**ASSIGNMENT – 12**

**1.PROBLEM STATEMENT**

Write a program in C to implement symmetric matrix using one dimensional array

**2.ALGORITHMS**

Algorithm **Is\_Symmetric\_Matrix**

**Input:** The matrix arr[row][col] with its dimensions ‘row’ and ‘col’.

**Output:** If the entered matrix arr[row][col] is not a symmetric matrix, an error message is shown and a flag variable indicates invalid input.

**Remarks:** It is assumed that the matrix is not empty.

**Steps:**

1. **For**(i=1 to row) **do** //traversing the matrix
2. **For**(j=1 to col) **do**
3. **If**(i≠j) **then** //for non diagonal elements
4. **If**(arr[i][j]≠arr[j][i]) **then** //unequal transpose values
5. **Print** “Input matrix is not symmetric
6. **Exit** //terminate the procedure
7. **EndIf**
8. **EndIf**
9. **EndFor**
10. **EndFor**

Algorithm **Symmetric\_Matrix\_To\_1D\_Array**

**Input:** The matrix arr[row][col] an the list in which the symmetric matrix is to be stored.

**Output:** The upper triangular matrix stored in the array representing the symmetric matrix.

**Remarks:** Memory needed in the 1-D array: row\*(row+1)/2

**Steps:**

1. k=0
2. **For**(i=0 to row) **do** //traversing the matrix
3. **For**(j=1 to col) **do**
4. list[k]=arr[i][j] /mapping matrix linearly to array
5. k=k+1
6. **EndFor**
7. **EndFor**

**3.SOURCE CODE**

#include<stdio.h>

#include<stdlib.h>

//function for input validation

void validate(int row,int col)

{

    if(row!=col)

    {

        printf("Triangular Matrix Must Be A Square Matrix!\nPlease Try Again\n");

        exit(1);

    }

    return;

}

//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 display a matrix

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

{

    int i,j;

    for(i=0;i<row;i++)

    {

        for(j=0;j<col;j++)

            printf("%d\t",arr[i][j]);

        printf("\n");

    }

}

//function to map matrix to linear array

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

{

    int i,j,k=0;

    for(i=0;i<row;i++) //traversing the matrix

        for(j=i;j<col;j++)

        {

            list[k]=arr[i][j]; //mapping matrix to array

            k++;

        }

}

//function to display a 1-D array

void displist(int\*arr,int size)

{

    int i;

    for(i=0;i<size;i++)

        printf("%d ",arr[i]);

}

//function to check if a given matrix is symmetric or not

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

{

    int i,j;

    for(i=0;i<row;i++) //traversing

        for(j=0;j<col;j++)

            if(i!=j) //for non diagonal elements

            {

                if(arr[i][j]!=arr[j][i]) //for unequal transpose elements

                {

                    printf("Entered matrix is not symmetric");

                    exit(0);

                }

            }

}

int main(void)

{

    int ch,row,col,arr[20][20],\*list,size;

    printf("Enter Number Of Rows: ");

    scanf("%d",&row);

    printf("Enter Number Of Columns: ");

    scanf("%d",&col);

    validate(row,col); //input validation

    printf("\nEnter %d elements of the matrix: \n",row\*col);

    getmat(arr,row,col);

    printf("\nEntered Matrix: \n");

    dispmat(arr,row,col);

    issymatrix(arr,row,col); //checking entered matrix

    puts("\nEntered Matrix Is Symmetric");

    size=(row\*(row+1))/2; //number of elements of 1-D array

    list=(int\*)malloc(size\*sizeof(int));

    makelist(list,arr,row,col);

    printf("\nGenerated List: ");

