Consider a university having a very big campus spread in acres of land. The university is undergoing computerization. All the departments (at-most 50) are to be connected to form the intranet of the university. You have to write a program, implementing **Prims algorithm**, which will suggest the network topology and also minimise the total length of cable for connecting all the departments. Input to the program will be names of all the departments and straight line distances between the departments (Only those pairs of departments between which cable can be laid will be given). Output of the program should be the minimum length of the cable required.

**input**

The first line will contain 2 natural numbers, N and M, separated by a blank space. N indicated the number of departments in the university and M indicates the number of pairs of departments where the cables can be laid. The following M lines will specify the distances between M pairs of departments as

dept1 dept2 distance

Where dept1 and dept2 are names of the departments (maximum 20 characters) and distance is a positive integer (>0). Assume that the given distances between each pairs of departments will be unique and these M lines will contain atleast one pair for each department.

**output**

The first line of the output will be names of the departments as they are included in the solution separated by blank space. If two or more departments are included at a time then their names should be printed in the alphabetic order. The next line will be the minimum length of cable required to form the intranet, terminated with a new line character.

**Input**

7 10

physics chemistry 8

biology physics 9

biology office 15

chemistry office 4

chemistry sanskrit 5

sanskrit office 7

english office 16

english sanskrit 19

english cs 12

sanskrit cs 6

**Output**

chemistry office sanskrit cs physics biology english

44

Testcase

7 10

physics chemistry 8

biology physics 9

biology office 15

chemistry office 4

chemistry sanskrit 5

sanskrit office 7

english office 16

english sanskrit 19

english cs 12

sanskrit cs 6

#include <stdio.h>

#include <string.h>

#define MAX\_DEPT 50

#define MAX\_CHAR 21

#define FALSE 0

#define TRUE 1

typedef struct \_connectivity {

char dept1[MAX\_CHAR];

char dept2[MAX\_CHAR];

int cableLen;

int isAdded;

}CONNECTIVITY;

void init(CONNECTIVITY deptConn[MAX\_DEPT], char deptNames[MAX\_DEPT][MAX\_CHAR], int deptDone[MAX\_DEPT]);

int readConnectivity(CONNECTIVITY deptConn[MAX\_DEPT], char deptNames[MAX\_DEPT][MAX\_CHAR],int \*numDept, int \*numCable);

void prims(CONNECTIVITY deptConn[MAX\_DEPT], char deptNames[MAX\_DEPT][MAX\_CHAR],int numDept, int numCable, int deptDone[MAX\_DEPT], int cableLen);

int addDeptName(char deptNames[MAX\_DEPT][MAX\_CHAR], int numDeptAdded, char dept[]);

int getCableIdx(CONNECTIVITY deptConn[MAX\_DEPT], char deptNames[MAX\_DEPT][MAX\_CHAR],int numDept, int numCable, int deptDone[MAX\_DEPT]);

void printDeptConn(CONNECTIVITY dept[MAX\_DEPT], int numCables);

void printDeptName(char deptNames[MAX\_DEPT][MAX\_CHAR], int numDept);

int main()

{

char deptNames[MAX\_DEPT][MAX\_CHAR];

int numDept, numCable,

deptDone[MAX\_DEPT];

CONNECTIVITY deptConn[MAX\_DEPT];

int smindex, index;

init(deptConn, deptNames, deptDone);

smindex = readConnectivity(deptConn, deptNames, &numDept, &numCable);

printf("%s %s",deptConn[smindex].dept1,deptConn[smindex].dept2);

deptConn[smindex].isAdded = TRUE;

index = getDeptNameIdx(deptNames,numDept, deptConn[smindex].dept1);

deptDone[index] = TRUE;

index = getDeptNameIdx(deptNames,numDept, deptConn[smindex].dept2);

deptDone[index] = TRUE;

prims(deptConn,deptNames,numDept,numCable,deptDone, deptConn[smindex].cableLen);

return 0;

}

void init(CONNECTIVITY deptConn[MAX\_DEPT], char deptNames[MAX\_DEPT][MAX\_CHAR], int deptDone[MAX\_DEPT])

{

int i;

for(i=0; i < MAX\_DEPT; i++){

deptConn[i].cableLen = 0;

deptConn[i].isAdded = FALSE;

deptDone[i] = FALSE;

}

}

int readConnectivity(CONNECTIVITY deptConn[MAX\_DEPT], char deptNames[MAX\_DEPT][MAX\_CHAR],int \*numDept, int \*numCable)

{

char dept1[MAX\_CHAR], dept2[MAX\_CHAR];

int numDeptAdded = 0;

int smindex;

int retVal;

int i;

scanf("%d %d",numDept, numCable);

for(i = 0; i < \*numCable; i++){

scanf("%s %s %d",dept1,dept2,&(deptConn[i].cableLen));

if(strcmp(dept1,dept2) >=0 ){

strcpy(deptConn[i].dept1,dept2);

strcpy(deptConn[i].dept2,dept1);

}

else{

strcpy(deptConn[i].dept1,dept1);

strcpy(deptConn[i].dept2,dept2);

