Practical 02

Name: Shruti Rajesh Kumbhare

Roll no: 11

Batch: B1

Aim: To study array ADT and to implement various operations on matrix(addition, multiplication, transpose, row major, column major)

Code:

```
#include<stdio.h>

struct Matrix{

   int row,column,arr[10][10];

};

void rowmajor(struct Matrix M)

{

   printf("no. of rows?\n");

   scanf("%d",&M.row);

   printf("no. of columns?\n");

   scanf("%d",&M.column);

   printf("enter elements\n");

   for(int i=0;i<M.row;i++)

   {

      for(int j=0;j<M.column;j++)

      {

         scanf("%d",&M.arr[i][j]);

      }

   }  printf("row major order :\n");

   for(int i=0;i<M.row;i++)

   {

      for(int j=0;j<M.column;j++)

      {

         printf("%d ",M.arr[i][j]);

      }
```

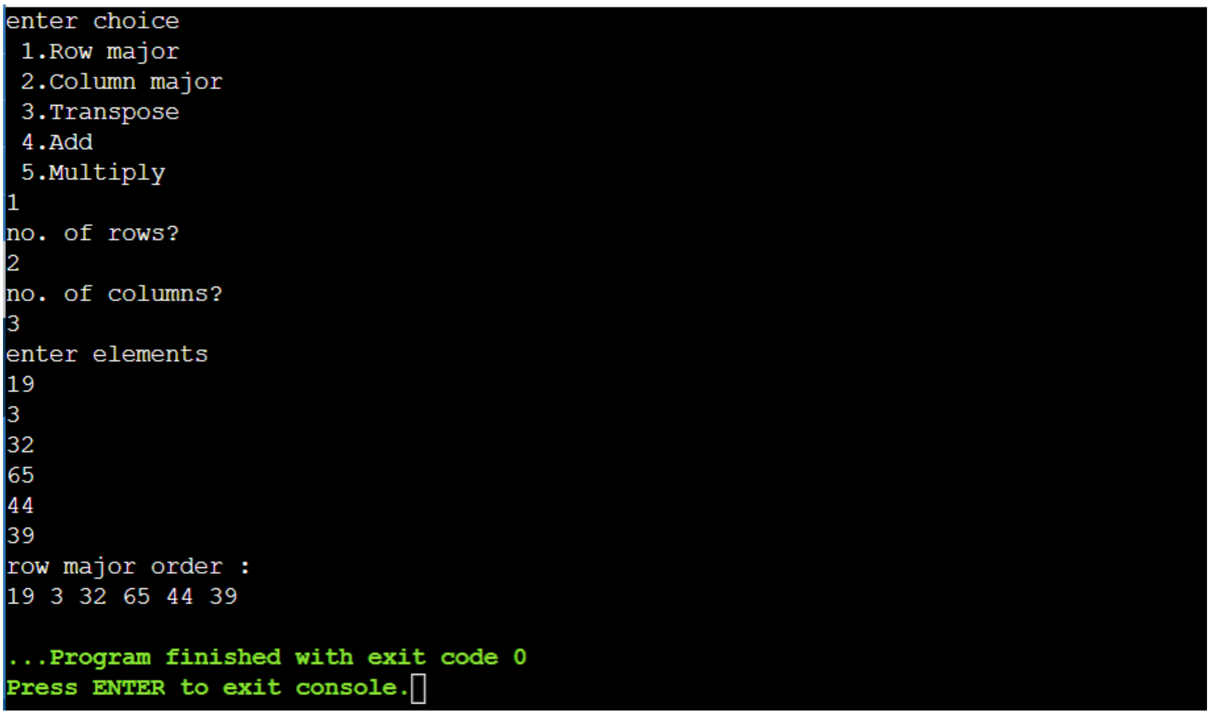
```
    }
}
void columnmajor(struct Matrix M)
{
    printf("no. of rows?\n");
    scanf("%d",&M.row);
    printf("no. of columns?\n");
    scanf("%d",&M.column);
    printf("enter elements\n");
    for(int i=0;i<M.row;i++)
    {
        for(int j=0;j<M.column;j++)
        {
            scanf("%d",&M.arr[i][j]);
        }
    }
    printf("column major order :\n");
    for(int i=0;i<M.row;i++)
    {
        for(int j=0;j<M.column;j++)
        {
            printf("%d ",M.arr[j][i]);
        }
    }
}
void transpose(struct Matrix M)
{   int t[5][5];
    printf("no. of rows?\n");
    scanf("%d",&M.row);
    printf("no. of columns?\n");
    scanf("%d",&M.column);
    printf("enter elements\n");
    for(int i=0;i<M.row;i++)
    {
        for(int j=0;j<M.column;j++)
        {
            scanf("%d",&M.arr[i][j]);
        }
    }
    for(int i=0;i<M.row;i++)
    {
        for(int j=0;j<M.column;j++)
        {
            t[i][j]=M.arr[j][i];
        }

    }
    printf("transpose:\n");
    for(int i=0;i<M.row;i++)
    {
        for(int j=0;j<M.column;j++)
        {
            printf("%d ",t[i][j]);
        }
    printf("\n");
    }
}
void add(struct Matrix M,struct Matrix N)
{   int P[5][5];
    printf("no. of rows?\n");
    scanf("%d",&M.row);
    printf("no. of columns?\n");
```

```
scanf("%d",&M.column);
printf("enter elements of M\n");
for(int i=0;i<M.row;i++)
{
    for(int j=0;j<M.column;j++)
    {
        scanf("%d",&M.arr[i][j]);
    }
}
printf("enter elements of N\n");
for(int i=0;i<M.row;i++)
{
    for(int j=0;j<M.column;j++)
    {
        scanf("%d",&N.arr[i][j]);
    }
}

for(int i=0;i<M.row;i++)
{
    for(int j=0;j<M.column;j++)
    {
        P[i][j]=M.arr[i][j]+N.arr[i][j];
    }


}
printf("added matrix:\n");
for(int i=0;i<M.row;i++)
{
    for(int j=0;j<M.column;j++)
    {
         printf("%d ",P[i][j]);
      }
      printf("\n");

   }
}
void multiply(struct Matrix M, struct Matrix N)
{
  int P[5][5];
   printf("no. of rows of M?\n");
   scanf("%d",&M.row);
   printf("no. of columns  of M?\n");
   scanf("%d",&M.column);
   printf("enter elements of M\n");
   for(int i=0;i<M.row;i++)
   {
      for(int j=0;j<M.column;j++)
      {
         scanf("%d",&M.arr[i][j]);
      }
   }

   printf("no. of rows  of N?\n");
   scanf("%d",&N.row);
   printf("no. of columns of N?\n");
   scanf("%d",&N.column);
   printf("enter elements of N\n");
   for(int i=0;i<N.row;i++)
   {
      for(int j=0;j<N.column;j++)
      {
        scanf("%d",&N.arr[i][j]);
    }
}
if(M.column!=N.row)
{
    printf(" multiplication cannot be done,\n");


}
else{
    for(int i=0;i<M.row;i++)
    {
        for(int j=0;j<N.column;j++)
        {
            P[i][j]=0;
            // Multiplying i’th row with j’th column
            for(int k=0;k<M.row;k++)
            {
                P[i][j]+=M.arr[i][k]*N.arr[k][j];
            }
        }
    }
     printf("Multiplied matrix\n");
    for(int i=0;i<M.row;i++)
    {
        for(int j=0;j<N.column;j++)
        {
            printf("%d ",P[i][j]);
        }
        printf("\n");
    }
}
```

