Course Name : Introduction to Computer Science

Course Group : Group 1

Instructor Name : Dr. Öğr. Üyesi Mehmet Amaç Güvensan

Assignment Number : Assignment 2

Student Id : 16011706

Student Name and Surname : Duygu Erduran

Q1

a. Question :

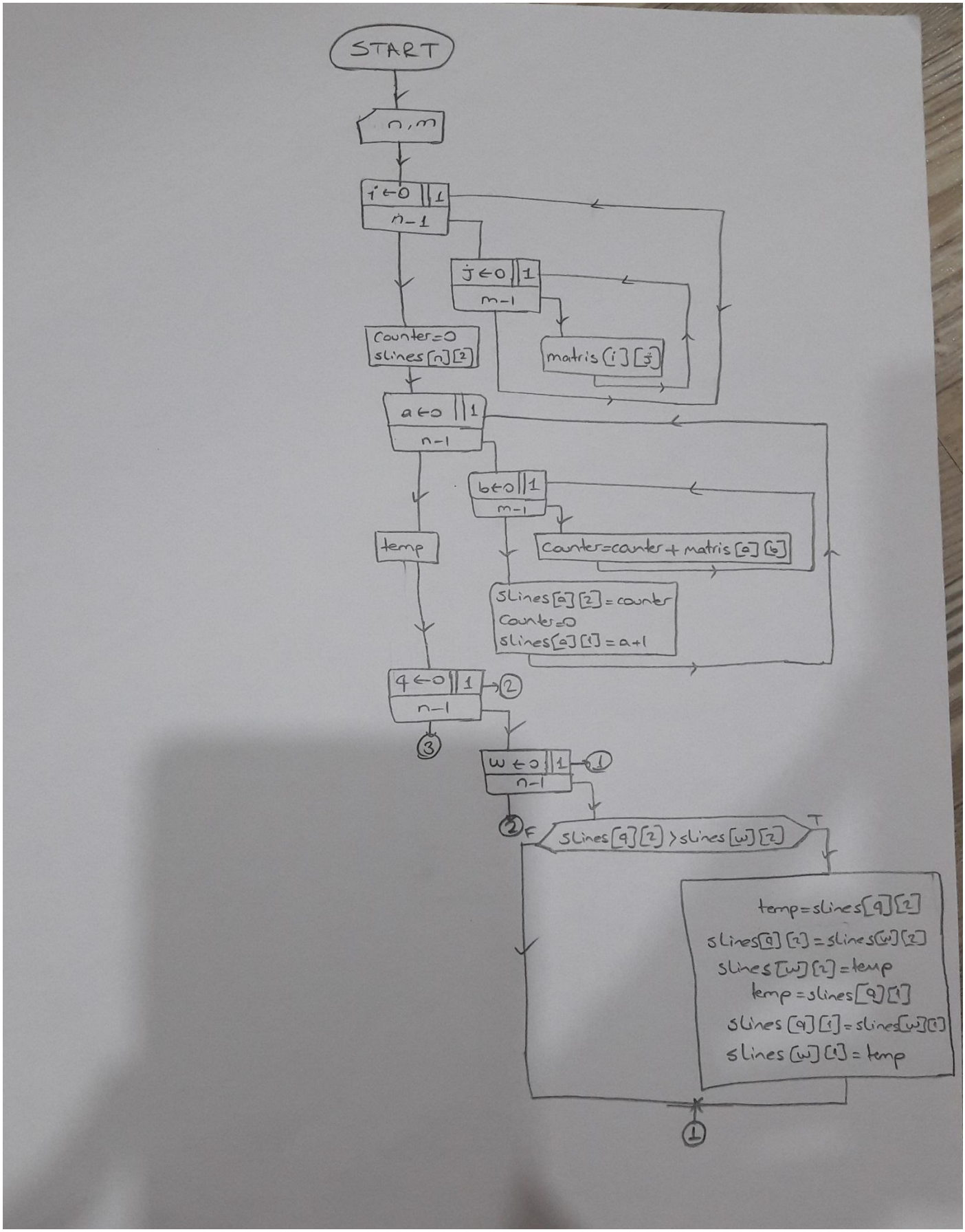
There are several bus lines and bus stops in a city. Suppose that you are given the number of buses (N), the number of bus stops (M) in whole city, and a matrix (NxM) consisting of 1s and 0s. 1s represent the stops of a bus line. Design an algorithm

