January 2006 Problem 'prom' Analysis

by Bruce Merry

Construct a directed graph with cows as vertices and ropes as directed edges. A set of cows forms a group if and only if there are more than one of them (a group) and they form a strongly connected component. Any good algorithms textbook should have a discussion of strongly connected components and how to find them.

```
#include <fstream>
#include <vector>
#include <algorithm>

using namespace std;

struct cow {
    int id;
    vector<int> edges;
};

static int N;
static vector<cow> cows;
static vector<int> component;
static int G = 0;

static int dfs(int x) {
    static int pool = 0;
    int top;

    component.push_back(x);
    cows[x].id = pool++;
    top = cows[x].id;
    for (vector<int>::iterator e = cows[x].edges.begin(); e !=
cows[x].edges.end(); e++) {
        if (cows[*e].id == -1)
            top <?= dfs(*e);
        else
            top <?= cows[*e].id;
    }

    if (top == cows[x].id) {
        int sz = 0;
        while (component.back() != x)
        {
            sz++;
            cows[component.back()].id = INT_MAX;
            component.pop_back();
        }
        sz++;
        component.pop_back();
        cows[x].id = INT_MAX;
        if (sz > 1) G++;
    }
    return top;
```

```
}

int main() {
    int M;

    ifstream in("prom.in");
    in >> N >> M;
    cows.resize(N);

    for (int i = 0; i < M; i++) {
        int A, B;
        in >> A >> B;
        assert(1 <= A && A <= N);
        assert(1 <= B && B <= N);
        assert(A != B);
        A--;
        B--;
        cows[A].edges.push_back(B);
    }

    for (int i = 0; i < N; i++)
        cows[i].id = -1;
    for (int i = 0; i < N; i++)
        if (cows[i].id == -1) dfs(i);

    ofstream out("prom.out");
    out << G << "\n";
    return 0;
}
```