ACADEMY OF TECHNOLOGY

NAME : PRIYADARSHAN GHOSH

COLLEGE ROLL NO: 72

UNIVERSITY ROLL NO: 16900319072

DEPARTMENT: ECE-1(Y)

SEMESTER: 3rd

PAPER CODE : ES-CS391

➢ Laboratory *Assignment #2*

A. Write a program to store element in 2D array and calculate the address of a particular location using row major and column major representation.

Ans:

```
#include<stdio.h>
int main()
{
int l1, u1, l2, u2, ba, size, i, j, rma, cma, R, C;
 printf("\n Enter lower index 1 and upper index 1:");
 scanf("%d%d",&l1,&u1);
 printf("\n Enter lower index 2 and upper index 2:");
 scanf("%d%d",&l2,&u2);
 R = u1-l1+1;
 C = u2-l2+1;
 printf("\n Enter base address and size of memory block");
 scanf("%d%d",&ba, &size);
 printf("\n Enter location index to calculate address:");
 scanf("%d%d",&i,&j);
 i = i-l1;
 j = j-l2;
 rma = ba + (i*C+j)*size;
 cma = ba + (j*R+i)*size;
printf("In row major concept we have: %d",rma);
printf("In col major concept we have: %d",cma);
return 0;
}
```

OUTPUT =>

```
Enter lower index 1 and upper index 1:-2 -4

Enter lower index 2 and upper index 2:3 4

Enter base address and size of memory block 2500 4

Enter location index to calculate address:4 8
In row major concept we have: 2568In col major concept we have: 2504

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

B. WRITE A PROGRAM TO CHECK WHETHER A GIVEN MATRIX IS SPARSE OR NOT. IF IT IS SPARSE THEN REPRESENT IT IN 3-TUPE FORMAT.

Ans:

```
#include <stdio.h>
#include <stdlib.h>
typedef struct sparse
{
        int row;
        int col;
        int value;
}stype;
```

```
void display(stype sp[], int n)
{
	int i;
	for(i=0;i<n;i++)
	printf("\n%d\t%d\t%d",sp[i].row, sp[i].col, sp[i].value);
}
void create_sparse(stype sp[], int *p)
{
	int i,j,k=0,nor,noc,x;
	printf("\n Enter number of rows, cols & non-zero values:");
	scanf("%d%d%d",&nor,&noc,&sp[k].value);
	sp[k].row=nor;
	sp[k].col=noc;
	k++;
	printf("\n Enter elements one by one:");
	for(i=0;i<nor;i++)
	{
		for(j=0;j<noc;j++)
		{
			scanf("%d",&x);
			if(x!=0)
			{
			sp[k].row=i;
			sp[k].col=j;
			sp[k].value=x;
			k++;
			}
```

```
            }
        }
        *p=k;
}
int main()
{
        int r,c,i,j,sparse_counter=0,x,k=0,p1;
        stype sp[10];
        printf("enter the no. of rows and cols. :");
        scanf("%d %d",&r,&c);
        printf("\nEnter the elements of matrix\n");
        for(i=0;i<r;i++)
        {
                for(j=0;j<c;j++)
                {
                        scanf("%d",&x);
                        if(x!=0)
                        {
                                sparse_counter++;
                        }
                }
        }
        if(sparse_counter < ((r*c)/3) )
        {
                printf("Matrix is sparse representable\n");
                create_sparse(sp,&p1);
                display(sp,p1);
```

```
        }
        else{
                printf("Matrix is not sparse matrix");
        }
}
```

OUTPUT =>

