Homework Assignment #5: Programming Exercise #2

Note

This assignment constitutes 4% of your grade and is due 2PM Thursday, June 4, 2009. Please write or type your answers/code on A4 (or similar size) paper. Drop your homework by the due time in Yih-Kuen Tsay's mail box on the first floor of Management College Building II. Late submission will be penalized by 20% for each working day overdue. You may discuss the problems with others, but copying answers/code is strictly forbidden.

Your work will be graded according to its correctness and presentation. Specifically, you should demonstrate evidences showing that your program is correct. You should also organize and document your program in such a way that your classmates, for example, can understand it.

Problem

Solve either Problem B “My Bad” or Problem E “Fare and Balanced” of the 2009 Annual ACM International Collegiate Programming Contest World Finals (see the appended).

Please prepare an input file with more interesting cases and test your program on the input. In the documentation of your program, you should describe how you have applied the algorithmic techniques, in particular design by induction, learned in class.

Problem B
My Bad
Input file: bad.in

A logic circuit maps its input through various gates to its output with no feedback loops in the circuit. The input and output are an ordered set of logical values, represented here by ones and zeros. The circuits we consider are comprised of *and* gates (which output 1 only when their two inputs are both 1), *or* gates (which output 1 when one or both of their inputs are 1), *exclusive or* (xor) gates (which output 1 only when exactly one of the two inputs is 1), and *not* gates (which output the complement of their single input). The figures below show two circuits.

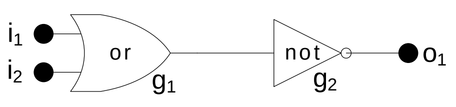

Figure 1: Circuit with 2 gates

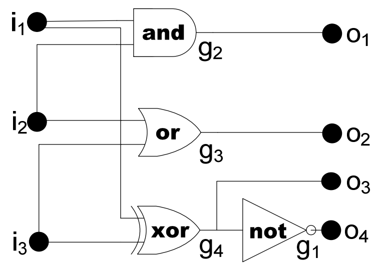

Figure 2: Circuit with 4 gates

Unfortunately, real gates sometimes fail. Although the failures may occur in many different ways, this problem limits attention to gates that fail in one of three ways: 1) always inverting the correct output, 2) always yielding 0, and 3) always yielding 1. In the circuits for this problem, at most one gate will fail.

You must write a program that analyzes a circuit and a number of observations of its input and output to see if the circuit is performing correctly or incorrectly. If at least one set of inputs produces the wrong output, your program must also attempt to determine the unique failing gate and the way in which this gate is failing. This may not always be possible.

Input

The input consists of multiple test cases, each representing a circuit with input and output descriptions. Each test case has the following parts in order.

1. A line containing three positive integers giving the number of inputs ($N \le 8$), the number of gates ($G \le 19$), and the number of outputs ($U \le 19$) in the circuit.
2. One line of input for each gate. The first line describes gate g_1. If there are several gates, the next line describes gate g_2, and so on. Each of these lines contains the gate type (`a` = *and*, `n` = *not*, `o` = *or*, and `x` = *exclusive or*), and identification of the input(s) to the gate. Gate input comes from the circuit inputs (`i1`, `i2`, …) or the output of another gate (`g1`, `g2`, …).
3. A line containing the numbers of the gates connected to the U outputs u_1, u_2, …. For example, if there are three outputs, and u_1 comes from g_5, u_2 from g_1, and u_3 from g_4, then the line would contain: `5 1 4`
4. A line containing an integer which is the number of observations of the circuit's behavior (B).
5. Finally B lines, each containing N values (ones and zeros) giving the observed input values and U values giving the corresponding observed output values. No two observations have the same input values.

Consecutive entries on any line of the input are separated by a single space. The input is terminated with a line containing three zeros.

Output

For each circuit in the input, print its case number (starting with 1), followed by a colon and a blank, and then the circuit analysis, which will be one of the following (with # replaced by the appropriate gate number):

```
No faults detected
Gate # is failing; output inverted
Gate # is failing; output stuck at 0
Gate # is failing; output stuck at 1
Unable to totally classify the failure
```

The circuits pictured in Figure 1 and Figure 2 are used in the first and last sample test cases.

Sample Input

```
2 2 1
o i1 i2
n g1
2
2
1 0 0
0 0 1
2 1 1
a i1 i2
1
1
1 0 1
2 1 1
a i1 i2
1
2
1 0 1
1 1 1
1 1 1
n i1
1
2
1 1
0 0
3 4 4
n g4
a i1 i2
o i2 i3
x i3 i1
2 3 4 1
4
0 1 0 0 1 0 1
0 1 1 0 1 1 0
1 1 1 0 1 0 1
0 0 0 0 0 0 1
0 0 0
```

Output for the Sample Input

```
Case 1: No faults detected
Case 2: Unable to totally classify the failure
Case 3: Gate 1 is failing; output stuck at 1
Case 4: Gate 1 is failing; output inverted
Case 5: Gate 2 is failing; output stuck at 0
```

Problem E
Fare and Balanced
Input: fare.in

Handling traffic congestion is a difficult challenge for young urban planners. Millions of drivers, each with different goals and each making independent choices, combine to form a complex system with sometimes predictable, sometimes chaotic behavior. As a devoted civil servant, you have been tasked with optimizing rush-hour traffic over collections of roads.

