**ASSIGNMENT-10**

**1.AIM**

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

**2.ALGORITHM**

**Algorithm Is\_Diagonal\_Matrix**

**Input:** A pointer arr to the two dimesional matrix taken as input and the dimension ‘max’ .

**Output:** If the input matrix is not a diagonal matrix , an error message is displayed and the program terminates,else nothing is returned.

**Remarks:** Input matrix must be a square matrix.

**Steps:**

1. for(i=1 to max) //row traversal
2. for(j=1 to max) //column traversal
3. if(i≠j) //for non diagonal elements
4. if(arr[i][j]≠0)
5. print”Input Matrix Is Not A Diagonal Matrix”
6. Exit //terminate the program
7. **Return** //if above does not terminate,control is returned back
8. **Stop**

**3.SOURCE CODE**

**#include<stdio.h>**

**#include<stdlib.h>**

//function to take input in a matrix from the user

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 check if a matrix is a diagonal matrix

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

{

int i,j;

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

for(j=0;j<col;j++) //column traversal

{

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

if(arr[i][j]!=0) // if not equal to 0

{

printf("\nEntered Matrix Is Not A Diagonal Matrix\n");

exit(1);

}

}

printf("\nEntered Matrix Is A Diagonal Matrix");

}

//function to store the diagonal elements in a 1-D array

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

{

int i,j,k=0;

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

for(j=0;j<col;j++) //column traversal

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

{

list[k]=arr[i][j]; //assigning to 1-D array

k++;

}

}

//function to display an array

void displist(int\* list , int max)

{

int i;

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

printf("%d ",list[i]);

}

//function to reconstruct matrix from 1-D array

void recons(int\*list,int max)

{

int i,j;

for(i=0;i<max;i++) //row traversal

{

for(j=0;j<max;j++)//column traversal

{

if(i==j)//for diagonal positions

printf("%d\t",list[i]);//print array

else

printf("0\t");

}

printf("\n");

}

}

int main(void)

{

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

//input validation

while(1)

{

printf("Enter the number of rows needed: ");

scanf("%d",&row);

printf("Enter the number of columns needed: ");

scanf("%d",&col);

if(row==col)

break;

else

printf("Input Matrix Is Not A Square Matrix....Please Try Again");

}

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

getmat(arr,row,col); //taking input in matrix

printf("\nEntered Matrix: \n");

dispmat(arr,row,col);//displaying entered matrix

isdiagmat(arr,row,col);//checkin for diagonal matrix

list=(int\*)calloc(row,sizeof(int));//creating array in heap

genlist(arr,list,row,col);//setting up array

printf("\nGenerated List: ");

displist(list,row);//displaying generated array

printf("\nReconstructed Matrix:\n");

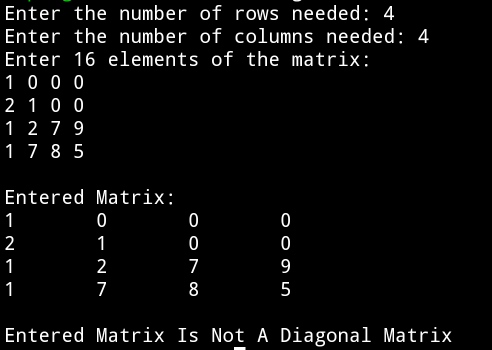
recons(list,row);//reconstructing matrix from array

return 0;

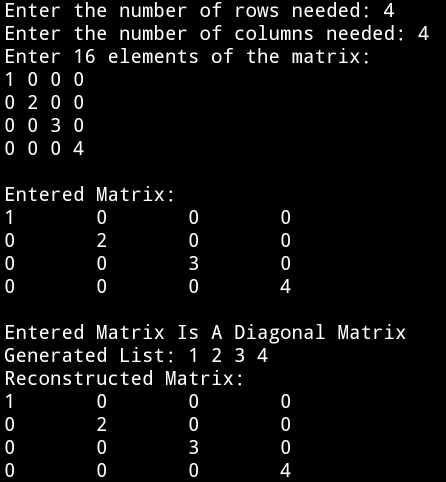
}

**4.OUTPUT**

**SET 1:** Input matrix is not a diagonal matrix



**SET 2:** Input matrix is a diagonal matrix



**5.DISCUSSIONS:**

1. **Variable Description**

**In main function**

* **arr[20][20]:** 2-D array to hold a matrix
* **\*list:** Pointer to 1-D array
* **row:** number of rows in 2-D matrix
* **col:** number of columns in 2-D matrix

**loop counters used: i,j,k**

1. **Limitations**

* The program uses arrays to hold the input matrix and 1-D array, since arrays are a static data structure,their size cannot be changed once they are constructed in the memory.

1. **Uses**

* In a diagonal matrix, all the elements except the main diagonal elements are zero , if we store only the main diagonal elements in a one dimesional array and discard all the non diagonal values since they are zero, we can reconstruct the same matrix from a much smaller 1-D array.Thus this program shows memory optimization for storing data.

1. **Future Scope**

* Instead of using arrays, linked lists can be used to store the matrix and the list , allowing more flexibility in manipulation of the size.