8/6/23

Date / /

LAB - 0

1) Program to Multiply, Add, Subtract, Transpose, symmetry and diagonal sum.

A)
```
#include<stdio.h>
void multi(int mat1[][100], mat2[][100], mult[][100], int m,
                int n, int p, int q);
void add( int mat1[][100], mat2[][100], int add[][100], int m, int n)
void sub add (int mat1[][100], mat2[][100], int sub[][100], int m, int n)
void transpose( int mat[][100], int trans[][100] , int m, int n)
void symmetry( int mat1[][100], int m , int n);
void diagonal( int mat1[][100], int m, int n);

int main()
{
   int mat1[100][100], mat2[100][100] , add[100][100], sub[100][100],
   trans1[100][100], trans2[100][100], int m, n, p,q; i, j;

  printf("\nEnter the rows of first matrix:");
  scanf("%d %d", &m, &n);
  printf("\n Enter the rows and columns of 2nd matrix");
   scanf("%d %d", &p, &q);

  printf("\n Enter the first matrix elements");
   for(i=0; i<m; i++)
  { for(j=0; j<n; j++)
       scanf("%d", &mat1[i][j]);
   }

  printf("\n Enter the second matrix elements");
   for(i=0; i<p; i++)
  {  for(j=0; j<q; j++)
       scanf("%d", &mat2[i][j]);
   }
```

```
if ( n != r)
  printf("\n Matrix cannot be multiplied");
else
{  multi( mat1, mat2, mult, m, n, p, q)
   printf("\n Multiplication Result : \n");
   for (i=0; i<m; i++)
  { for (j=0; j<q; j++)
      printf("%d", mult[i][j]);
    printf("\n");
  }

   add( mat1, mat2, add , m, ~~p~~n)
 { printf("\n Addition Result : \n");
   for (i=0; i<m; i++)
 { for (j=0; j<~~p~~n; j++)
     printf("%d", add[i][j]);
   printf("\n");
 }

  sub( mat1, mat2, sub, m, n )
  printf("\n Subtraction Result : \n");
  for (i=0; i<m; i++)
 { for (j=0; j<n; j++)
   printf("%d", sub[i][j]);
   printf("\n");
 }

 transpose( mat1, trans1, m, n)
  printf("\n Transpose Matrix 1 \n");
  for (i=0; i<m; i++)
 { for (j=0; j<n; j++)
   printf("%d", trans[i][j]);
   printf("\n");
 }
```

```
  transpose( mat2, trans2, p,q)
 printf("\n Transpose Matrix 2\n");
   for(i=0; i<p; i++)
 {  for(j=0; j<q; j++)
      printf("%d", trans2[i][j]);
    printf("\n");
 }

  symmetry( mat1, m, n);
  symmetry( mat2, p,q);

  diagonal( mat1, m,n);
  diagonal( mat2, p,q);
  }
  return 0;
 }

 void multi( int mat1[][100], int mat2[][100], int mult[][100]
                  int m, int n, int p, int q)
 {  int i, j, k;
    for(i=0; i<m; i++)
  {  for(j=0; j<q; j++)
       mult[i][j] =0;
     }

   for(i=0; i<m; i++)
  {  for(j=0; j<q; j++)
   {  for(k=0; k<n; k++)
       mult[i][j] = mat1[i][k] + mat2[k][j];
     }
   }
```

```
void add( int mat1[][100], int mat2[][100], int add sub[][100],
                int m, int n)

{  int i,j;
   for(i=0; i<m; i++)
  {  for(j=0; j<n; j++)
       add[i][j]= mat1[i][j]+ mat2[i][j];
   }
}

void sub( int mat1[][100], int mat2[][100], int sub[][100],
              int m, int n)
{  int i,j;
   for(i=0; i<m; i++)
  {  for(j=0; j<n; j++)
       sub[i][j] = mat[i][j] - mat2[i][j];
   }
}

void transpose( int mat[][100], int trans[][100], int m, int n)
{  int i,j;
   for(i=0; i< n; i++)
  {  for(j=0; j<m; j++)
       trans[i][j]= mat[j][i],
   }
}

  void symmetry( int mat[][100], int m, int n)
{  int i,j;
   int flag=1;
   if (m!=n)
  {  printf("\n Matrix is unsymmetric");
     return; }
   for (i=0; i<m; i++)
  {  for(j=0; j<n; j++)
{  if (mat[i][j] != mat[j][i])
     { flag = 0;
       break; }
   }
   if (flag == 0)
     break;
 }
 if (flag == 1)
     printf("\n Matrix is symmetrical");
 else
     printf("\n Matrix is unsymmetric");
}

void rowcolumnsum( int mat[][100], int m, int n)
{  int i, j, a, b;
   printf("\n Sum of Each row");
   for (i=0; i<m; i++)
   {  a = 0;
      for (j=0; j<n; j++)
         a += mat[i][j];
      printf("%d", a);
   }
   printf("\n");

   printf("\n Sum of each column");
   for (i=0; i<n; i++)
   {  b = 0;
      for (j=0; j<m; j++)
         b += mat[j][i];
      printf("%d", b);
   }
   printf("\n");
}
```

```
void diagonal( int mat[][100], int m, int n)
{ int i, j, a=0, b=0;
  if (m!=n)
{  printf(" Matrix is not square matrix! ");
   return; }

 for(i=0; i<m; i++)
{  for(j=0; j<n; j++)
{   if(i==j)
      a+= mat[i][j];
   if( i+j == ~~row~~ m -1)
      b+= mat[i][j];

 }
 }
} printf("Principal Diagonal Sum:%d", a);
 printf("Non-Principal Diagonal Sum: %d", b);
 }
```