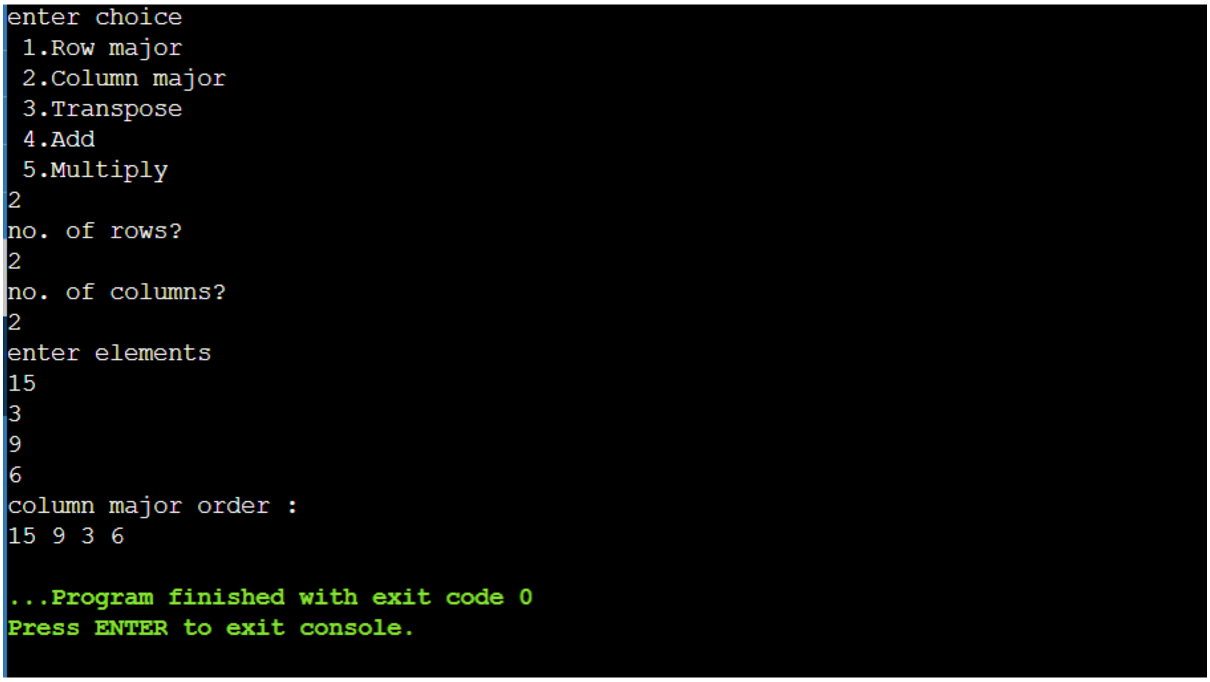
```
}
int main()
{   int choice;
     printf("enter choice\n 1.Row major\n 2.Column major\n 3.Transpose\n 4.Add\n 5.Multiply\n");
     scanf("%d",&choice);
    struct Matrix M;
    struct Matrix N;
    switch(choice)
    {
        case 1:
        {
          rowmajor(M);
          break;
        }
        case 2:
        {
            columnmajor(M);
            break;
        }
        case 3:
        {
            transpose(M);
            break;
        }
        case 4:
        {
            add(M,N);
            break;
        }
        case 5:
        {
```

```
      multiply(M,N);

      break;

    }

    default:

    {

      printf("invalid choice");

      break;

    }

  }


}
```