}

if( i == 0)

smindex = i;

else if(deptConn[i].cableLen < deptConn[smindex].cableLen)

smindex = i;

else if(deptConn[i].cableLen == deptConn[smindex].cableLen){

retVal = strcmp(deptConn[smindex].dept1,deptConn[i].dept1);

if( retVal > 0 )

smindex = i;

else if( retVal == 0){

if(strcmp(deptConn[smindex].dept2,deptConn[i].dept2) > 0){

smindex = i;

}

}

}

numDeptAdded += addDeptName(deptNames,numDeptAdded,dept1);

numDeptAdded += addDeptName(deptNames,numDeptAdded,dept2);

}

return smindex;

}

void prims(CONNECTIVITY deptConn[MAX\_DEPT], char deptNames[MAX\_DEPT][MAX\_CHAR],int numDept, int numCable, int deptDone[MAX\_DEPT], int cableLen)

{

int numDeptAdded = 2, totalCableLen = cableLen;

int index, deptNameIdx;

while( numDeptAdded < numDept ){

index = getCableIdx(deptConn,deptNames,numDept,numCable,deptDone);

deptNameIdx = getDeptNameIdx(deptNames,numDept, deptConn[index].dept1);

if( deptDone[deptNameIdx] == FALSE ){

printf(" %s",deptConn[index].dept1);

}

else{

printf(" %s",deptConn[index].dept2);

deptNameIdx = getDeptNameIdx(deptNames,numDept, deptConn[index].dept2);

}

numDeptAdded++;

deptDone[deptNameIdx] = TRUE;

deptConn[index].isAdded == TRUE;

totalCableLen += deptConn[index].cableLen;

}

printf("\n%d\n",totalCableLen);

}

int getCableIdx(CONNECTIVITY deptConn[MAX\_DEPT], char deptNames[MAX\_DEPT][MAX\_CHAR],int numDept, int numCable, int deptDone[MAX\_DEPT])

{

int i, first = TRUE;

int oldIndex1, oldIndex2;

int index1, index2;

int smindex = -1;

int retVal;

for(i = 0; i < numCable; i++){

if( deptConn[i].isAdded == FALSE ){

index1 = getDeptNameIdx(deptNames,numDept,deptConn[i].dept1);

index2 = getDeptNameIdx(deptNames,numDept,deptConn[i].dept2);

if( deptDone[index1] ^ deptDone[index2] ){

if( smindex == -1){

smindex = i; oldIndex1 = index1; oldIndex2 = index2;

}

else if(deptConn[i].cableLen < deptConn[smindex].cableLen)

smindex = i;

else if(deptConn[i].cableLen == deptConn[smindex].cableLen){

if( !deptDone[oldIndex1] && !deptDone[index1] ){

if( compDepts(deptNames,oldIndex1,index1,oldIndex2, index2) )

smindex = i;

}

else if( !deptDone[oldIndex1] && !deptDone[index2] ){

if( compDepts(deptNames,oldIndex1,index2,oldIndex2, index1) )

smindex = i;

}

else if( !deptDone[oldIndex2] && !deptDone[index1] ){

if( compDepts(deptNames,oldIndex2,index1,oldIndex1, index1) )

smindex = i;

}

else if( !deptDone[oldIndex2] && !deptDone[index2] ){

if( compDepts(deptNames,oldIndex2,index2,oldIndex1, index1) )

smindex = i;

}

}

}

}

}

return smindex;

}

int compDepts(char deptNames[MAX\_DEPT][MAX\_CHAR], int f1, int f2, int s1, int s2)

{

int retVal;

retVal = strcmp(deptNames[f1],deptNames[f2]);

if(retVal > 0 )

return TRUE;

else if( retVal == 0 && strcmp(deptNames[s1],deptNames[s1]) > 0 )

return TRUE;

return FALSE;

}

int getDeptNameIdx(char deptNames[MAX\_DEPT][MAX\_CHAR], int numDept, char dept[MAX\_CHAR])

{

int i;

for(i=0;i<numDept;i++){

if(strcmp(deptNames[i],dept) == 0){

return i;

}

}

}

int addDeptName(char deptNames[MAX\_DEPT][MAX\_CHAR], int numDeptAdded, char dept[])

{

int i;

int found = FALSE;

for(i = 0; i < numDeptAdded && !found; i++){

if(strcmp(deptNames[i],dept) == 0 )

found = TRUE;

}

if(found == FALSE){

strcpy(deptNames[numDeptAdded],dept);

return 1;

}

return 0;

}

void printDeptConn(CONNECTIVITY dept[MAX\_DEPT], int numCables)

{

int i;

for(i=0; i < numCables; i++)

printf("%s %s %d\n",dept[i].dept1,dept[i].dept2,dept[i].cableLen);

}

void printDeptName(char deptNames[MAX\_DEPT][MAX\_CHAR], int numDept)

{

int i;

for( i = 0; i < numDept; i++){

printf("%s ",deptNames[i]);

}

printf("\n");

}