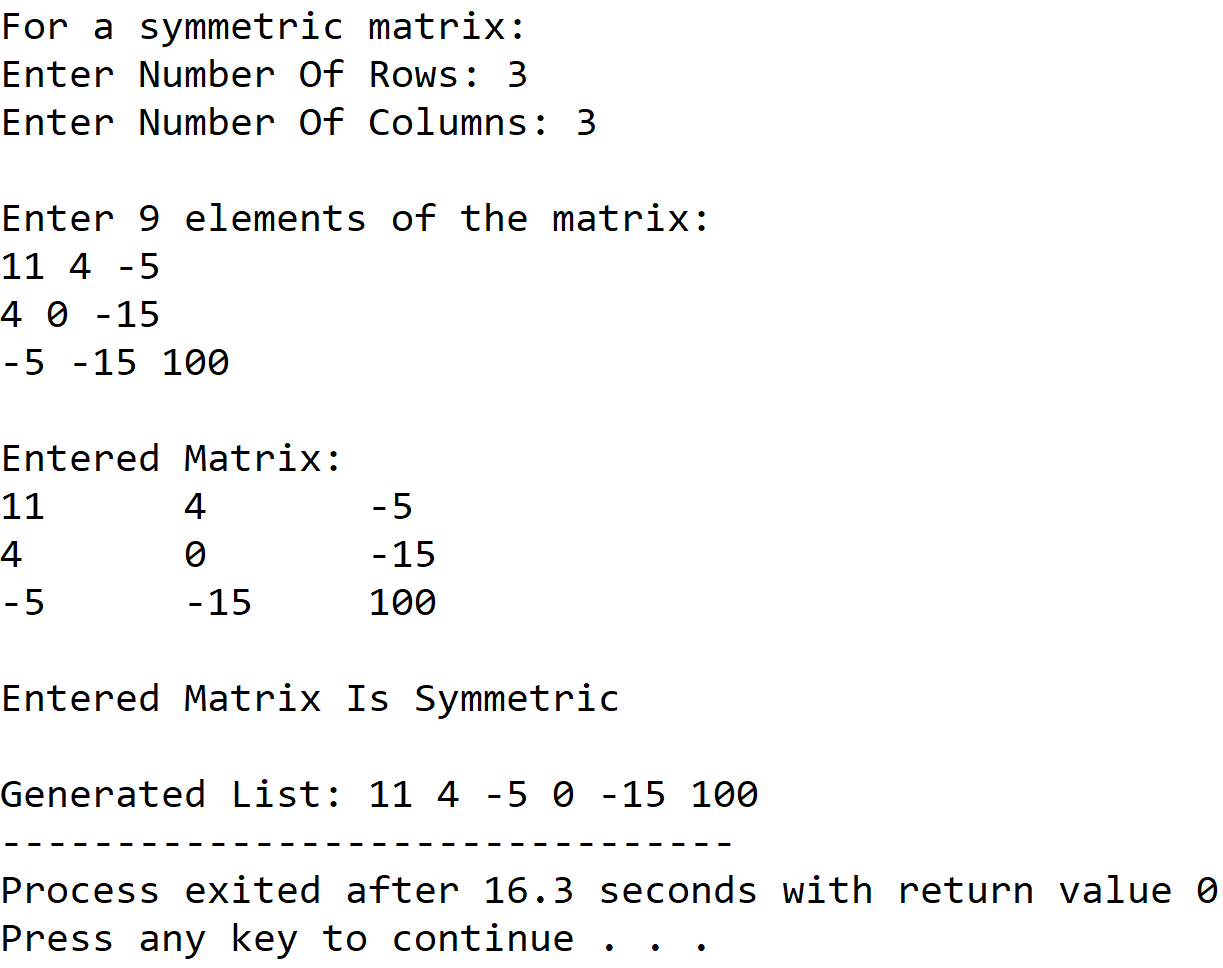
    displist(list,size);

    return 0;