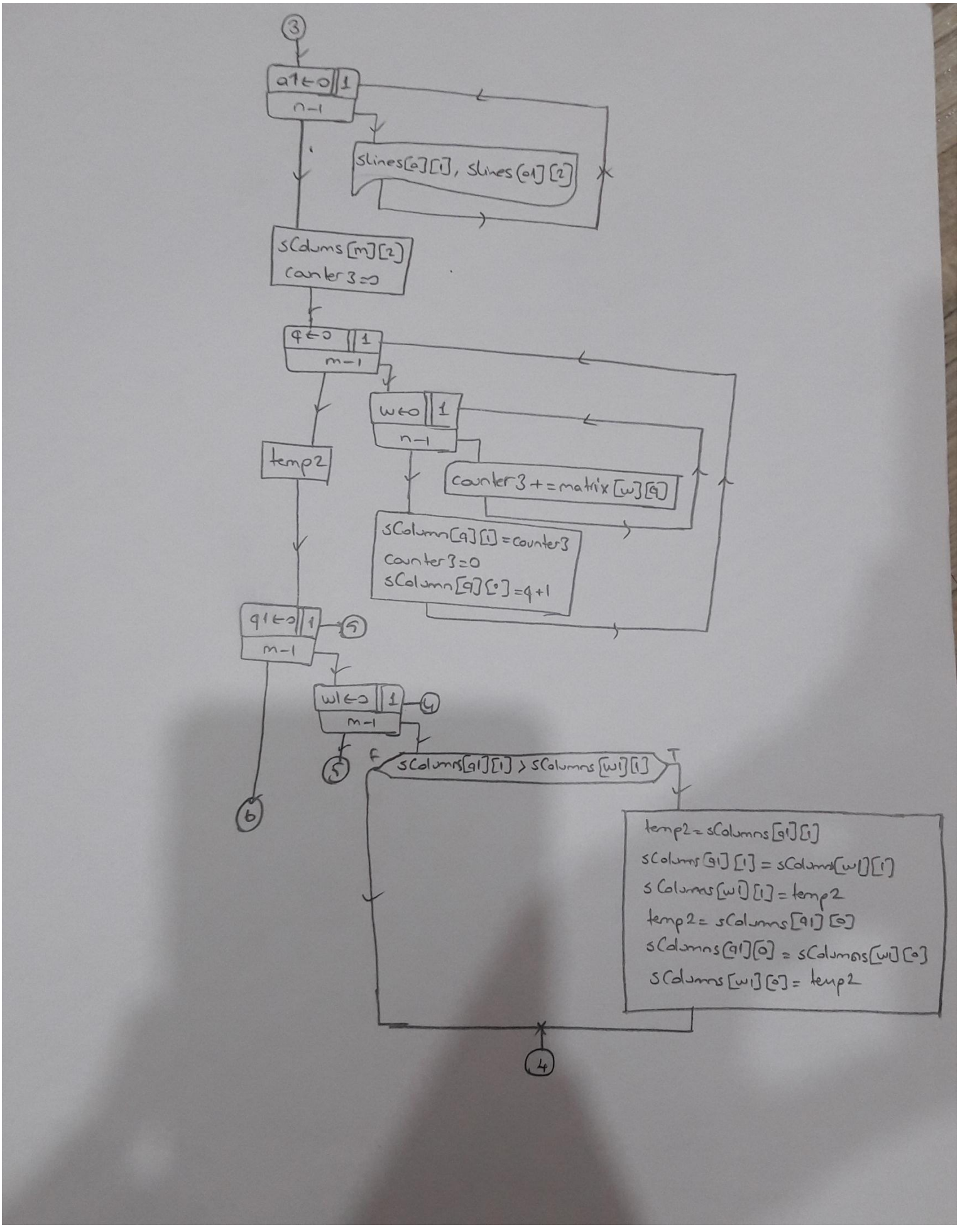
-which sorts and list the bus lines based on the number of their bus stops in descending way.

