
 }  
 **int** sumOddC=0, sumEvenR=0;  
  
 **for**(**int** i=0 ; i<m ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 **if**(i%2==0){  
 sumEvenR+=M[i][j];  
 }  
 **if**(j%2==1){  
 sumOddC+=M[i][j];  
 }  
 }  
 }  
 printf(**"%d"**,sumOddC-sumEvenR);  
 **return** 0;  
}

# 2.

*/\*Write a program that for a given matrix read from SI will replace the elements from the main diagonal  
with the difference between the maximum and minimum element from the matrix. Print the result matrix.\*/*#include **<stdio.h>  
  
int** main(){  
 **int** n;  
 scanf(**"%d"**,&n);  
 **int** M[100][100];  
 **for**(**int** i=0 ; i < n ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 scanf(**"%d"**,&M[i][j]);  
 }  
 }  
 **int** min,max;  
 min=max=M[0][0];  
 **for**(**int** i=0 ; i < n ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 **if**(max<M[i][j]){  
 max=M[i][j];  
 }  
 **if**(min>M[i][j]){  
 min=M[i][j];  
 }  
 }  
 }  
 **for**(**int** i=0 ; i<n ; i++){  
 M[i][i]=max-min;  
 }  
  
 **for**(**int** i=0 ; i < n ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 printf(**"%d "**,M[i][j]);  
 }  
 printf(**"\n"**);  
 }  
 **return** 0;  
}

# 3.

*/\*Write a program that will print on screen if a given matrix is symetric based on the main diagonal.  
Dimensions and the matrix are read from SI.\*/*#include **<stdio.h>  
  
int** main(){  
 **int** n;  
 scanf(**"%d"**,&n);  
 **int** M[100][100];  
 **for**(**int** i=0 ; i < n ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 scanf(**"%d"**,&M[i][j]);  
 }  
 }  
 **int** symmetrical=1;  
 *// for (int i = 0; i < n; i++) {  
// for (int j = 0; j < n; j++) {  
// if (i!=j && matrix[i][j]!=matrix[j][i]){  
// symmetrical = 0;  
// break;  
// }  
// }  
// if (symmetrical==0){  
// break;  
// }  
// }* **for**(**int** i=0 ; i<n ; i++){  
 **for**(**int** j=i+1 ; j<n ; j++){  
 **if**(M[i][j]!=M[j][i]){  
 symmetrical=0;  
 **break**;  
 }  
 }  
 **if**(symmetrical==0){  
 **break**;  
 }  
 }  
 **if**(symmetrical){  
 printf(**"Symmetrical"**);  
 } **else**{  
 printf(**"Not symmetrical"**);  
 }  
 **return** 0;  
}

# 4.

# 5.

*/\*A squared matrix is read from SI. First, the number of rows and columns N is read, and then the N\*N elements of the matrix.  
Change the sign of the elements located on the main diagonal (the positive numbers should become negative and reverse).  
Print the transformed matrix on the screen (each element is printed with 3 places using %3d).\*/*#include **<stdio.h>  
  
int** main(){  
 **int** n;  
 scanf(**"%d"**,&n);  
 **int** M[100][100];  
 **for**(**int** i=0 ; i < n ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 scanf(**"%d"**,&M[i][j]);  
 }  
 }  
 **for**(**int** i=0 ; i<n ; i++){  
 M[i][i]\*=-1;  
 }  
 **for**(**int** i=0 ; i<n ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 printf(**"%d "**,M[i][j]);  
 }  
 printf(**"\n"**);  
 }  
 **return** 0;  
}

# 6.

*/\*A matrix with m rows and n columns is read. Firstly, the dimensions m and n are read, followed by the matrix elements.  
Perform a min-max normalization of each column in the matrix i.e. each element in every column should be replaced with the value (x-min)⁄(max-min) where x is an element in a given column. Max and min are correspondingly the maximum and minimum elements in the corresponding column.  
Print the transformed matrix on SO.  
Explanation of the example:  
The elements on the 0-th column are 1,6,11,16,21. Min is 1, max is 21.  
Each element in the column should be transformed in the following way:  
(1-1)/(21-1) = 0/20 = 0.00  
(6-1)/(21-1) = 5/20 = 0.25  
(11-1)/(21-1) = 10/20 = 0.50  
(16-1)/(21-1) = 15/20 = 0.75  
(21-1)/(21-1) = 20/20 = 1.00  
\*/*#include **<stdio.h>  
  
int** main(){  
 **int** m,n;  
 scanf(**"%d%d"**,&m,&n);  
 **int** M[100][100];  
 **for**(**int** i=0 ; i < m ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 scanf(**"%d"**,&M[i][j]);  
 }  
 }  
 **double** N[100][100],min,max;  
 **for**(**int** i=0 ; i<m ; i++){  
 min=max=M[0][i];  
 **for**(**int** j=0 ; j<n ; j++){  
 **if**(max<M[j][i]){  
 max=M[j][i];  
 }  
 **if**(min>M[j][i]){  
 min=M[j][i];  
 }  
 }  
 **for**(**int** j=0 ; j<n ; j++){  
 N[j][i]=(M[j][i] - min) / (max - min);  
 }  
 }  
  
