```cpp
#include "MGraph.h"
#include "CSTree.h"
//从v出发深度优先遍历连通图（连通子图）G，建立以T为根的生成树
void DFSTree(MGraph G, int v, CSTree &T){
    visited[v] = true;
    bool first = true;  //用以标识当前结点是不是树的第一个结点
    CSNode *p,*q;
    for(int w=FirstAdjVex(G, G.vexs[v]); w>=0; w=NextAdjVex(G, G.vexs[v], G.vexs[w]) ){
        if(!visited[w]){
            p = new CSNode;                         //分配孩子结点
            *p = {GetVex(G,w), NULL, NULL};
            if(first){                              //w是v的第一个未被访问的邻接顶点
                T->lchild = p;                      //是根的左孩子结点
                first = false;
            }
            else                                    //w是v的其他未被访问的邻接顶点
                q->nextsibling = p;                   //是上一个邻接顶点的右兄弟结点
            q = p;
            DFSTree(G,w,q);
        }//if
    }
}//DFSTree

//（UDG）建立树的深度优先生成森林(左孩子，右兄弟表示法)
//生成树用的是孩子兄弟表示法，所以最终返回会的将是一个用二叉树表示的树/森林
void DFSForest(MGraph G, CSTree &T){
    T = NULL;
    for(int v=0; v<G.vexnum; v++)        //初始化访问数组
        visited[v] = false;

    CSNode *p=NULL;
    CSNode *q=NULL;
    for(int v=0; v<G.vexnum; v++)
        if(!visited[v]){                 //第v顶点为新的生成树的根节点
            p = new CSNode;              //分配根结点
            *p = {GetVex(G,v), NULL, NULL};      //为p赋值
            if(!T)                       //是第一颗生成树的根节点
                T=p;
            else//是其他生成树的根节点（在孩子兄弟二叉树中，应放在最后结点的nextsibling）
                q->nextsibling = p;
            //q指示当前生成树的根，当时第一颗生成树是，q=T
            q=p;
        DFSTree(G,v,p);      //建立以p为根的生成树
        }
}

int main(){
    MGraph G;
    CSTree T;
    CreateGraph(G);
    PrintAdjMatrix(G);
    DFSForest(G, T);

    cout << T->data << endl;
    cout << T->lchild->data <<endl;
    return 0;
}

```