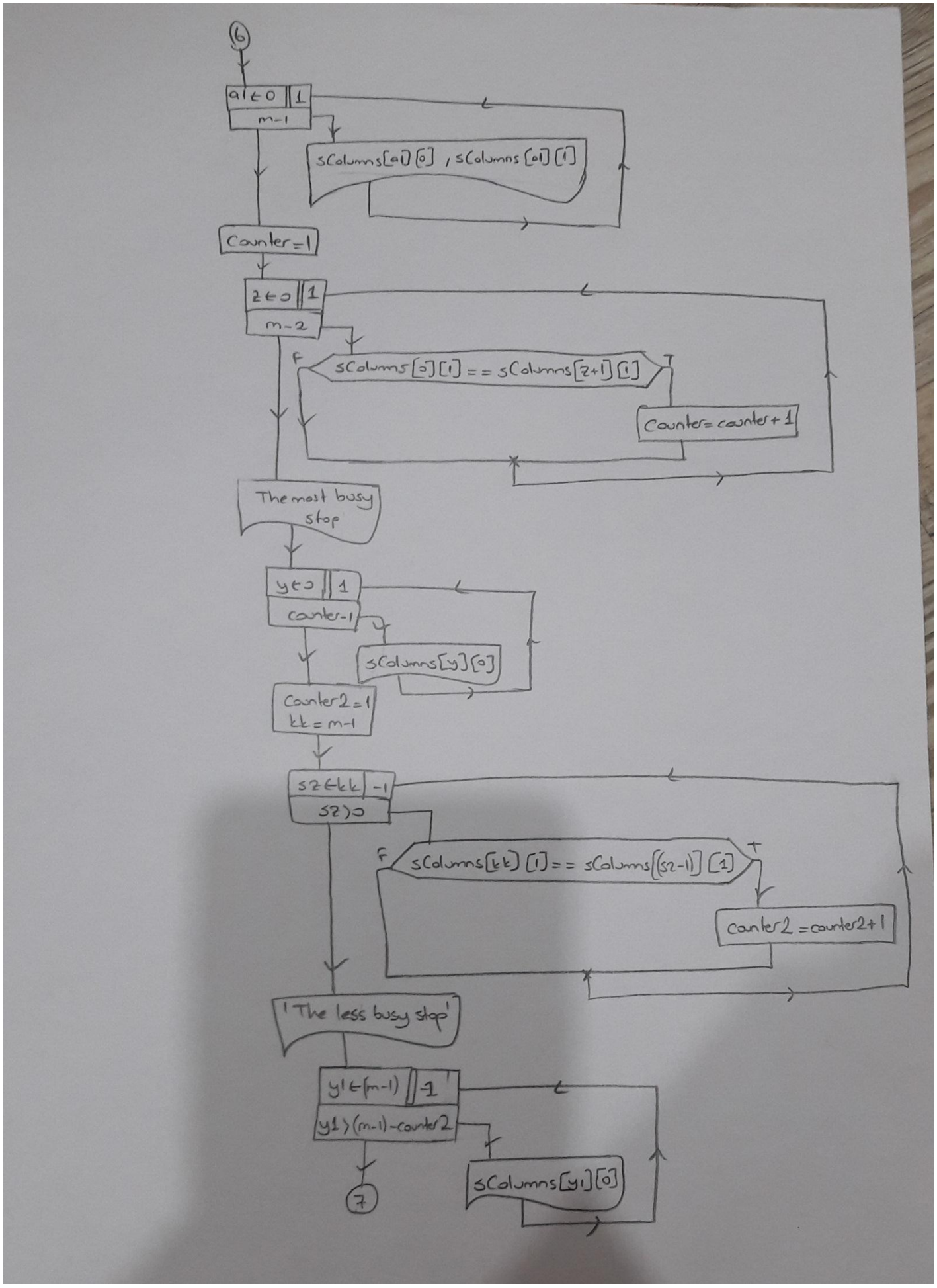
-which finds the most and least busy bus stops

