```
#include <cstdio>
#include <climits>
#include <algorithm>
#include <vector>
#include <cstring>
using namespace std;

typedef pair< int, int > pii;
const int INF = INT_MAX;
const int MAX = 1<<14;
const int LOG = 14;

vector< pii > G[MAX];
int root[MAX][LOG], dist[MAX], pi[MAX], lvl[MAX];

void dfs(int par, int u, int depth) {
        int sz = G[u].size(), i, v, w;
        lvl[u] = depth;
        for(i = 0; i < sz; i++) {
                v = G[u][i].first;
                w = G[u][i].second;
                if(v != par) {
                        dist[v] = dist[u] + w;
                        pi[v] = u;
                        dfs(u, v, depth+1);
                }
        }
}

void calcRoot(int n) {
        int i, j;
        memset(root, -1, sizeof root);
        for(i = 1; i <= n; i++) root[i][0] = pi[i];
        for(j = 1; 1<<j < n; j++)
                for(i = 1; i <= n; i++)
                        if(root[i][j-1]!=-1)
                                root[i][j] = root[root[i][j-1]][j-1];
}

int lca(int p, int q) {
        int i, stp;
        if(lvl[p] < lvl[q]) swap(p, q);
        for(stp = 1; 1<<stp <= lvl[p]; stp++); stp--;
        for(i = stp; i >= 0; i--)
                if(lvl[p] - (1<<i) >= lvl[q])
                        p = root[p][i];
        if(p == q) return p;
        for(i = stp; i >= 0; i--)
                if(root[p][i]!=-1 && root[p][i]!=root[q][i])
                        p = root[p][i], q = root[q][i];
        return pi[p];
}

int find(int p, int t) {
        int i, stp;
        for(stp = 1; 1<<stp <= lvl[p]; stp++); stp--;
        for(i = stp; i >= 0; i--)
                if(lvl[p] - (1<<i) >= t)
                        p = root[p][i];
        return p;
}

int main() {
        int test, n, i, u, v, w, st, en, k;
        char query[8];
        scanf("%d", &test);
        while(test--) {
                scanf("%d", &n);
                for(i = 1; i <= n; i++) {
                        G[i].clear();
                        dist[i] = 0;
                        pi[i] = -1;
                }
```

```
                for(i = 1; i < n; i++) {
                        scanf("%d%d%d", &u, &v, &w);
                        G[u].push_back(pii(v, w));
                        G[v].push_back(pii(u, w));
                }
                dist[1] = 0;
                dfs(-1, 1, 0);
                calcRoot(n);
                while(scanf("%s", query)==1) {
                        if(query[1]=='O') break;
                        scanf("%d%d", &st, &en);
                        u = lca(st, en);
                        if(query[1]=='I') printf("%d\n", dist[st]+dist[en]-2*dist[u]
);
                        else if(query[1]=='T') {
                                scanf("%d", &k);
                                if(lvl[st]-lvl[u]+1 >= k) v = find(st, lvl[st]-k+1);
                                else v = find(en, 2*lvl[u]+k-lvl[st]-1);
                                printf("%d\n", v);
                        }
                }
                printf("\n");
        }
        return 0;
}
```