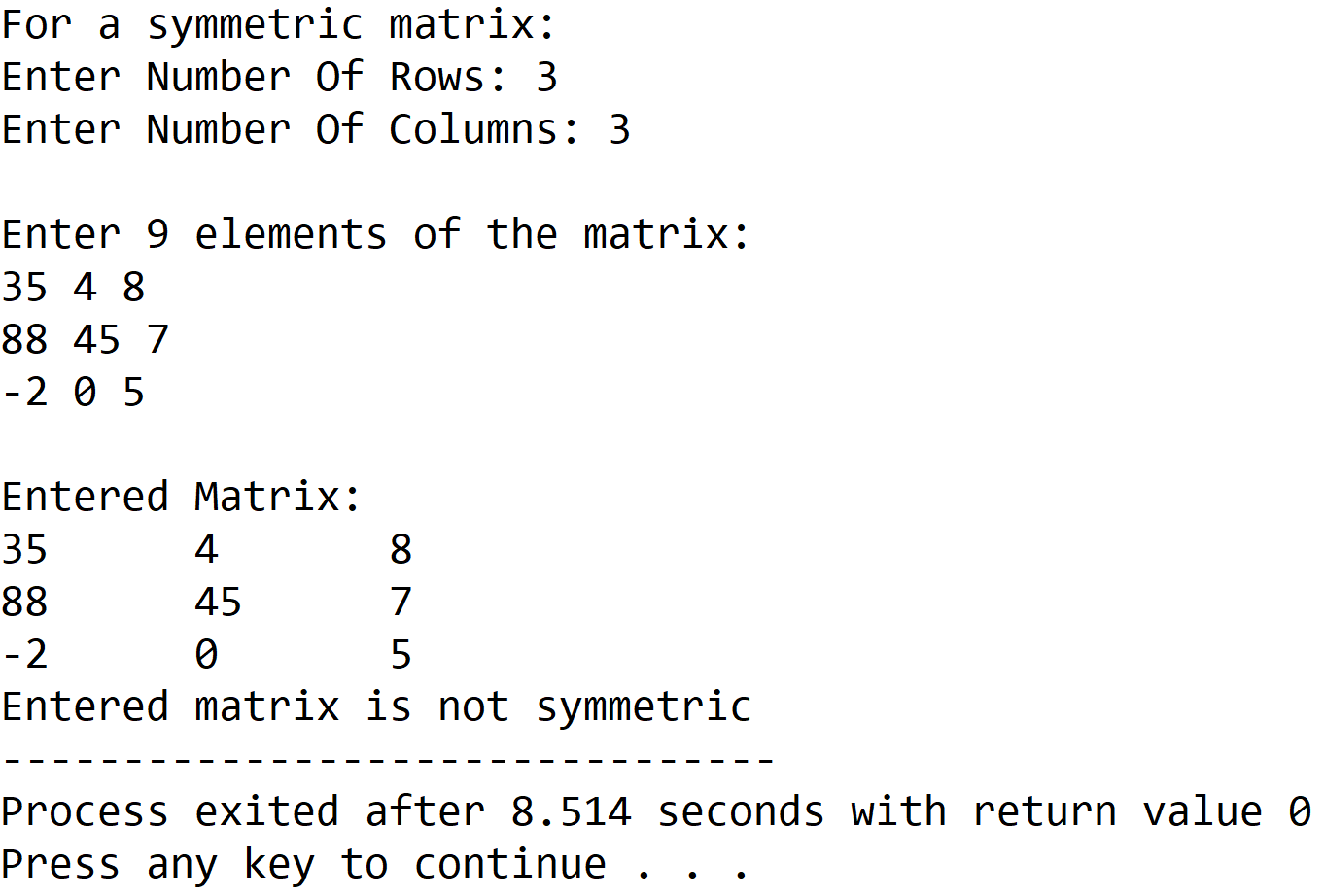
}

**4.OUTPUT**

**SET 1:** Input matrix is a symmetric matrix



**SET 2:** Input matrix is not a symmetric matrix



**5.DISCUSSIONS**

**Variable Description**

* **arr:** two dimensional array to store a matrix.
* **\*list:** pointer to an integer array to hold the symmetric matrix.
* **size:** number of elements required in list.
* **row,col:** dimensions of arr.
* **i,j,k:** loop counters.

**Limitations**

* The program uses a statically declared two dimensional matrix thus memory locations are either wasted or there is a shortage of memory.
* The symmetric matrix is stored in a 1-D array which is a static data structure which means that its size cannot be changed once it is constructed in memory.

**Uses**

* The program depicts how a particular data structure like a symmetric matrix can be stored in a much more efficient manner by holding only that data which is essential and not including repetitive, unrequired data.

**Future Scope**

* The 1-D array can be replaced with a linked list eliminating the need of contiguous memory locations and also making it easier to manipulate the size.
* The two dimensional array can be allocated dynamically using an array of pointers.