20/09/2020

**Experiment No. 5**

**SPARSE MATRIX**

**AIM:**

Write a program to enter two matrices in normal form. Write a function to convert two matrices to tuple form and display it. Also find the transpose of the two matrices represented in tuple form and display it. Find the sum of the two matrices in tuple form and display the sum in tuple form.

**DATA STRUCTURES USED:**

Arrays

**ALGORITHM:**

START

1. Accept the two matrix in normal form and R is the Resultant Matrix

2. Traverse throught the matrix such that k starts from 1

3. Find non zero values

4. Store its row in R[i][0] and column in R[i][1] and value in R[i][2]

5. Store R[0][0] = num of rows

6. Store R[0][1] = num of columns

7. Store R[0][0] = k-1 (Number of non-zero values)

8. Print the resultant Tuple Representation

9. Function Transpose(int sp[][3])

10. Check whether sp[0][2] is 0: then return "No elements"

11. Copy sp[0][0] into spt[0][0]

12. Copy sp[0][1] into spt[0][1]

13. Copy sp[0][2] into spt[0][2]

14. k = 1

15. for i=0 till number of columns

16. for j=1 till the number of non zero values

17. if i == a[j][1], insert the entire row into Resultant Array

18. k++

19. End if

20. End for

21. End for

22. Print Resultant Array

23. Function Addition(int sp1[][3],int sp2[][3])

24. If matrices doesn't match in size (i.e, rows and columns are not equal), print "Invalid operation"

25. Else

26. while i <= sp1[0][2] or j <= sp2[0][2] do

27. If sp1[i][0] < sp2[j][0]

28. Copy the data of ith row of sp1 to Resultant, i++, k++

29. Else if sp1[i][0] > sp2[j][0]

30. Copy the data of jth row of sp2 to Resultant, j++, k++

31. Else

32. If sp1[i][1] < sp2[j][1]

33. Copy the data of ith row of sp1 to Resultant, i++, k++

34. Else if sp1[i][1] > sp2[j][1]

35. Copy the data of jth row of sp2 to Resultant, j++, k++

36. Else

37. Add the values and insert to Resultant along with the row and column data, i++, j++, k++

38. End if

39. End if

40. End while

41. End if

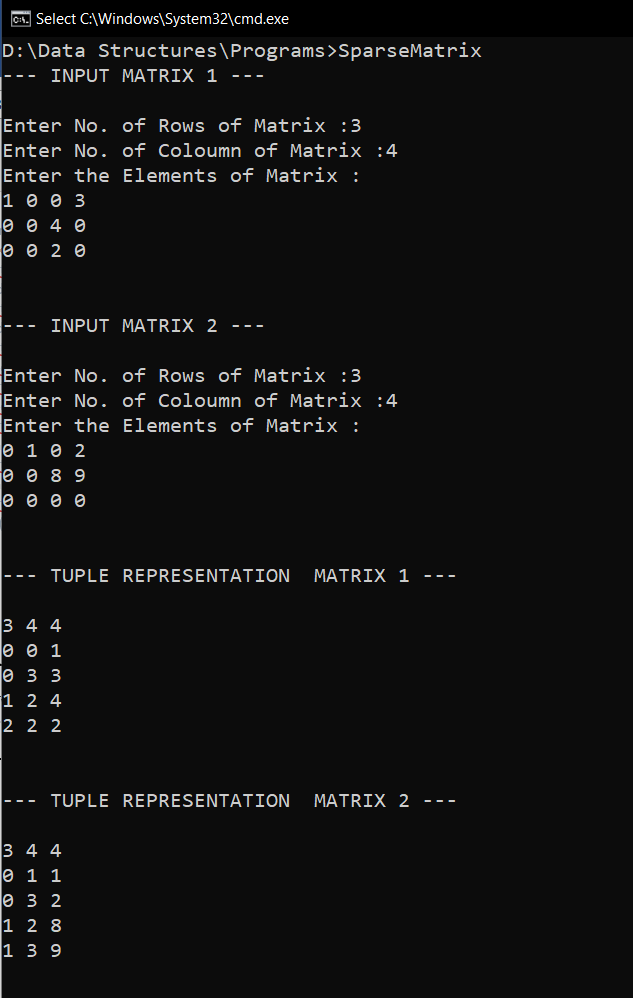
42. Print the Resultant Tuple Representation