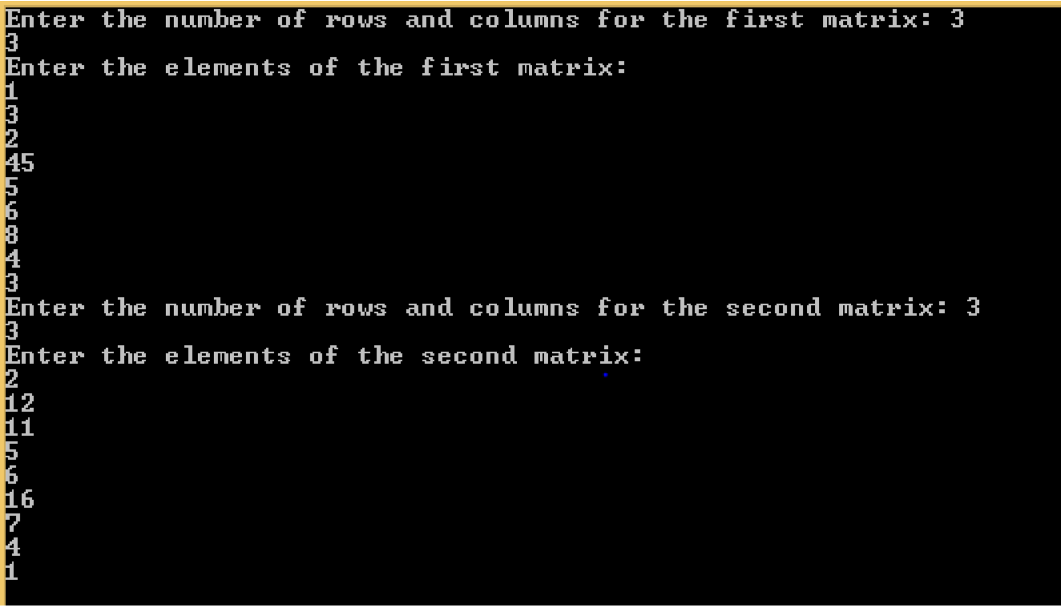
main function - X
incomplete

Output:

