题意

- 给定一个无向图
- 求出该图的一棵生成树，满足1号点的度数不超过k，而且生成树最小
- 总点数不超过50

思路分析

- 由于1号点有限制，那么先不考虑1号点
  去掉无向图中的1号点，原图被分成m个连通子图，对这m个连通子图求m棵最小生成树
- 将1号点和这m棵树连起来，此时得到了一个1 号点度为m的生成树
- 1号点再连一个点，会形成一个环，去掉这个环上的另外一条边，仍然是一个生成树，而此时得到了一个1号点度为m+1的子图
- 当无法通过上一步骤减小生成树权值的时候，就得到了最终的答案

具体实现细节

定义变量

变量名	类型	功能	1.
n_edge	int	边数	
k	int	1号点的度数限制	
edge[10005]	cre	边	
maps[55][55]	int	原图的带权邻接矩阵	
maps2[55][55]	int	生成树的邻接矩阵	
dis[55]	int	1号点和每棵生成树的距离	
to[55]	int	每棵树距离1号点最近的点	
dp[55]	cre	在当前求得的生成树中路径1->i上与1点无关联且权值最大的边	

代码

```
1  #include<iostream>
```

```cpp
#include<map>
#include<cstring>
#include<algorithm>
using namespace std;
typedef long long LL;
const int INF =0x3f3f3f3f;
struct cre{
    int u,v,w;
};
bool cmp(cre a,cre b)
{
    return a.w<b.w;
}
/*
    全局变量
*/
int n,k;
int fa[55];
map<string,int>po;
int n_po = 0;
cre edge[100005];
int n_edge = 0;
int maps[55][55];
int maps2[55][55];
int ans = 0;
int du = 0;
int dis[55],to[55];
cre dp[55];
/*
    函数
*/
void dfs(int now,int pre)
{
    for(int v = 2;v<=n_po;v++)
    {
        if(v == pre) continue;
        if(maps2[now][v])
        {
            if(dp[v].w != -1)
            {
                if(dp[now].w < maps[now][v]) dp[v].u = now, dp[v].v = v, dp[v].w = maps[now][v];
                else dp[v] = dp[now];
            }
            dfs(v, now);
        }
```

```
    }
}
int getf(int a)
{
    if(fa[a]==a)
    return a;

    return fa[a] = getf(fa[a]);
}
void kru()
{
    for(int i=1;i<=n_po;i++)
    {
        fa[i] = i;
    }
    for(int i=1;i<=n_edge;i++)
    {
        if(edge[i].u ==1 ||edge[i].v ==1)
            continue;
        int f1 = getf(edge[i].u) , f2 = getf(edge[i].v);
        if(f1 != f2)
        {
            fa[f1] = f2;
            maps2[edge[i].u][edge[i].v] = 1;
            maps2[edge[i].v][edge[i].u] = 1;
            ans+=edge[i].w;
        }
    }
    return ;
}
void get_dis()
{
    for(int i=1;i<=n_po;i++) dis[i] = INF;
    for(int i=2;i<=n_po;i++)
    {
        if(maps[1][i]==-1) continue;
        if(maps[1][i] <dis[getf(i)])
        {
            dis[getf(i)] = maps[1][i];
            to[getf(i)] = i;
        }
    }
    for(int i=1;i<=n_po;i++)
    {
        if(dis[i]!=INF)
        {
```

```
            maps2[1][to[i]] = 1;
            maps2[to[i]][1] = 1;
            du++;
            ans += dis[i];
        }
    }
}
void add()
{

    for(du = du+1;du<=k;du++)
    {
        dp[1].w = -1;
        for(int v = 2;v<=n_po;v++)
        {
            if(maps2[1][v] == 1) dp[v].w = -1;
            else dp[v].w = 0;
        }
        dfs(1,0);

        int minn=12345,too;
        for(int v = 2; v <= n_po; v++)
        {
            if(maps[1][v] == -1) continue;
            if(minn > maps[1][v]-dp[v].w) minn = maps[1][v]-dp[v].w, too
= v;

        }
        if(minn >= 0) break;
        maps2[1][too] = maps2[too][1] = 1;
        maps2[dp[too].u][dp[too].v] = maps2[dp[too].v][dp[too].u] = 0;
        ans += minn;
    }
}
int main()
{
    ios::sync_with_stdio(false);
    cin>>n;
    string a,b;
    int d;
    po["Park"] = ++n_po;
    memset(maps,-1,sizeof(maps));
    for(int i=1;i<=n;i++)
    {
        cin>>a>>b>>d;
        if(!po[a]) po[a] = ++n_po;
```

```
        if(!po[b]) po[b] = ++n_po;
        edge[++n_edge].u = po[a];
        edge[n_edge].v = po[b];
        edge[n_edge].w = d;
        maps[po[a]][po[b]] = d;
        maps[po[b]][po[a]] = d;
    }
    cin>>k;
    sort(edge+1,edge+n_edge+1,cmp);
    kru();
    get_dis();
    add();
    cout<<"Total miles driven: "<<ans<<endl;
}
```