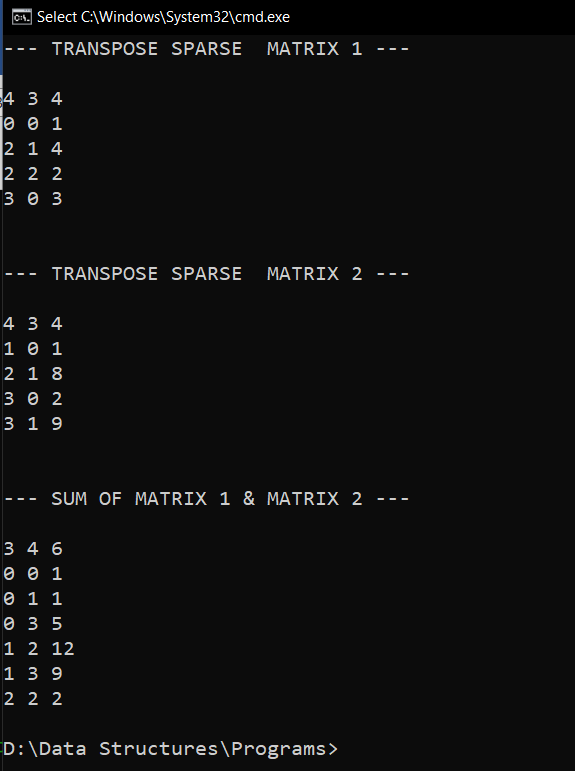
STOP

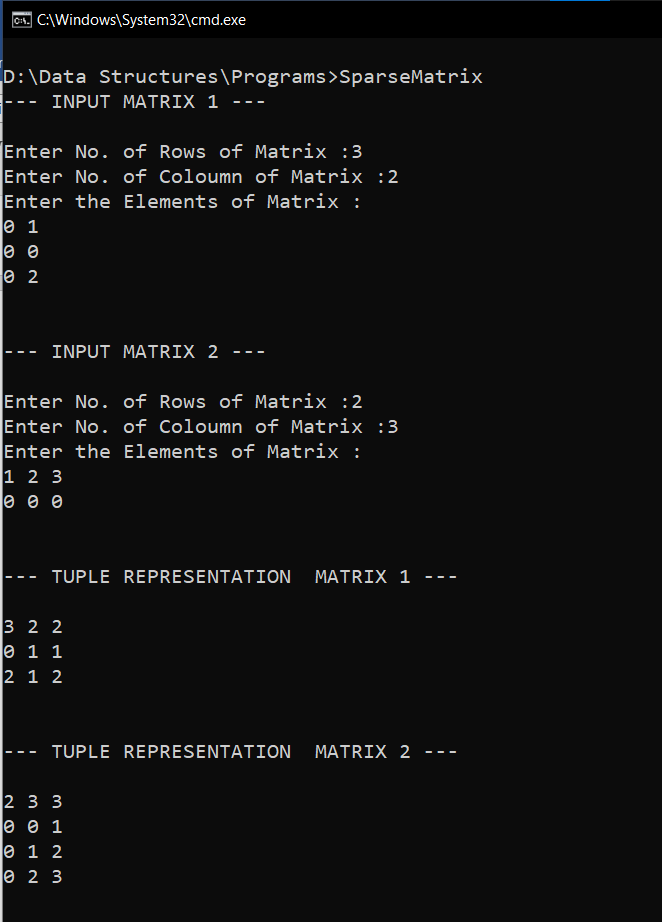
**PROGRAM:**

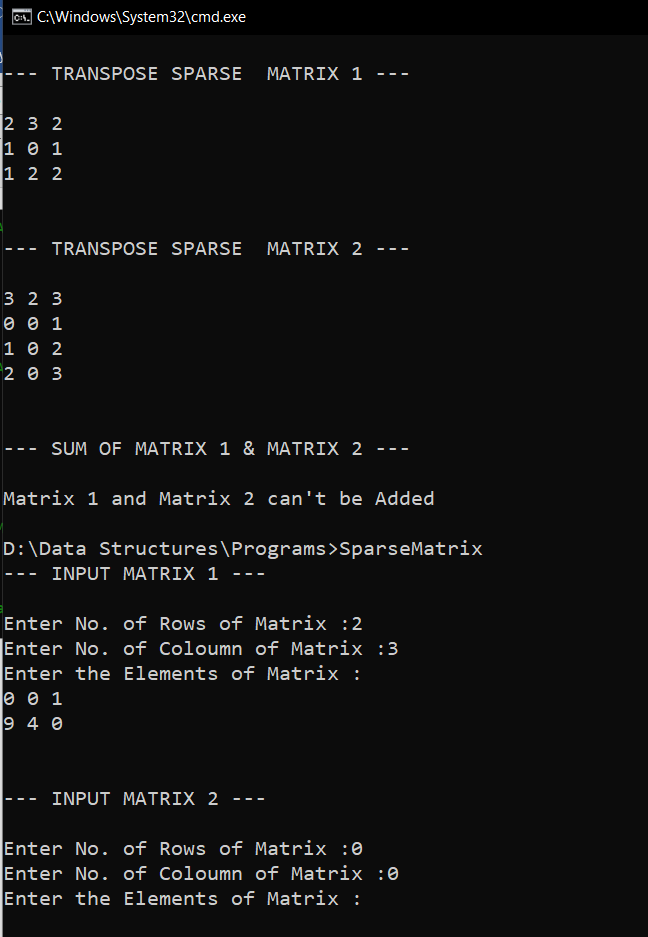
#include <stdio.h>  
struct sparse{  
 int row, col;  
 int arr[10][10];  
 int sarr[50][3];  
 int tarr[50][3];  
};  
  