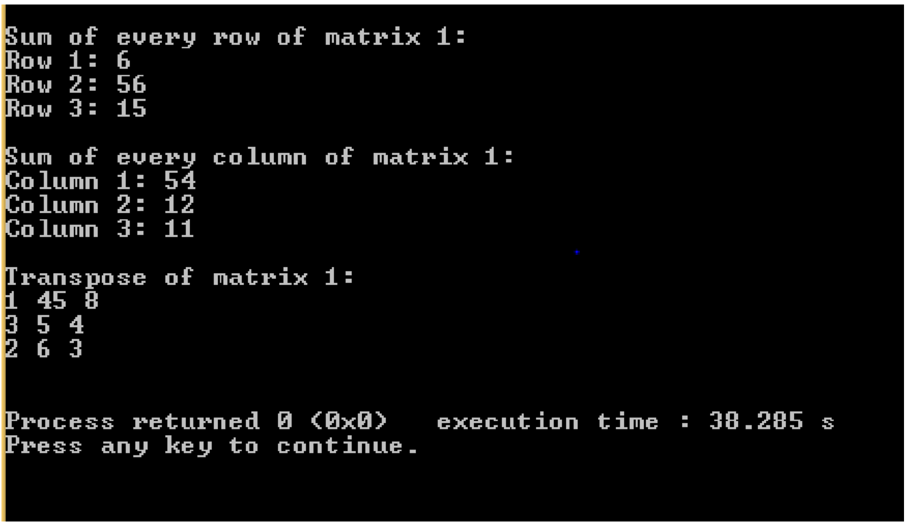
```
Enter number of rows and colums of matrix 1:  3 3
Enter number of rows and colums of matrix 2:  3 3

Enter elements of first matrix:
 1  3  2
 45  5  6
 8  4  3

Enter elements of second matrix:

 2  12  11
 3  6  16
 7  4  1
```

Date / / /

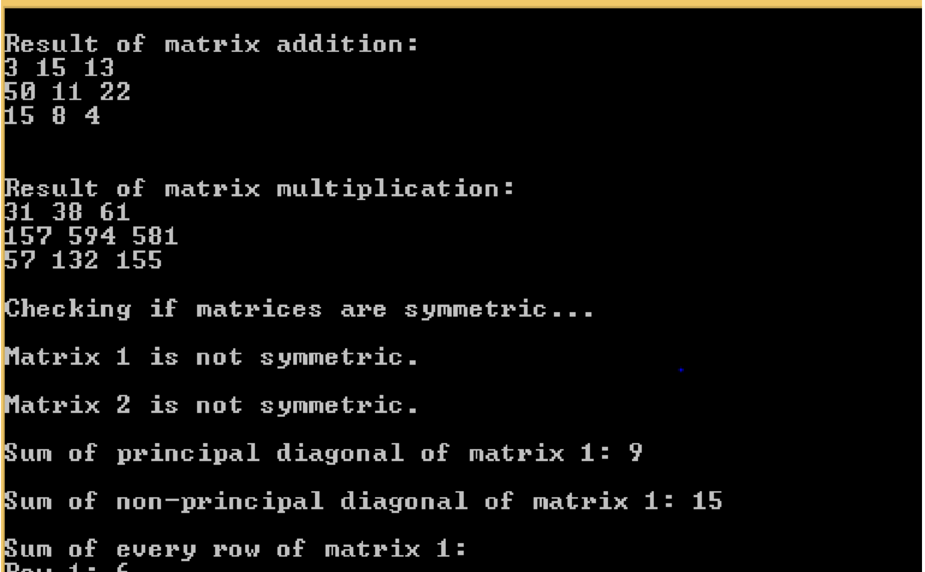
```
Result of matrix addition:
  3   15  13
  50  11  22
  15   8   4

Result of matrix multiplication:
 37  38  61
157 594 581
 57 132 155

Checking if matrices are symmetric...

Matrix 1 isnot symmetric
Matrix 2 is not symmetric.

Sum of principal diagonal of matrix 1: 9
sum of non-principal diagonal of matrix 1: 15

Sum of rows of matrix 1:
  6  56  15

Sum of columns of matrix 1:
 54  12  11

Transpose of matrix
  1  45  8
  3   5  4
  2   6  3
```



```
Enter the number of rows and columns for the first matrix: 3
3
Enter the elements of the first matrix:
1
3
2
45
5
6
8
4
3
Enter the number of rows and columns for the second matrix: 3
3
Enter the elements of the second matrix:
2
12
11
5
6
16
7
4
1
```



```
Sum of every row of matrix 1:
Row 1: 6
Row 2: 56
Row 3: 15

Sum of every column of matrix 1:
Column 1: 54
Column 2: 12
Column 3: 11

Transpose of matrix 1:
1 45 8
3 5 4
2 6 3


Process returned 0 (0x0)   execution time : 38.285 s
Press any key to continue.
```



```
Result of matrix addition:
3 15 13
50 11 22
15 8 4


Result of matrix multiplication:
31 38 61
157 594 581
57 132 155

Checking if matrices are symmetric...

Matrix 1 is not symmetric.

Matrix 2 is not symmetric.

Sum of principal diagonal of matrix 1: 9

Sum of non-principal diagonal of matrix 1: 15

Sum of every row of matrix 1:
Row 1: 6
```