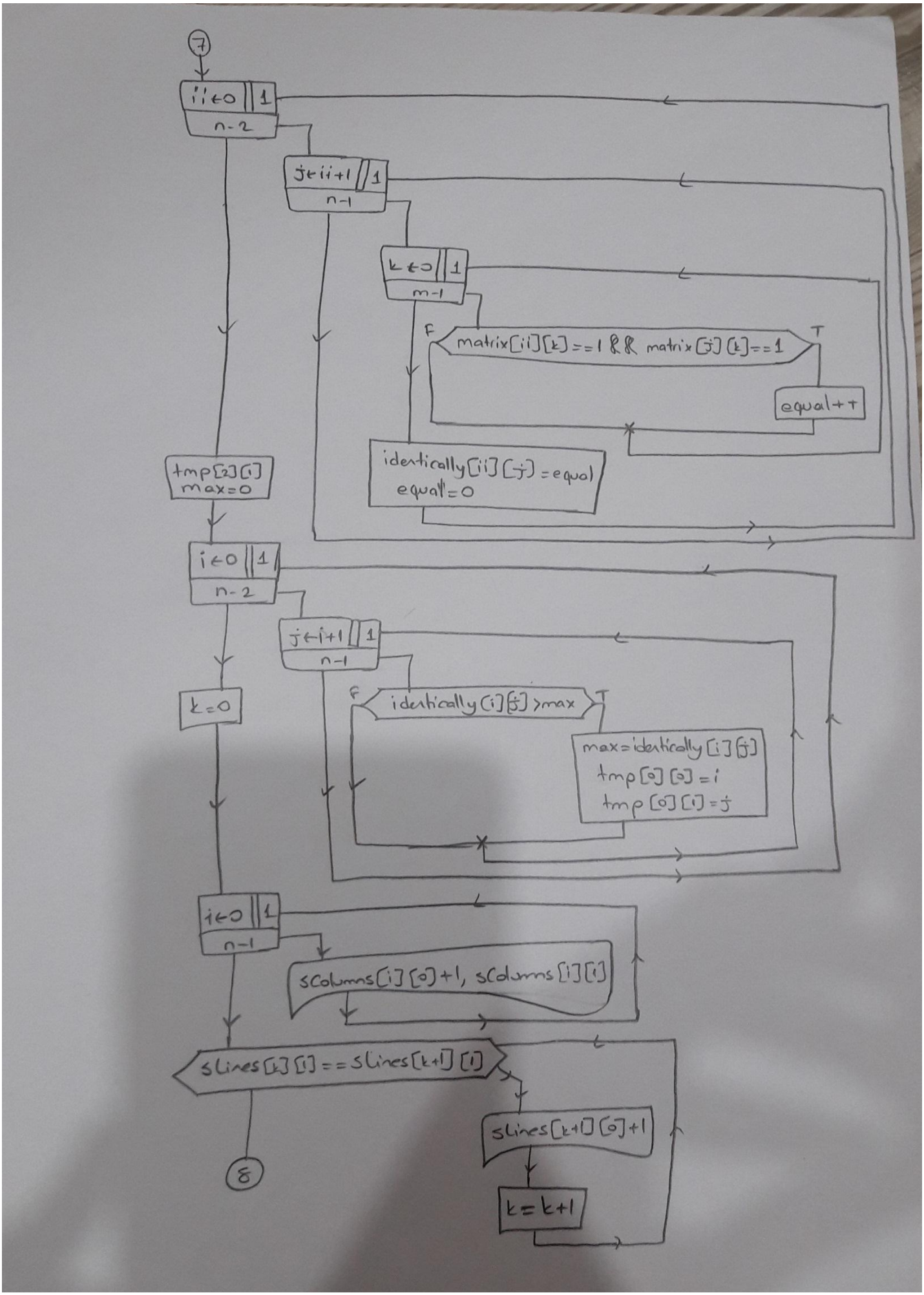
-which finds the most identical two bus lines in the city.