 **for**(**int** i=0 ; i<n ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 printf(**"%.2lf "**,N[i][j]);  
 }  
 printf(**"\n"**);  
 }  
 **return** 0;  
}

# 7.

*/\*A squared matrix with dimension n is read from SI. Firstly the number n is read, followed by the matrix elements.  
If n is an odd number, the message ERROR should be printed.  
Otherwise, you need to fold the matrix as demonstrated in the figure below. When folding the matrix the elements in the matching positions are summed.\*/*#include **<stdio.h>  
  
int** main(){  
 **int** n;  
 scanf(**"%d"**,&n);  
 **if**(n%2!=0){  
 printf(**"ERROR"**);  
 **return** 0;  
 }  
 **int** M[100][100];  
 **for**(**int** i=0 ; i<n ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 scanf(**"%d"**,&M[i][j]);  
 }  
 }  
 **int** countR=0, countC=0;  
 **for**(**int** i=n-1 ; i>=n/2 ; i--){  
 **for**(**int** j=0 ; j<n ; j++){  
 M[countR][countC]+=M[i][j];  
 countC++;  
 }  
 countR++;  
 countC=0;  
 }  
 countC=countR=0;  
 **for**(**int** i=0 ; i<n/2 ; i++){  
 **for**(**int** j=n-1 ; j>=n/2 ; j--){  
 M[countR][countC]+=M[i][j];  
 countC++;  
 }  
 countR++;  
 countC=0;  
 }  
 **for**(**int** i = 0; i < n / 2; i++) {  
 **for**(**int** j = 0; j < n / 2; j++) {  
 printf(**"%d "**, M[i][j]);  
 }  
 printf(**"\n"**);  
 }  
 **return** 0;  
}

Another way

#include **<stdio.h>**

**int** main(){  
 **int** M[100][100], n, i, j;  
 scanf(**"%d"**, &n);  
 **if**(n%2 != 0){  
 printf(**"ERROR"**);  
 **return** 0;  
 }  
 **for**(i = 0; i < n; i++){  
 **for**(j = 0; j < n; j++){  
 scanf(**"%d"**, &M[i][j]);  
 }  
 }  
 **for**(i = 0; i < n/2; i++){  
 **for**(j = 0; j < n/2; j++){  
 M[i][j] += M[n-1-i][j];  
 M[i][j] += M[i][n-1-j];  
 M[i][j] += M[n-1-i][n-1-j];  
 printf(**"%d "**, M[i][j]);  
 }  
 printf(**"\n"**);  
 }  
 **return** 0;  
}

Another way

#include **<stdio.h>  
  
int** main(){  
 **int** n;  
 scanf(**"%d"**,&n);  
 **if**(n%2!=0){  
 printf(**"ERROR"**);  
 **return** 0;  
 }  
 **int** M[100][100];  
 **for**(**int** i=0 ; i<n ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 scanf(**"%d"**,&M[i][j]);  
 }  
 }  
 **for**(**int** i=0 ; i<n/2 ; i++){  
 **for**(**int** j=0 ; j<n/2 ; j++){  
 M[i][j]= M[i][j] + M[i][n - 1 - j] + M[n - 1 - i][n - 1 - j] + M[n - 1 - i][j];  
 }  
 }  
 **for**(**int** i=0 ; i<n/2 ; i++){  
 **for**(**int** j=0 ; j<n/2 ; j++){  
 printf(**"%d "**, M[i][j]);  
 }  
 printf(**"\n"**);  
 }  
 **return** 0;  
}

# 8.

*/\*Write a program that reads two integer matrices with dimension m x n,  
and then print how many columns of the first matrix are in the second matrix.\*/*#include **<stdio.h>  
  
int** main(){  
 **int** m,n;  
 scanf(**"%d%d"**,&m,&n);  
 **int** M1[100][100],M2[100][100];  
  
 **for**(**int** i=0 ; i<m ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 scanf(**"%d"**,&M1[i][j]);  
 }  
 }  
 **for**(**int** i=0 ; i<m ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 scanf(**"%d"**,&M2[i][j]);  
 }  
 }  
 **int** countC=0;  
 **for**(**int** i=0 ; i<m ; i++){  
  
