ASSIGNMENT NO:5 **DATE:27/02/2015**

PROGRAM TITLE:Add two Sparse matrices.

THEORY:A matrix which has atleast 75% of its elements as zero is known as a sparse matrix. The matrix is stored in the form of a structure array so as to minimise memory use.

PROGRAM ALGORITHM:

```
Algo_sparseadd(a,b,d,r,c)
{
      initialise metadata of d;
      x=1;
      y=1;
      z=1;
      while((x!=number of elements in a)&&(y!=number of elements of b))
      {
            if(index of a[x]<index of b[y])
            {
                  store data in a[x] in d[z];
                  x++;
                  z++;
            }
            else if(index of a[x]>index of b[y])
            {
                  store data in b[y] in d[z];
                  y++;
                  z++;
            }
            else
            {
                  add value in a[x] and b[y] and store in d[z] if result
is not zero;
                  x++;
                  y++;
                  z++;
            }
      }
}
```

PROGRAM CODE:

```
/*C Program to add two Sparse Matrices*/
#include <stdio.h>
#include <ctype.h>
#include <stdlib.h>
struct Sparse
{
      int i,j,val;
};
int checkrc(int a,int b);//checks if valid row and column number
int checksparse(int a,int b);//checks if number of elements acceptable
for sparse matrix
int checkzero(int x);//checks for zero, returns 1 if zero else returns 0
```

```
int check(int a,int b);//compares to numbers, returns -1 if a<b,0 if
a==b, 1 if a>b
int structip(struct Sparse * a,int r,int c,int val);// inputs matrix
int structstore(struct Sparse * d,int z,int i,int j,int val);//stores in
the structure
int structadd(struct Sparse * a,struct Sparse * b,struct Sparse * d,int
r,int c);//adds the content of two structures
int structdisp(struct Sparse * a);//displays the structure array
int main()
{
      int r,c,x,y;
      printf("Enter size ::");
      scanf("%d %d",&r,&c);
      checkrc(r,c);
      y=.25*r*c;
      printf("Enter number of non-zero elements in 1st Matrix::");
      scanf("%d",&x);
      checksparse(x,y);
      struct Sparse a[x+1];
      structip(a,r,c,x);
      printf("Enter number of non-zero elements in 2nd Matrix::");
      scanf("%d",&x);
      checksparse(x,y);
      struct Sparse b[x+1],d[x+a[0].val];
      structip(b,r,c,x);
      structadd(a,b,d,r,c);
      printf("The First Matrix is::\n");
      structdisp(a);
      printf("The Second Matrix is::\n");
      structdisp(b);
      printf("The Final Matrix is::\n");
      structdisp(d);
      return 0;
}
int checkrc(int a,int b)
{
      if(a<=0||b<=0)
      {
            printf("Row or Column Number cant be accepted.\nProgram
Terminated.\n");
            exit (1);
      }
      else
            return 1;
}
int checksparse(int a,int b)
{
      if(a>b)
      {
            printf("No of non-zero elements is not suitable for sparse
matrix.\nProgram Terminated.\n");
            exit (2);
      }
      else
            return 2;
```

```
}
int checkzero(int x)
{
      if(x==0)
            return 1;
      return 0;
}
int check(int a,int b)
{
      if(a<b)
            return -1;
      else if(a==b)
            return 0;
      else
            return 1;
}
int structip(struct Sparse * a,int r,int c,int val)
{
      int i,x;
      a[0].i=r;
      a[0].j=c;
      a[0].val=val;
      printf("Enter the elements of the matrix along with position in
ascending
order::\n\ti\tj\tval\n-----------------------------------------------\n");
      for(i=1;i<val+1;i++)
      {
            scanf("%d %d %d",&a[i].i,&a[i].j,&x);
            if(checkzero(x))
            {
                  i--;
                  continue;
            }
            a[i].val=x;
            checkrc(a[i].i,a[i].j);
            if((a[i].i>a[0].i)||(a[i].j>a[0].j))
            {
                  printf("Entered value exceeds boundary.\nProgram
Terminated.\n");
                  exit(4);
            }
      }
      return 3;
}
int structstore(struct Sparse * d,int z,int i,int j,int val)
{
      d[z].i=i;
      d[z].j=j;
      d[z].val=val;
      d[0].val++;
      return 0;
}
int structadd(struct Sparse * a,struct Sparse * b,struct Sparse * d,int
r,int c)
{
```

```
int x,y,z;
d[0].i=r;
d[0].j=c;
d[0].val=0;
for(x=1,y=1,z=1;(x<=a[0].val)&&(y<=b[0].val);)
{
      if(check(a[x].i,b[y].i)==-1)
      {
            structstore(d,z,a[x].i,a[x].j,a[x].val);
            x++;
            z++;
      }
      else if(check(a[x].i,b[y].i)==1)
      {
            structstore(d,z,b[y].i,b[y].j,b[y].val);
            y++;
            z++;
      }
      else
      {
                  if(check(a[x].j,b[y].j)==-1)
            {
                  structstore(d,z,a[x].i,a[x].j,a[x].val);
                  x++;
                  z++;
            }
            else if(check(a[x].j,b[y].j)==1)
            {
                  structstore(d,z,b[y].i,b[y].j,b[y].val);
                  y++;
                  z++;
            }
            else
            {
                  if(!checkzero(a[x].val+b[y].val))
                  {

structstore(d,z,b[y].i,b[y].j,a[x].val+b[y].val);
                        z++;
                  }
                  x++;
                  y++;
            }
      }
}
if(x>a[0].val)
{
      for(;y<=b[0].val;)
      {
            structstore(d,z,b[y].i,b[y].j,b[y].val);
            y++;
            z++;
      }
}
else if(y>b[0].val)
```

```
	{
		for(;x<=a[0].val;)
		{
			structstore(d,z,a[x].i,a[x].j,a[x].val);
			x++;
			z++;
		}
	}
	return 4;
}
int structdisp(struct Sparse * a)
{
	int i;

printf("\n\ti\tj\tval\n-----------------------------------------------\n")
;
	for(i=0;i<=a[0].val;i++)
	{
		printf("\n\t%d\t%d\t%d\n",a[i].i,a[i].j,a[i].val);
	}
	return 0;
}
```

OUTPUT:

```
Enter size ::4 4
Enter number of non-zero elements in 1st Matrix::4
Enter the elements of the matrix along with position in ascending order::
	i	j	val
-----------------------------------------------
	1	1	2
	1	2	3
	1	3	-1
	3	4	2
Enter number of non-zero elements in 2nd Matrix::4
Enter the elements of the matrix along with position in ascending order::
	i	j	val
-----------------------------------------------
	1	2	5
	1	3	1
	2	1	3
	3	1	8
The First Matrix is::

	i	j	val
-----------------------------------------------

	4	4	4

	1	1	2

	1	2	3

	1	3	-1
```

```
      3     4     2
The Second Matrix is::

      i     j     val
-----------------------------------------

      4     4     4

      1     2     5

      1     3     1

      2     1     3

      3     1     8
The Final Matrix is::

      i     j     val
-----------------------------------------

      4     4     5

      1     1     2

      1     2     8

      2     1     3

      3     1     8

      3     4     2
```

DISCUSSION:

The complexity of the Program is O(n). The program stores only if the value entered is not zero. Also, the user has to take care to enter the indexes in the ascending order.