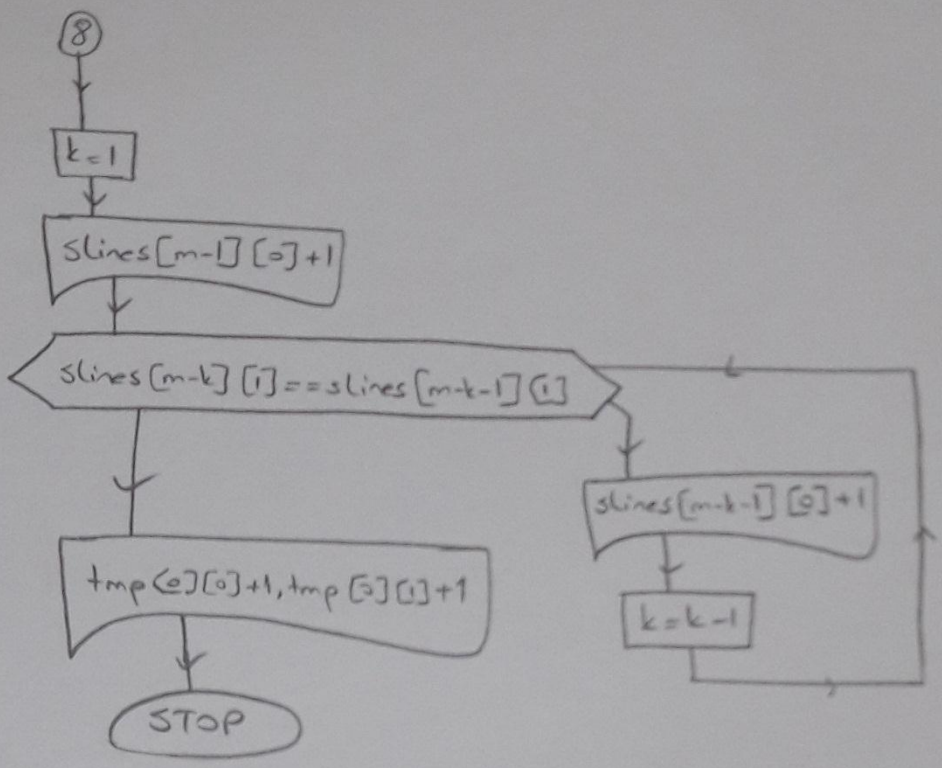
b. Flowchart


START
n,m
i←0 | 1
n-1
j←0 | 1
n-1
counter=0
slines[n][2]
matris[i][j]
a←0 | 1
n-1
b←0 | 1
n-1
temp
counter=counter+matris[a][b]
slines[a][2]=counter
counter=0
slines[a][1]=a+1
4←0 | 1
n-1
2
3
w←0 | 1
n-1
1
2
F
slines[q][2]>slines[w][2]
T
temp=slines[q][2]
slines[q][2]=slines[w][2]
slines[w][2]=temp
temp=slines[q][1]
slines[q][1]=slines[w][1]
slines[w][1]=temp
1


3
q1←0
1
n-1
slines[q][i], slines[q1][2]
sColums[m][2]
counter3=0
q←0
1
m-1
w←0
1
n-1
temp2
counter3+=matrix[w][q]
sColumn[q][i]=counter3
counter3=0
sColumn[q][0]=q+1
q1←0
1
5
m-1
w1←0
1
4
m-1
5
F
sColumns[q1][i] > sColumns[w1][i]
T
6
temp2=sColumns[q1][i]
sColumns[q1][i]=sColumns[w1][i]
sColumns[w1][i]=temp2
temp2=sColumns[q1][0]
sColumns[q1][0]=sColumns[w1][0]
sColumns[w1][0]=temp2
4


6
a1 ← 0 | 1
m-1
sColumns[a1][0] , sColumns[a1][1]
counter=1
2 ← 0 | 1
m-2
F
sColumns[0][1] == sColumns[2+1][1]
T
counter= counter+1
The most busy stop
y ← 0 | 1
counter-1
sColumns[y][0]
counter2=1
kk = m-1
s2 ← kk | -1
s2 > 0
F
sColumns[kk][1] == sColumns[(s2-1)][1]
T
counter2 = counter2+1
'The less busy stop'
y1 ← (m-1) | -1
y1 > (m-1)-counter2
7
sColumns[y1][0]


7
i←0 || 1
n-2
j←i+1 || 1
n-1
k←0 || 1
m-1
F
matrix[i][k]==1 && matrix[j][k]==1
T
equal++
identically[i][j]=equal
equal=0
tmp[2][i]
max=0
i←0 || 1
n-2
j←i+1 || 1
n-1
k=0
F
identically[i][j]>max
T
max=identically[i][j]
tmp[0][0]=i
tmp[0][1]=j
i←0 || 1
n-1
sColumns[i][0]+1, sColumns[i][i]
sLines[k][i]==sLines[k+1][i]
sLines[k+1][0]+1
k=k+1
8


8
k=1
slines[m-1][0]+1
slines[m-k][i]==slines[m-k-1][1]
slines[m-k-1][0]+1
k=k-1
tmp[0][0]+1, tmp[0][1]+1
STOP

c. Source Code

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
main(){
        int m; //Değişkenler Tanımlandı
        int n;
        int a1,a,b,q,w,z,l,q1,w1;
        int oneandzero;
        int counter = 0;
        time_t t;
         srand((unsigned) time(&t));
        printf("Bus Lines and Stops:\n");
        scanf("%d",&n);  // Bus Lines
        scanf("%d",&m);  // Bus Stops
            printf("\n");
        int matrix[n][m];  // Lines and Stops matrix
        for (int i=0;i<n;i++){  // matris içine 1 ve 0 atamaları yapıldı.
            for (int j=0;j<m;j++){
            /*      scanf("%d",&oneandzero);
                    matrix[i][j] = oneandzero;*/
            matrix[i][j] = rand() % 2;
            }
        }
        int sLines[n][2]; // 2 sütunlu matris. İlkk sütununda hattın yoğunluk değeri, ikinci sütununda hattın numarası tutulacak.
        for (a=0;a<n;a++){ // bu döngüde hattın yoğunluğu - 1 değerlerinin toplamı - hesaplandı
            for (b=0;b<m;b++){
```