  
void read(struct sparse \*sp){  
 printf("Enter No. of Rows of Matrix :");  
 scanf("%d", &sp->row);  
 printf("Enter No. of Coloumn of Matrix :");  
 scanf("%d",&sp->col);  
 printf("Enter the Elements of Matrix :\n");  
 for(int i=0;i<sp->row;i++){  
 for(int j=0;j<sp->col;j++){  
 scanf("%d", &sp->arr[i][j]);  
 }  
 }  
}  
  
void tupleRepresentation(struct sparse \*sp){  
 int k=0;  
 sp->sarr[0][0] = sp->row;  
 sp->sarr[0][1] = sp->col;  
 for(int i=0;i<sp->row;i++){  
 for(int j=0;j<sp->col;j++){  
 if(sp->arr[i][j] != 0){  
 k++;  
 sp->sarr[k][0] = i;  
 sp->sarr[k][1] = j;  
 sp->sarr[k][2] = sp->arr[i][j];  
 }  
 }  
 }  
 sp->sarr[0][2] = k;  
 for(int i=0;i<=sp->sarr[0][2];i++) {  
 printf("%d ", sp->sarr[i][0]);  
 printf("%d ", sp->sarr[i][1]);  
 printf("%d \n", sp->sarr[i][2]);  
 }  
}  
  
  
  
void transpose(struct sparse \*sp){  
  
 if(sp->sarr[0][2] == 0){  
 printf("Matrix Cannot be Transposed\n");  
 }  
 else{  
 sp->tarr[0][0] = sp->sarr[0][1];  
 sp->tarr[0][1] = sp->sarr[0][0];  
 sp->tarr[0][2] = sp->sarr[0][2];  
 int k=1;  
 for(int i=0;i<sp->sarr[0][1];i++){  
 for(int j=1;j<=sp->sarr[0][2];j++){  
 if(i == sp->sarr[j][1]){  
 sp->tarr[k][0] = sp->sarr[j][1];  
 sp->tarr[k][1] = sp->sarr[j][0];  
 sp->tarr[k][2] = sp->sarr[j][2];  
 k++;  
 }  
 }  
 }  
 for(int i=0;i<=sp->tarr[0][2];i++) {  
 printf("%d ", sp->tarr[i][0]);  
 printf("%d ", sp->tarr[i][1]);  
 printf("%d \n", sp->tarr[i][2]);  
 }  
 }  
  