```
enter the no. of rows and cols. :3 3

Enter the elements of matrix
1 0 0
0 0 0
0 0 2
Matrix is sparse representable

 Enter number of rows, cols & non-zero values:3 3 2

 Enter elements one by one:
1 0 0
0 0 0
0 0 2

3       3       2
0       0       1
2       2       2

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

C. WRITE A PROGRAM TO ADD TWO SPARSE MATRICES.

Ans:

```
#include<stdio.h>
#include<conio.h>
typedef struct sparse
{
int row;
int col;
int value;
}stype;

void create_sparse(stype [],int *);
void add_sparse(stype [], int, stype [],int);
void display(stype [], int);
int main()
{
        stype sp1[10],sp2[10],sp3[20];
        int p1,p2;
        printf("\n Enter data for matrix 1:\n");
        create_sparse(sp1,&p1);
        printf("\n The matrix 1=\n");
        display(sp1,p1);
        printf("\n Enter data for matrix 2:\n");
        create_sparse(sp2,&p2);
        printf("\n The matrix 2=\n");
```

```
        display(sp2,p2);
        add_sparse(sp1,p1,sp2,p2);
        printf("\n The transpose of matrix 1 is:\n");
        display(sp1,p1);
        return 0;
}
void create_sparse(stype sp[], int *p)
{
        int i,j,k=0,nor,noc,x;
        printf("\n Enter number of rows, cols & non-zero values:");
        scanf("%d%d%d",&nor,&noc,&sp[k].value);
        sp[k].row=nor;
        sp[k].col=noc;
        k++;
        printf("\n Enter elements one by one:");
        for(i=0;i<nor;i++)
        {
                for(j=0;j<noc;j++)
                {
                        scanf("%d",&x);
                        if(x!=0)
                        {
                                sp[k].row=i;
                                sp[k].col=j;
                                sp[k].value=x;
                                k++;
                        }
```

```
            }
        }
        *p=k;
}
void display(stype sp[], int n)
{
        int i;
        for(i=0;i<n;i++)
        printf("\n%d\t%d\t%d",sp[i].row, sp[i].col, sp[i].value);
}
void add_sparse( stype sp1[], int p1, stype sp2[], int p2)
{
        int i,j,k=1,flag=0;
        stype sp3[20];
        if(sp1[0].row!=sp2[0].row || sp1[0].col!=sp2[0].col)
        {
                printf("\n Addition is not possible");
                return;
        }
        sp3[0].row=sp1[0].row;
        sp3[0].col=sp1[0].col;
        for(i=1;i<p1;i++)
        {
                flag=0;
                for(j=1;j<p2;j++)
                {
                        if(sp1[i].row==sp2[j].row && sp1[i].col==sp2[j].col)
```

```
            {
                sp3[k].row=sp1[i].row;
                sp3[k].col=sp1[i].col;
                sp3[k].value=sp1[i].value+sp2[j].value;
                k++;
                flag=1;
                break;
            }
        }
        if(flag==0)
        {
            sp3[k].row=sp1[i].row;
            sp3[k].col=sp1[i].col;
            sp3[k].value=sp1[i].value;
            k++;
        }
    }
    for(j=1;j<p2;j++)
    {
        flag=0;
        for(i=1;i<p1;i++)
        {
            if(sp1[i].row==sp2[j].row && sp1[i].col==sp2[j].col)
            {
                flag=1;
                break;
            }
```

```
            }
            if(flag==0)
            {
                  sp3[k].row=sp2[j].row;
                  sp3[k].col=sp2[j].col;
                  sp3[k].value=sp2[j].value;
                  k++;
            }
      }
      sp3[0].value=k-1;
      printf("\n The resultant matrix is=\n");
      display(sp3,k);
}
```

OUTPUT =>

```
 Enter data for matrix 1:

 Enter number of rows, cols & non-zero values:3 3 2

 Enter elements one by one:0 1 0
0 0 2
0 0 0

 The matrix 1=

3      3      2
0      1      1
1      2      2
```

Enter data for matrix 2:

Enter number of rows, cols & non-zero values:3 3 2

Enter elements one by one:0 1 0

0 0 2

0 0 0

The matrix 2=

3 3 2

0 1 1

1 2 2

The resultant matrix is=

3 3 2

0 1 2

1 2 4

The transpose of matrix 1 is:

3 3 2

0 1 1

1 2 2

...Program finished with exit code 0

Press ENTER to exit console.