一、树的重心的定义：  
在一棵树中，如果我们选择某个结点为根，可以使得它的所有子树中最大的子树最小，那么这个结点就被称作这棵树的重心。

二、树重心的性质：  
1．以重心为树根时，所有子树的大小不超过全树大小的一半。

2．如果树的所有边权都为1，那么树中所有点到某个点的距离和中，到重心的距离和是最小的；如果有两个重心，那么到它们的距离和一样。反过来，距离和最小的点一定是重心。

3．把两棵树通过一条边相连得到一棵新的树，则新的重心在较大的一棵树一侧的连接点与原重心之间的简单路径上。如果两棵树大小一样，则重心就是两个连接点。  
另外一种说法：把两棵树通过一条边相连得到一棵新的树，那么新的树的重心在连接原来两棵树的重心的路径上。

4．在一棵树上添加或删除一个叶子，那么它的重心最多只移动一条边的距离。

5．一颗树最多只有两个重心，且这两个重心一定是相邻的。此时树一定有偶数个节点，且可以被划分为两个大小相等的分支，每个分支各自包含一个重心。

6．一棵树的重心一定在根节点所在的重链上。

7．一棵树的重心一定是以该树根节点重儿子为根的子树的重心的祖先。

8．往树上增加或减少一个叶子，如果原节点数是奇数，那么重心可能增加一个，原重心仍是重心；如果原节点数是偶数，重心可能减少一个，另一个重心仍是重心。

三、找重心  
遍历树，对每个点求 fufu 表示最大子树。边遍历边求最小 fufu。

int n, minn = 1e8;

void dfs(int u,int pre){

sz[u] = 1; //自己也要算进去

mx[u] = 0;

for(int i = head[u];i;i = nxt[i]){

int v = ver[i];

if(v == pre) continue;

dfs(v,u);

sz[u] += sz[v]; //dfs搜索完v之后，更新sz[u]

mx[u] = max(mx[u],sz[v]); //更新最大子树的大小

}

mx[u] = max(mx[u], n-sz[u]); //还要与到根节点那颗子树的大小比较

if(mx[u] < minn) minn = mx[u];

}

int main()

{

int u,v;

cin >> n;

for(int i = 1;i <= n-1;i++){ //建立边

cin >> u >> v;

addedge(u,v);

}

dfs(1,0); //进行dfs遍历

int len=0,p[2];

for(int i = 1;i <= n;i++){ //寻找树的重心

if(mx[i] == minn)

p[len++] = i;

}

cout << len << endl; //输出

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

cout << p[i] << endl;

}

}