```
// Encontrar puentes.
#include ...
using namespace std;

const int MAXN = 10005;

vector<int> g[MAXN];
int p[MAXN], d[MAXN], low[MAXN], tick;

int find(int x) {
    return p[x] == x ? x : p[x] = find(p[x]);
}

int link(int x, int y) {
    int a = find(x), b = find(y);
    if (a != b) {
        p[a] = b;
    }
}

// It's assumed that there is at most one edge
// between two nodes.
void dfs(int u, int parent = -1) {
    d[u] = low[u] = tick++;
    foreach(out, g[u]) {
        int v = *out;
        if (v == parent) continue;
        if (d[v] == -1) {
            dfs(v, u);
            low[u] = min(low[u], low[v]);
        } else {
            low[u] = min(low[u], d[v]);
        }

        if (low[v] > d[u]) {
            //printf("edge from %d to %d is a bridge\n", u + 1, v + 1);
            link(u, v);
        }
    }
}

int main(){
    int n, e, q;
    while (scanf("%d %d %d", &n, &e, &q) == 3) {
        if (n == 0 and e == 0 and q == 0) break;
        for (int i = 0; i < n; ++i) {
            g[i].clear();
            p[i] = i;
            d[i] = -1;
        }
        // read edges
        for (int i = 0; i < e; ++i) {
            int u, v; scanf("%d %d", &u, &v);
            u--, v--;
            g[u].push_back(v);
            g[v].push_back(u);
        }

        tick = 0;
        for (int i = 0; i < n; ++i) {
```

```
            if (d[i] == -1) dfs(i);
        }

        // read queries
        for (int i = 0; i < q; ++i) {
            int u, v; scanf("%d %d", &u, &v);
            u--, v--;

            if (find(u) == find(v)) {
                puts("Y");
            } else {
                puts("N");
            }
        }
        puts("-");
    }
    return 0;
}
```