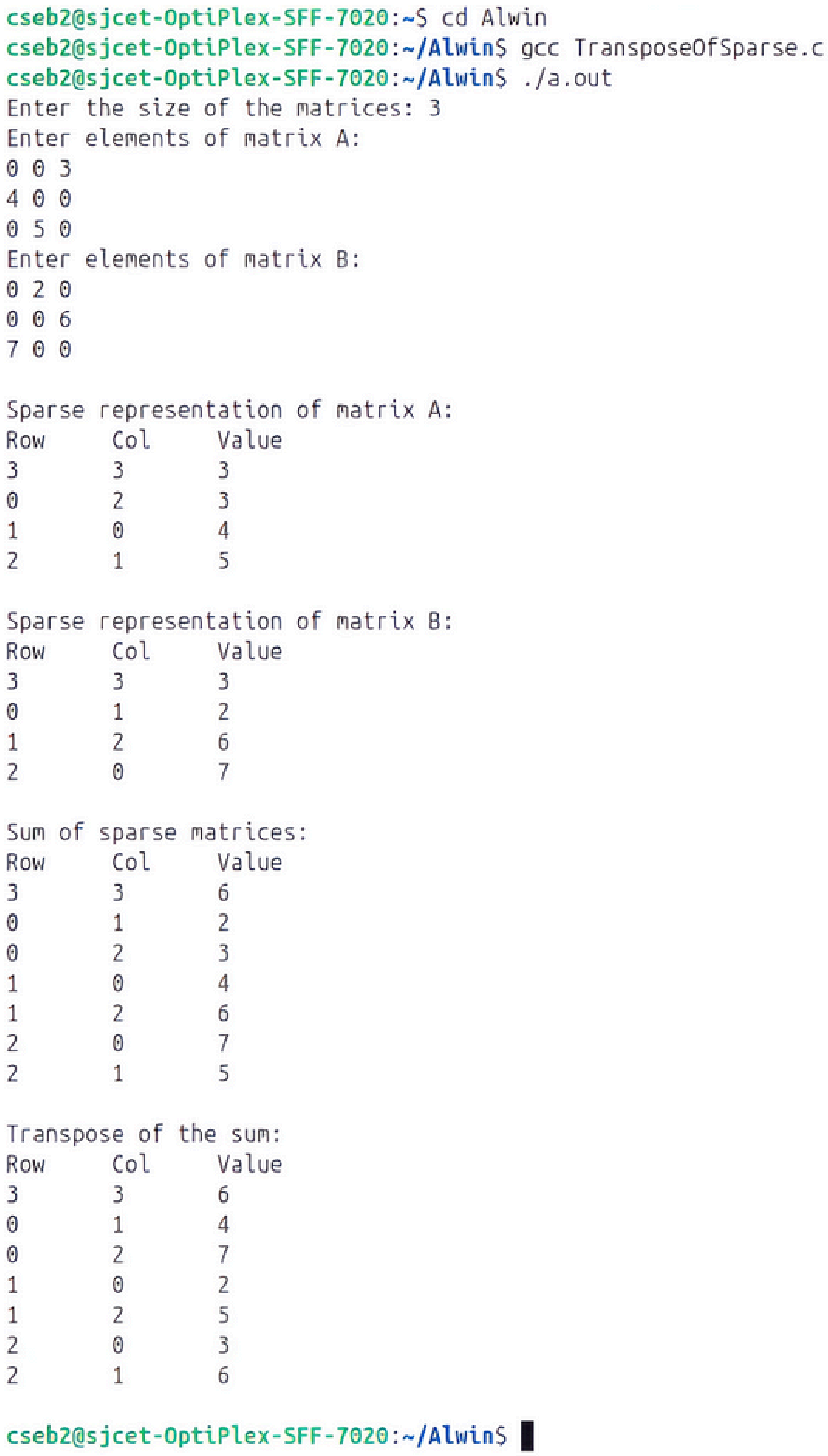
EXPERIMENT-2

Program :

```
#include <stdio.h>

int n;

int toCompact(int matrix[][n],int compactMatrix[][3],int *size){

    *size=0;

    for(int i=0;i<n;i++){

        for(int j=0;j<n;j++){

            if(matrix[i][j]!=0){

                (*size)++;

            }

        }

    }

    compactMatrix[0][0]=n;

    compactMatrix[0][1]=n;

    compactMatrix[0][2]=*size;

    int k=1;

    for(int i=0;i<n;i++){

        for(int j=0;j<n;j++){

            if(matrix[i][j]!=0){

                compactMatrix[k][0]=i;

                compactMatrix[k][1]=j;

                compactMatrix[k][2]=matrix[i][j];

                k++;

            }

        }
    }
}
int sumofcompact(int compactA[][3],int compactB[][3],int result[][3],int sizeA,int sizeB){
    int i=1,j=1,k=1;
    result[0][0]=compactA[0][0];
    result[0][1]=compactA[0][1];


    while (i<=sizeA && j<=sizeB)
    {
        if(compactA[i][0]<compactB[j][0] || (compactA[i][0]==compactB[j][0] && compactA[i][1]<compactB[j][1])){
            result[k][0]=compactA[i][0];
            result[k][1]=compactA[i][1];
            result[k][2]=compactA[i][2];
            i++;
        }
        else if(compactA[i][0]>compactB[j][0] || (compactA[i][0]==compactB[j][0] && compactA[i][1]>compactB[j][1])){
            result[k][0]=compactB[j][0];
            result[k][1]=compactB[j][1];
            result[k][2]=compactB[j][2];
            j++;
        }
        else{
            result[k][0]=compactB[j][0];
            result[k][1]=compactB[j][1];
            result[k][2]=compactA[i][2]+compactB[j][2];
            i++,j++;
        }
        k++;
    }
    while (i<=sizeA)
    {
        result[k][0]=compactA[i][0];
        result[k][1]=compactA[i][1];
        result[k][2]=compactA[i][2];
        k++,i++;
    }
    while (j<=sizeB)
    {
        result[k][0]=compactB[j][0];
        result[k][1]=compactB[j][1];
        result[k][2]=compactB[j][2];
        k++,j++;
    }
    result[0][2]=k-1;
    return k-1;
}
void transpose(int matrix[][3],int trans[][3],int size){
    trans[0][0]=matrix[0][1];
    trans[0][1]=matrix[0][0];
    trans[0][2]=matrix[0][2];
    int i,j,k=1;
    for(i=0; i<matrix[0][1];i++){
        for(j=1;j<=size;j++){
            if(matrix[j][1]==i){
                trans[k][0]=matrix[j][1];
                trans[k][1]=matrix[j][0];
