**ASSIGNMENT – 14**

**1.PROBLEM STATEMENT**

Write a program in C to implement matrix multiplication

**2.ALGORITHMS**

Algorithm **Matrix\_Multiplication**

**Input:** The pointer to the matrices ‘a1’,’a2’ and the result matrix ‘res’,the dimensions of the first matrix ‘row1’,’col1’ and then dimensions of the second matrix ‘row2’,’col2’.

**Output:**The product of the matrices ‘a1’ and ‘a2’ stored in ‘a3’.

**Remarks:** Matrix multiplication can be performed only if the number of columns of the first matrix is equal to the number of rows of the second matrix. The resultant matrix has number of rows equal to that of the first matrix and number of columns to that of second matrix.

**Steps:**

1. sum=0 //setting sum variable to zero
2. **For**(i=1 to row1) **do** //traversing along rows of first matrix
3. **For**(count=0 to col2) **do** //multiplying one row with all columns
4. sum=0
5. **For**(j=1 to row2) **do** //traversing rows of second matrix
6. sum=sum+(a1[i][j]\*a2[j][cnt]) //calculating sum
7. res[i][cnt]=sum // storing sum in result matrix
8. **Return**
9. **Stop**

**3.SOURCE CODE**

#include<stdio.h>

#include<stdlib.h>

//function to multiply two input matrices

void multiply(int(\*arrone)[20],int(\*arrtwo)[20],int(\*res)[20],int rowone,int colone,int rowtwo,int coltwo)

{

int i,j,cnt,sum=0;

for(i=0;i<rowone;i++) //traversing along the rows of first matrix

{

for(cnt=0;cnt<coltwo;cnt++)//multiplying row with all columns

{

sum=0;

for(j=0;j<rowtwo;j++) //traversing columns of second matrix

{

sum=sum+(arrone[i][j]\*arrtwo[j][cnt]);//finding sum

}

res[i][cnt]=sum;//storing result for one pass in res matrix

}

}

}

//function to display the resultant matrix

void dispres(int(\*res)[20],int rowone,int coltwo)

{

int i,j;

for(i=0;i<rowone;i++)

{

for(j=0;j<coltwo;j++)

printf("%d\t",res[i][j]);

printf("\n");

}

}

//function to take input in a matrix

void getmat(int(\*arr)[20],int row,int col)

{

int i,j;

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

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

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

}

//function to check if given matrices can be multiplied or not

void validate(int colone,int rowtwo)

{

if(colone!=rowtwo)

{

printf("Given matrices cannot be multiplied....Please try again");

exit(0);

}

}

int main(void)

{

int arrone[20][20],arrtwo[20][20],res[20][20],rowone,colone,rowtwo,coltwo;

printf("Enter the number of rows of the first matrix: ");

scanf("%d",&rowone);

printf("Enter the number of columns of the first matrix: ");

scanf("%d",&colone);

printf("Enter the number of rows of the second matrix: ");

scanf("%d",&rowtwo);

printf("Enter the number of columns of the second matrix: ");

scanf("%d",&coltwo);

validate(colone,rowtwo); //input validation

printf("Enter the elements of the first matrix: \n");

getmat(arrone,rowone,colone);

printf("Enter the elements of the second matrix: \n");

getmat(arrtwo,rowtwo,coltwo);

multiply(arrone,arrtwo,res,rowone,colone,rowtwo,coltwo);

printf("\nRESULT: \n");

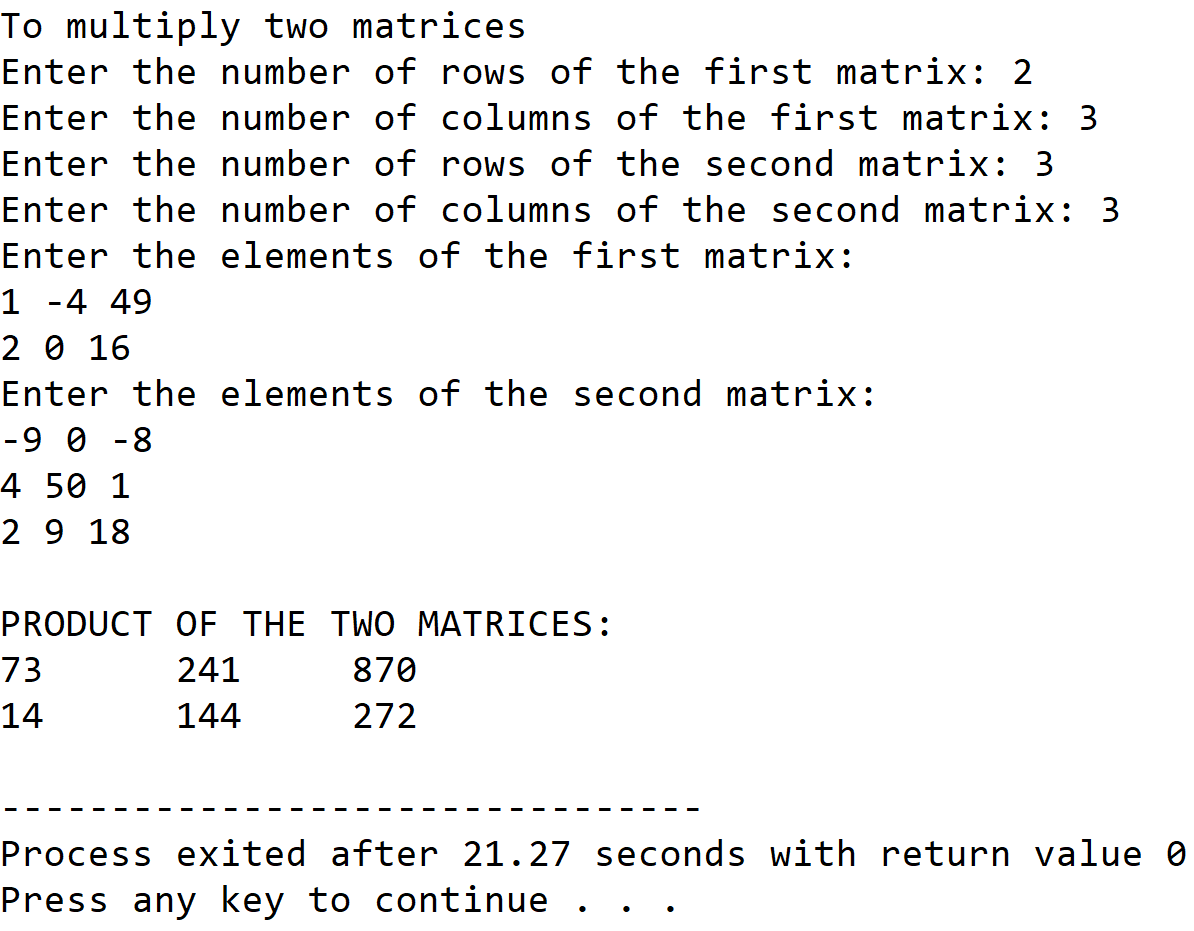
dispres(res,rowone,coltwo);

