#include <iostream>  
using namespace std;  
  
#define MaxInt 32767    
#define MVNum 100   
#define OK 1  
typedef char VerTexType;  
typedef int  ArcType;  
typedef int  Status;  
  
typedef struct{  
    VerTexType vexs[MVNum];  
    ArcType arcs[MVNum][MVNum];  
    int vexnum,arcnum;  
}AMGraph;  
  
Status LocateVer(AMGraph G,VerTexType v){  
    for (int i = 0; i < G.vexnum; ++i) {  
        if (G.vexs[i]==v){  
            return i;  
        }  
    }  
    return -1;  
}  
//采用邻接矩阵创建无向图  
Status CreateUDN(AMGraph &G){  
    int i,j,k;  
       // cout<<"please input the vexnum and the arcnum:"<<endl;  
   cout<<"1.输入总顶点数和总边数"<<endl;  
    cin>>G.vexnum;  
    cin>>G.arcnum;  
      
    cout<<"2.依次输入点的信息存入顶点表中"<<endl;  
    for ( i = 0; i < G.vexnum; ++i) {  
        cin>>G.vexs[i] ;  
    }  
      cout<<"3.初始化邻接矩阵，使每个权值初始化为最大值"<<endl;  
    for (i  = 0; i < G.vexnum; ++i) {  
        for ( j = 0; j < G.vexnum; ++j) {  
            G.arcs[i][j]=MaxInt;  
        }  
    }  
    
    cout<<"4.构造邻接矩阵,依次输入每条边依附的顶点和权值"<<endl;  
    for ( k = 0; k < G.arcnum; ++k) {  
        //1.定义两个顶点和边的权值  
        VerTexType v1,v2;  
        ArcType w;  
        cout<<"Please enter the vertex and weight of each edge in turn:"<<endl;  
        cin>>v1>>v2>>w  
        i=LocateVer(G,v1);  
        j=LocateVer(G,v2);  
        G.arcs[i][j]=w;  
        G.arcs[j][i]=G.arcs[i][j];  
  
  
    }  
    return OK;  
    cout<<endl;  
}  
  
int main() {  
    cout<<"----------------Use the Adjacency Matrix CreateUDN------------"<<endl;  
    AMGraph G;  
    CreateUDN(G);  
    cout<<"----------------Show the Adjacency Matrix---------------------"<<endl;  
    for (int i = 0; i < G.vexnum; ++i) {  
        for (int j = 0; j < G.vexnum; ++j) {  
            if (j!=G.vexnum-1){  
                if (G.arcs[i][j]!=MaxInt){  
                    cout<<G.arcs[i][j]<<"\t";  
                } else{  
                    cout << "∞" << "\t";  
                }  
            }else{  
                if(G.arcs[i][j]!=MaxInt){  
                    cout<<G.arcs[i][j]<<endl;  
                } else{  
                    cout << "∞" <<endl;  
                }  
            }  
  
        }  
  
    }  
    cout<<endl;  
    return 0;  
}