```
                printf("%d ",matrix[a][b]);
                counter = counter + matrix[a][b]; // "counter" değişkeni hattaki 1 lerin toplam değerini hesaplayıp diziye atmak amacıyla satırın tamamının toplamına eşitlendi.
            }
            printf("\n");
            sLines [a][2] = counter; // Hesaplanan değer oluşturulan dizinin ilk sütununa atandı
            counter = 0; // Diğer hatların hesabı için sayaç sıfırlandı.
            sLines [a][1] = a+1; // dizinini ikinci sütununa hattın numarası kaydedildi.
        }

            printf("\n");
        // Aşağıdaki kısım toplam değerleri yazdırıyo. Çıktıda sıralanmış hali istiyor. yazdırmak gereksiz.
        /*printf("\n");
        for(a1=0;a1<n;a1++){
        printf("%d %d - ",sLines[a1][0], sLines[a1][1]);
        }*/
        int temp;
        for (q=0;q<n;q++){ //bu döngü de bubble sort ile hatlar yoğunluğuna göre büyükten küçüğe sıralandı.
            for (w=0;w<n;w++){
                if(sLines[q][2] > sLines [w][2]){
                    temp = sLines[q][2]; //sıralama işlemi bubble
                    sLines[q][2] = sLines[w][2];
                    sLines[w][2] = temp;
                    temp = sLines[q][1]; // Sıralama işlemi hattın sıra numarasını değiştirmemek için hem 2. sütuna da yapıldı.
                    sLines[q][1] = sLines[w][1];
                    sLines[w][1] = temp;
```

```
            }
        }
}

for(a1=0;a1<n;a1++){ // Sıralanmış şekilde hatlar ve yoğunlukları - A ŞIKKI -
printf("Bus Line %d : %d Stops.",sLines[a1][1], sLines[a1][2]);
printf("\n");
}

// Sütundaki 1 değerlerinin toplamı;
int sColumns [m][2];

int counter3 = 0;
for (q=0;q<m;q++){
    for (w=0;w<n;w++){
            counter3 += matrix[w][q];
    }
sColumns[q][1] = counter3;
counter3=0;
sColumns[q][0] = q+1;
}

// sütun toplam matrisinin sıralanması
int temp2;

for (q1=0;q1<m;q1++){
    for (w1=0;w1<m;w1++){
            if(sColumns[q1][1] > sColumns [w1][1]){
                    temp2 = sColumns[q1][1]; //sıralama işlemi bubble
```

```
                    sColumns[q1][1] = sColumns[w1][1];
                    sColumns[w1][1] = temp2;

                    temp2 = sColumns[q1][0]; //Sıralama işlemi hattın sıra numarasını değiştirmemek için hem 2. sütuna da yapıldı.  bubble
                    sColumns[q1][0] = sColumns[w1][0];
                    sColumns[w1][0] = temp2;
                }
            }
        }
        printf("\n");
        printf("\n");
         for(a1=0;a1<m;a1++){ // SIRALANMIŞ SÜTUNlar yazdırmak istersen
        printf("Line %d : %d Stops.",sColumns[a1][0], sColumns[a1][1]);
        printf("\n");
        }
        printf("\n");
        printf("\n");
        // Most busy stop
        counter = 1; // sıralanmış sütun toplam değerlerden aynı olan en yüksek değerleri/değeri buluyor
        for (z=0;z<m-1;z++){
            if(sColumns[0][1] == sColumns[z+1][1]){
                counter++;
            }
        }
        printf("The most busy stop: "); // en yüksek değerlerden/değerden oluşan sütun yoğunluklarını, aynı olan dütun değeri sayısı kadar döngüde işleyip ekrana yazdırıyoruz.
        for (int y=0;y<counter;y++){ // artan bir sıralama ile dizi sıralandığından 0 dan başlatıp arttırarak işlem yapınca en yüksek değerlere ulaşıyoruz.
            printf("Stop %d,  ",sColumns[y][0]);
```