}  
  
  
void add(struct sparse \*sp1, struct sparse \*sp2, struct sparse \*sp3){  
 int i=1, j=1, k=1;  
 if(sp1->sarr[0][0]!=sp2->sarr[0][0]||sp1->sarr[0][1]!= sp2->sarr[0][1]){  
 printf("Matrix 1 and Matrix 2 can't be Added\n");  
 }  
else{  
 while(i<=sp1->sarr[0][2]||j<=sp2->sarr[0][2]){  
 if(sp1->sarr[i][0]==sp2->sarr[j][0]){  
 if(sp1->sarr[i][1]==sp2->sarr[j][1]){  
 sp3->sarr[k][2]=sp1->sarr[i][2]+sp2->sarr[j][2];  
 sp3->sarr[k][1] = sp1->sarr[i][1];  
 sp3->sarr[k][0] = sp1->sarr[i][0];  
 k++, i++, j++;  
 }  
 else if(sp1->sarr[i][1] < sp2->sarr[j][1]){  
 sp3->sarr[k][0] = sp1->sarr[i][0];  
 sp3->sarr[k][1] = sp1->sarr[i][1];  
 sp3->sarr[k][2] = sp1->sarr[i][2];  
 k++, i++;  
 }  
 else{  
 sp3->sarr[k][0] = sp2->sarr[j][0];  
 sp3->sarr[k][1] = sp2->sarr[j][1];  
 sp3->sarr[k][2] = sp2->sarr[j][2];  
 k++, j++;  
 }  
 }  
 else if(sp1->sarr[i][0] < sp2->sarr[j][0])  
 {  
 sp3->sarr[k][0] = sp1->sarr[i][0];  
 sp3->sarr[k][1] = sp1->sarr[i][1];  
 sp3->sarr[k][2] = sp1->sarr[i][2];  
 k++, i++;  
 }  
 else{  
 sp3->sarr[k][0] = sp2->sarr[j][0];  
 sp3->sarr[k][1] = sp2->sarr[j][1];  
 sp3->sarr[k][2] = sp2->sarr[j][2];  
 k++, j++;  
 }  
 }  
  
 sp3->sarr[0][0] = sp1->sarr[0][0];  
 sp3->sarr[0][1] = sp1->sarr[0][1];  
 sp3->sarr[0][2] = k-1;  
 for(int i=0;i<=sp3->sarr[0][2];i++) {  
 printf("%d ", sp3->sarr[i][0]);  
 printf("%d ", sp3->sarr[i][1]);  
 printf("%d \n", sp3->sarr[i][2]);  
 }  
 }  
}  
  
  
void main(){  
 struct sparse sp1, sp2,sp3;  
 printf("--- INPUT MATRIX 1 ---\n\n");  
 read(&sp1);  
 printf("\n\n--- INPUT MATRIX 2 ---\n\n");  
 read(&sp2);  
 printf("\n\n--- TUPLE REPRESENTATION MATRIX 1 ---\n\n");  
 tupleRepresentation(&sp1);  
 printf("\n\n--- TUPLE REPRESENTATION MATRIX 2 ---\n\n");  
 tupleRepresentation(&sp2);  
 printf("\n\n--- TRANSPOSE SPARSE MATRIX 1 ---\n\n");  
 transpose(&sp1);  
 printf("\n\n--- TRANSPOSE SPARSE MATRIX 2 ---\n\n");  
 transpose(&sp2);  
 printf("\n\n--- SUM OF MATRIX 1 & MATRIX 2 ---\n\n" );  
 add(&sp1, &sp2, &sp3);  
}

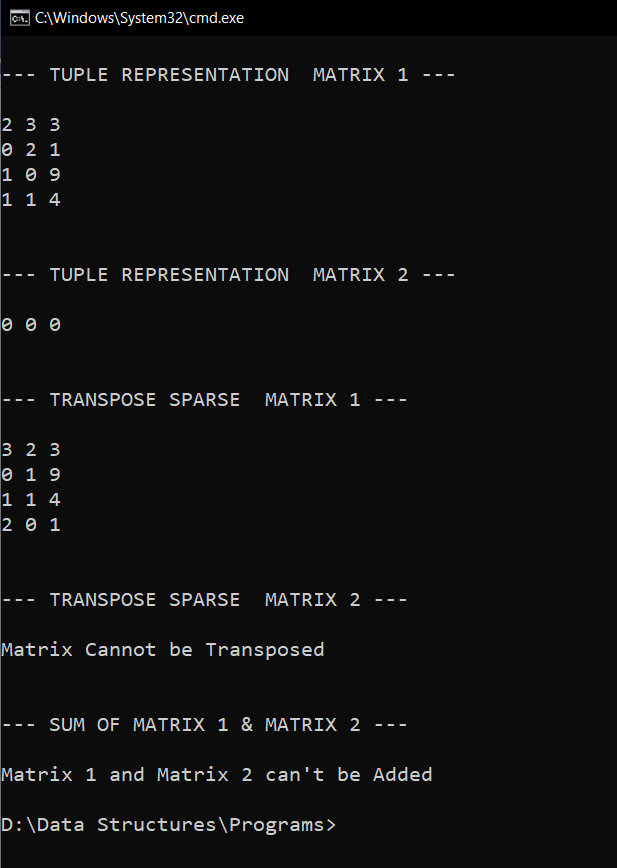
**OUTPUT:**











**RESULT:**

Two sparse matrices entered in normal form are converted to their tuple forms. The

tuple representations of their sum and each of their transposes are also found out.