 **int** tmpArray[100];  
  
 **for**(**int** j=0 ; j<n ; j++){  
 tmpArray[j]=M1[j][i];  
 }  
 **for**(**int** j=0 ; j<m ; j++){  
 **int** countElements=0;  
 **for**(**int** k=0 ; k<n ; k++){  
 **if**(tmpArray[k]==M2[k][j]){  
 countElements++;  
 }  
 }  
 **if**(countElements==n){  
 countC++;  
 }  
 }  
  
 }  
 printf(**"%d"**,countC);  
 **return** 0;  
}

# 9.

*/\*From standard input, a positive number n is read, which gives the dimensions of a square matrix of integers,  
which is then read. Write a program that for that matrix will print the length of the longest strictly increasing substring,  
if we look at the matrix row by row.\*/*#include **<stdio.h>**#include **<ctype.h>**#include **<string.h>  
  
int** main(){  
 **int** n;  
 scanf(**"%d"**,&n);  
 **int** matrix[100][100];  
 **for**(**int** i=0 ; i<n ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 scanf(**"%d"**,&matrix[i][j]);  
 }  
 }  
 **int** start=0,end=0,maxL=1,row=0;  
 **int** i,j,k;  
 **for**( i=0 ; i<n ; i++){  
 **for**( j=0 ; j<n ; j++){  
 **for**( k=j ; k<n-1 ; k++){  
 **if**(matrix[i][k]>matrix[i][k+1]){  
 **break**;  
 }  
 }  
 **int** length=k-j+1;  
 **if**(length>maxL){  
 maxL=length;  
 start=j;  
 end=k;  
 row=i;  
 }  
 j=k;  
 }  
 }  
  
 printf(**"Start: (%d, %d)\nEnd: (%d, %d)\nLength: %d"**,row,start,row,end,maxL);  
  
 **return** 0;  
}

# 10.

#include **<stdio.h>  
  
int** main(){  
 **int** m,n;  
 scanf(**"%d%d"**,&m,&n);  
 **int** x\_player,y\_player;  
 scanf(**"%d%d"**,&x\_player,&y\_player);  
 **int** x\_ball,y\_ball;  
 scanf(**"%d%d"**,&x\_ball,&y\_ball);  
 **int** x1\_goal,x2\_goal,y1\_goal,y2\_goal;  
 scanf(**"%d%d%d%d"**,&x1\_goal,&y1\_goal,&x2\_goal,&y2\_goal);  
 **char** footballField[100][100];  
 **for**(**int** i=0 ; i<m ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 footballField[i][j]=**'-'**;  
 }  
 }  
 footballField[x\_player][y\_player]=**'p'**;  
 footballField[x\_ball][y\_ball]=**'o'**;  
 footballField[x1\_goal][y1\_goal]=**']'**;  
 footballField[x2\_goal][y2\_goal]=**']'**;  
 **for**(**int** i=0 ; i<m ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 printf(**"%c"**,footballField[i][j]);  
 }  
 printf(**"\n"**);  
 }  
 **int** x\_displacement\_ball,y\_displacement\_ball;  
 scanf(**"%d%d"**,&x\_displacement\_ball,&y\_displacement\_ball);  
 **if**(x\_ball+x\_displacement\_ball>=m || y\_ball+y\_displacement\_ball>=n){  
 printf(**"OUT!"**);  
 **return** 0;  
 }  
 **char** array[1000];  
 **int** k=0;  
 **for**(**int** i=0 ; i<m ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 array[k++]=footballField[i][j];  
 }  
 }  
 **int** t,h;  
 **for**(**int** i=0 ;; i++){  
 **if**(array[i]==**']'**){  
 t=i;*//first post* **break**;  
 }  
 }  
 **for**(**int** i=t+1 ;; i++){  
 **if**(array[i]==**']'**){  
 h=i;*//second post* **break**;  
 }  
 }  
 **int** z=0;  
 footballField[x\_ball+x\_displacement\_ball][y\_ball+y\_displacement\_ball]=**'g'**;  
 k=0;  
 **for**(**int** i=0 ; i<m ; i++){  
 **for**(**int** j=0 ; j<n ; j++){  
 array[k++]=footballField[i][j];  
 }  
 }  
 **for**(**int** i=0 ;; i++){  
 **if**(array[i]==**'g'**){  
 z=i;*//ball dsiplacement* **break**;  
 }  
 }  
 **if**(z>=t && z<=h){  
 printf(**"GOAL!"**);  
 } **else**{  
 printf(**"%d, %d"**,x\_ball+x\_displacement\_ball,y\_ball+y\_displacement\_ball);  
 }  
 **return** 0;  
}