return 0;

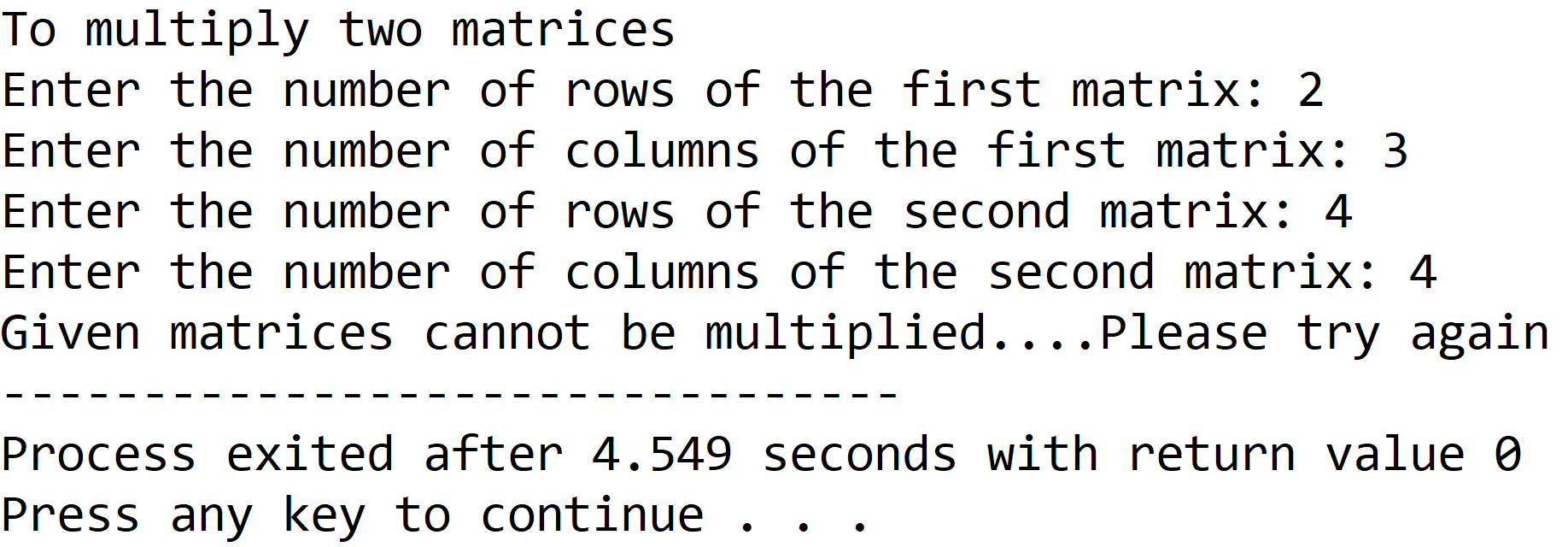
}

**4.OUTPUT**

**SET 1:** Valid Input



**SET 2:** Invalid Input



**5.DISCUSSIONS**

**Variable Description**

* **arrone:** 2-D array to hold first input matrix.
* **arrtwo:** 2-D array to hold second input matrix.
* **res:** 2-D array to hold the result of product of the input matrices.
* **rowone:** number of rows of the first matrix.
* **colone:** number of columns of the first matrix.
* **rowtwo:** number of rows of the second matrix.
* **coltwo:** number of columns of the second matrix.
* **sum:** variable for calculation of product.
* **i,j,cnt:** loop counters.

**Uses**

* The program can be used to calculate the product of any two matrices holding integer values. The program can find application in fields of data science wherever matrix multiplication is needed.

**Limitations**

* The two dimensional arrays used in this program are not dynamically allocated and can lead to either wastage, or lack of required memory locations.
* The product of the two matrices can also be stored in a one dimensional array that requires much less memory than a two dimensional array.

**Future Scope**

* Dynamic memory allocation can be used to construct the arrays.
* The data structure for storing the product can be changed to one dimensional arrays or even linked lists.

**Teacher’s Signature**