```
}
// Less busy stop


int counter2 = 1; // sıralanmış sütun toplam değerlerden aynı olan en düşük değerleri/değeri buluyor
int kk = m-1;
for (int sz=kk;sz>0;sz--){
    if(sColumns[kk][1] == sColumns[(sz-1)][1]){
        counter2++;
    }
}
printf("\n");
printf("\n");
printf("The less busy stop: "); // en düşük değerlerden/değerden oluşan sütun yoğunluklarını, aynı olan dütun değeri sayısı kadar döngüde işleyip ekrana yazdırıyoruz.
for (int y1=(m-1);y1>((m-1)-counter2);y1--){ // azalan bir sıralama ile diziyi sıraladığımızdan tersten çalıştırınca en düşük değerleri elde etmiş oluyoruz.
    printf("Stop %d,  ",sColumns[y1][0]);
}
printf("\n");
printf("\n");
    printf("\n");
printf("\n");
    printf("\n");
printf("\n");
    printf("\n");
printf("\n");        printf("\n");
printf("\n");        printf("\n");
printf("\n");
```

```
		/* ------------------- */
		int lineStops[20][50], stopLines[20][50], tmp[2][1], identically[50][50];
		int equal = 0;
		int max=0;
//Her bir hattı, bir diğeri ile karşılaştırıp ortak durak sayısını ayrı bir matrise yazan döngü */
for(int ii=0;ii<n-1;ii++){
for(int j=ii+1;j<n;j++){
for(int k=0;k<m;k++){
if(matrix[ii][k]==1 && matrix[j][k]==1){
equal++;
}
}
identically[ii][j]=equal;
equal=0;
}
}
//Ortak durak sayısını içeren matristeki en yüksek değerin bulunup o değerin ait olduğu hatları tespit eden döngü
for(int i=0;i<n-1;i++){
for(int j=i+1;j<n;j++){
if(identically[i][j]>max){
max=identically[i][j];
tmp[0][0]=i;
tmp[0][1]=j;
}
}
}
		int k=0;
printf("\n");
for(int i=0;i<n;i++){
```

```
printf("Bus Line %d : %d stops", sColumns[i][0]+1, sColumns[i][1]);
printf("\n");
}
printf("\n");
//En yoğun durağın hangisi olduğunun yazdırılması işlemi
printf("The most busy bus stop : Stop %d ", sLines[0][0]+1);
while (sLines[k][1]==sLines[k+1][1]){
printf("and Stop %d",sLines[k+1][0]+1);
k++;
}
k=1;
printf("\n");
//En seyrek durağın hangi durak olduğunun yazdırılması işlemi
printf("The least busy bus stop : Stop %d ", sLines[m-1][0]+1);
while (sLines[m-k][1]==sLines[m-k-1][1]){
printf("and Stop %d",sLines[m-k-1][0]+1);
k--;
}
printf("\n");
printf("\n");
//En benzer hatların hangisi olduğunun yazdırılması işlemi
printf("The most identical bus lines : Bus Line %d and Bus Line%d",tmp[0][0]+1,tmp[0][1]+1);
return 0;

}
```

d . Analysis

```
Bus Lines and Stops:
5
6

1 0 0 0 0 1
0 0 0 1 1 0
1 1 0 1 1 1
0 0 0 1 0 0
0 1 1 1 1 1

Bus Line 3 : 5 Stops.
Bus Line 5 : 5 Stops.
Bus Line 1 : 2 Stops.
Bus Line 2 : 2 Stops.
Bus Line 4 : 1 Stops.


Line 4 : 4 Stops.
Line 5 : 3 Stops.
Line 6 : 3 Stops.
Line 1 : 2 Stops.
Line 2 : 2 Stops.
Line 3 : 1 Stops.


The most busy stop: Stop 4,

The less busy stop: Stop 3,









Bus Line 5 : 4 stops
Bus Line 6 : 3 stops
Bus Line 7 : 3 stops
Bus Line 2 : 2 stops
Bus Line 3 : 2 stops

The most busy bus stop : Stop 6
The least busy bus stop : Stop 2

The most identical bus lines : Bus Line 3 and Bus Line5
--------------------------------
Process exited after 2.241 seconds with return value 0
Press any key to continue . . .
```