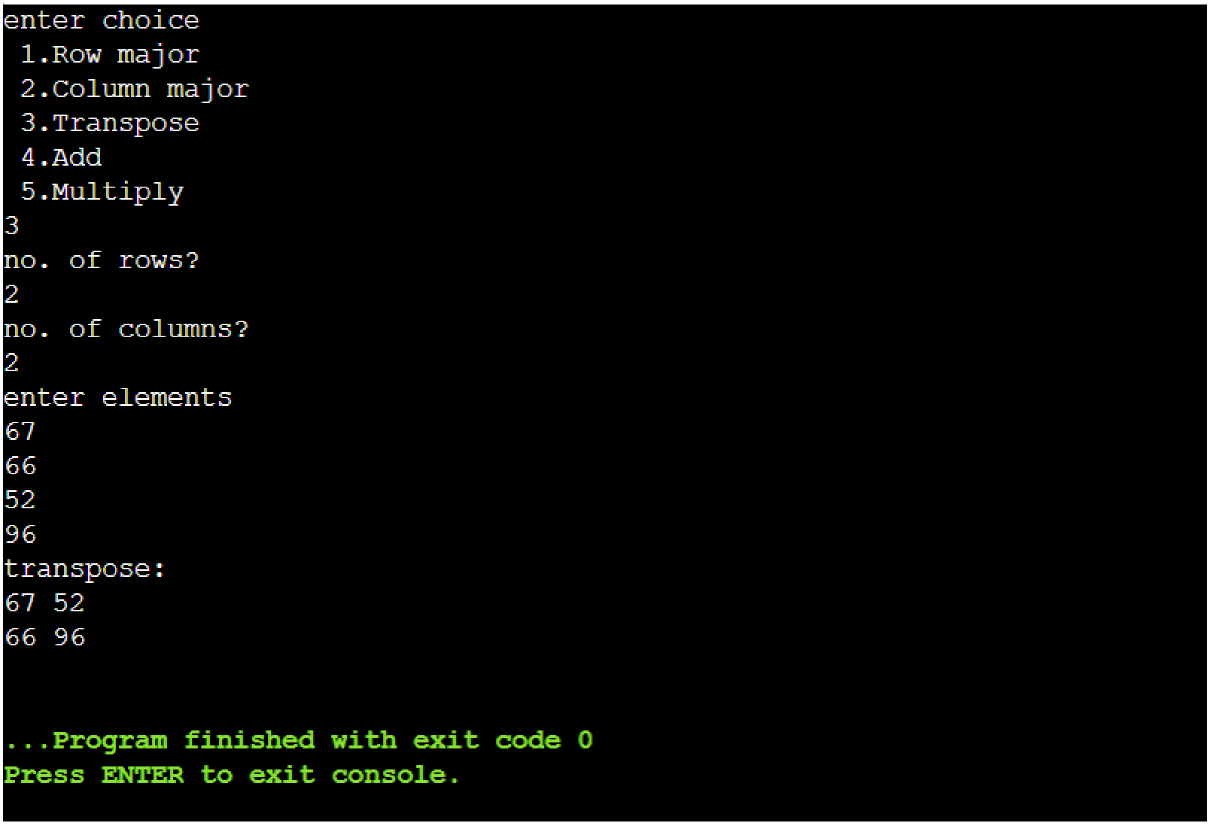
Output:



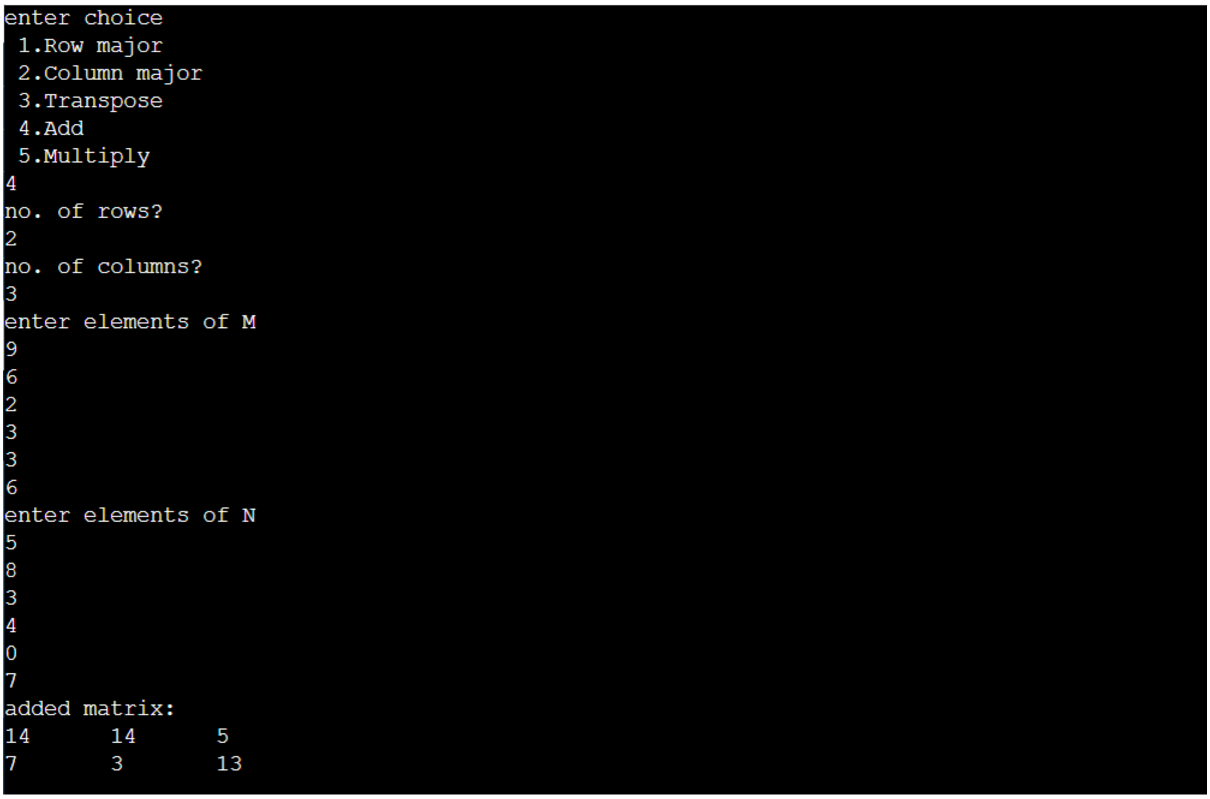
```
enter choice
 1.Row major
 2.Column major
 3.Transpose
 4.Add
 5.Multiply
1
no. of rows?
2
no. of columns?
3
enter elements
19
3
32
65
44
39
row major order :
19 3 32 65 44 39

...Program finished with exit code 0
Press ENTER to exit console.
```