                trans[k][2]=matrix[j][2];
                k++;
            }
        }
    }
}
void display(int matrix[][3],int size){
    for(int i=0;i<=size;i++){
        for(int j=0;j<3;j++){
            printf("%d\t",matrix[i][j]);
        }
        printf("\n");
    }
    printf("\n");
}
void readmatrix(int m[][n],int n){
    for(int i=0;i<n;i++){
            for(int j=0;j<n;j++){
                scanf("%d",&m[i][j]);
            }
        }
}
int main(){
```

```c
    int sizeA,sizeB;
    printf("Enter the size of two matrices: ");
    scanf("%d",&n);
    int matrixA[n][n];
    int matrixB[n][n];
    printf("\nEnter the elements of matrix 1: \n");
    readmatrix(matrixA,n);
    printf("\nEnter the elements of matrix 2: \n");
    readmatrix(matrixB,n);
    int compactA[n+1][3];
    int compactB[n+1][3];
    int result[n+1][3];
    int transposeMatrix[26][3];

    toCompact(matrixA,compactA,&sizeA);
    toCompact(matrixB,compactB,&sizeB);
    printf("Compact matrix of A: \n");
    display(compactA,sizeA);
    printf("Compact matrix of B: \n");
    display(compactB,sizeB);
    int sizeOfResult = sumofcompact(compactA,compactB,result,sizeA,sizeB);
    printf("Sum of compact matrices: \n");
    display(result,sizeOfResult);
    transpose(result,transposeMatrix,sizeOfResult);
    printf("Transpose of resultant matrix: \n");
    display(transposeMatrix,sizeOfResult);
}
```

Output :



```
cseb2@sjcet-OptiPlex-SFF-7020:~$ cd Alwin
cseb2@sjcet-OptiPlex-SFF-7020:~/Alwin$ gcc TransposeOfSparse.c
cseb2@sjcet-OptiPlex-SFF-7020:~/Alwin$ ./a.out
Enter the size of the matrices: 3
Enter elements of matrix A:
0 0 3
4 0 0
0 5 0
Enter elements of matrix B:
0 2 0
0 0 6
7 0 0

Sparse representation of matrix A:
Row     Col     Value
3       3       3
0       2       3
1       0       4
2       1       5

Sparse representation of matrix B:
Row     Col     Value
3       3       3
0       1       2
1       2       6
2       0       7

Sum of sparse matrices:
Row     Col     Value
3       3       6
0       1       2
0       2       3
1       0       4
1       2       6
2       0       7
2       1       5

Transpose of the sum:
Row     Col     Value
3       3       6
0       1       4
0       2       7
1       0       2
1       2       5
2       0       3
2       1       6

cseb2@sjcet-OptiPlex-SFF-7020:~/Alwin$
```