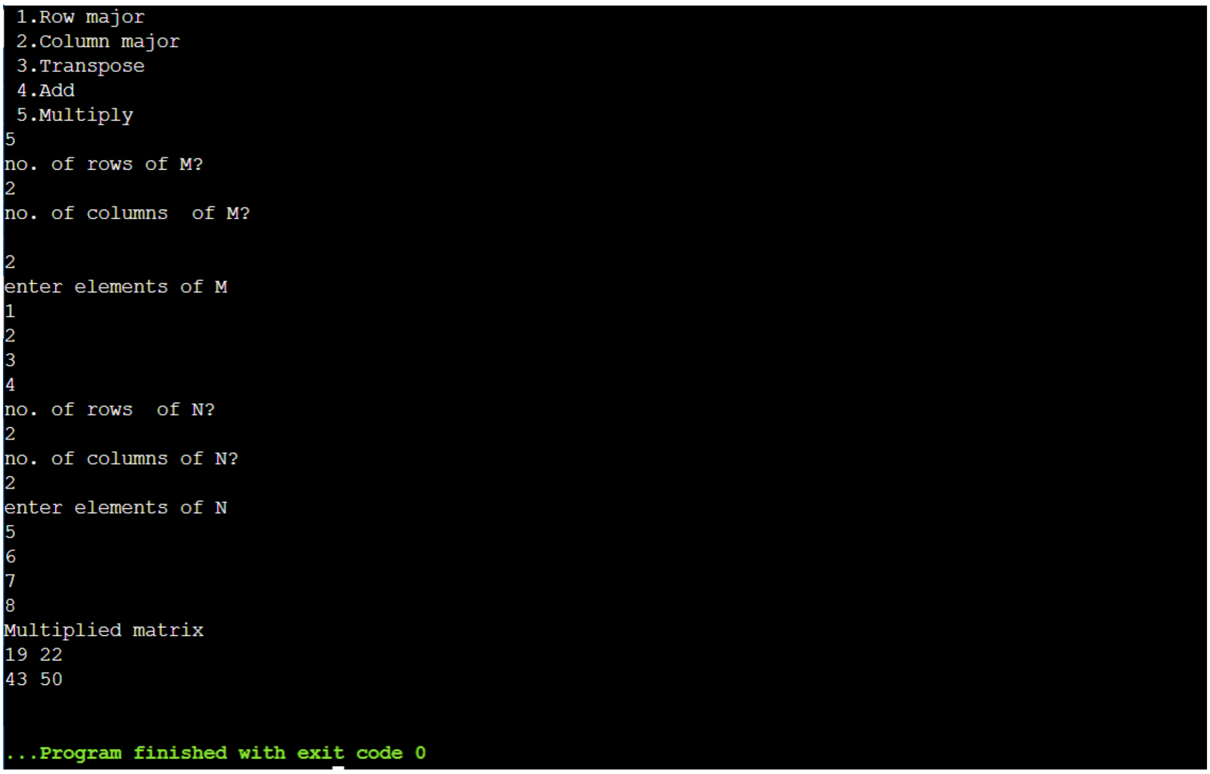
```
enter choice
 1.Row major
 2.Column major
 3.Transpose
 4.Add
 5.Multiply
2
no. of rows?
2
no. of columns?
2
enter elements
15
3
9
6
column major order :
15 9 3 6

...Program finished with exit code 0
Press ENTER to exit console.
```



```
enter choice
 1.Row major
 2.Column major
 3.Transpose
 4.Add
 5.Multiply
3
no. of rows?
2
no. of columns?
2
enter elements
67
66
52
96
transpose:
67 52
66 96


...Program finished with exit code 0
Press ENTER to exit console.
```



```
enter choice
 1.Row major
 2.Column major
 3.Transpose
 4.Add
 5.Multiply
4
no. of rows?
2
no. of columns?
3
enter elements of M
9
6
2
3
3
6
enter elements of N
5
8
3
4
0
7
added matrix:
14      14      5
7       3       13
```



```
 1.Row major
 2.Column major
 3.Transpose
 4.Add
 5.Multiply
5
no. of rows of M?
2
no. of columns  of M?

2
enter elements of M
1
2
3
4
no. of rows  of N?
2
no. of columns of N?
2
enter elements of N
5
6
7
8
Multiplied matrix
19 22
43 50


...Program finished with exit code 0
```