All the roads lie between a residential area and a downtown business district. In the morning, each person living in the residential area drives a route to the business district. The morning commuter traffic on any particular road travels in only one direction, and no route has cycles (morning drivers do not backtrack).

Each road takes a certain time to drive, so some routes are faster than others. Drivers are much more likely to choose the faster routes, leading to congestion on those roads. In order to balance the traffic as much as possible, you are to add tolls to some roads so that the perceived “cost” of every route ends up the same. However, to avoid annoying drivers too much, you must not levy a toll on any driver twice, no matter which route he or she takes.

Figure 5 shows a collection of five roads that form routes from the residential area (at intersection 1) to the downtown business district (at intersection 4). The driving cost of each road is written in large blue font. The dotted arrows show the three possible routes from 1 to 4. Initially the costs of the routes are 10, 8 and 12. After adding a toll of cost 2 to the road connecting 1 and 4 and a toll of cost 4 to the road connecting 3 and 4, the cost of each route becomes 12.

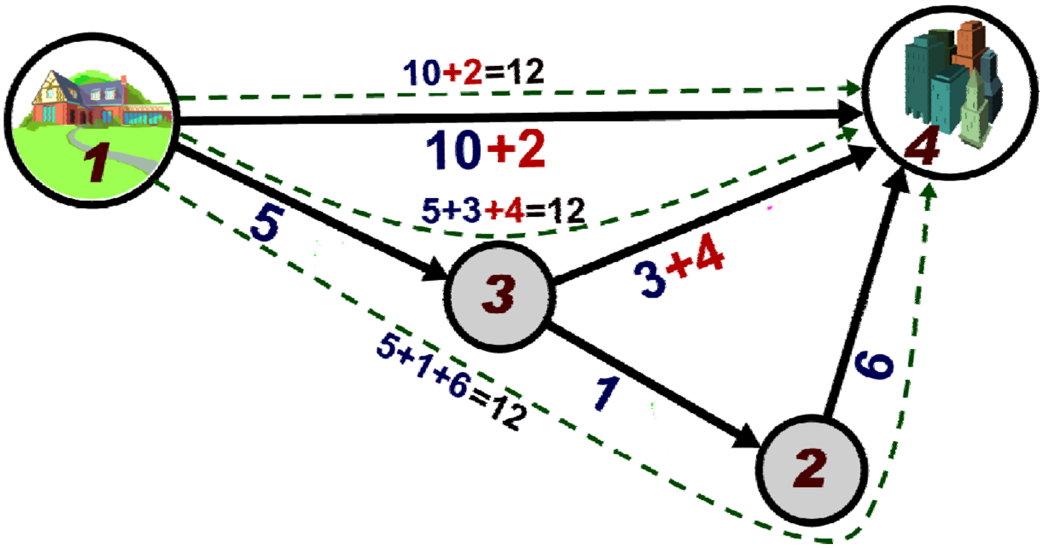

Figure 5: Roads connecting residential area at intersection 1 to business district at intersection 4

You must determine which roads should have tolls and how much each toll should be so that every route from start to finish has the same cost (driving time cost + possible toll) and no route contains more than one toll road. Additionally, the tolls should be chosen so as to minimize the final cost. In some settings, it might be impossible to impose tolls that satisfy the above conditions.

Input

Input consists of several test cases. A test case starts with a line containing an integer N ($2 \le N \le 50000$), which is the number of road intersections, and R ($1 \le R \le 50000$), which is the number of roads. Each of the next R lines contains three integers x_i, y_i, and c_i ($1 \le x_i, y_\text{i} \le N$, $1 \le c_i \le 1000$), indicating that morning traffic takes road i from intersection x_i to intersection y_i with a base driving time cost of c_i. Intersection 1 is the starting residential area, and intersection *N* is the goal business district. Roads are numbered from 1 to *R* in the given input order. Every intersection is part of a route from 1 to *N*, and there are no cycles.

The last test case is followed by a line containing two zeros.

Output

For each test case, print one line containing the case number (starting with 1), the number of roads to toll (*T*), and the final cost of every route. On the next *T* lines, print the road number *i* and the positive cost of the toll to apply to that road. If there are multiple minimal cost solutions, any will do. If there are none, print `No solution`. Follow the format of the sample output.

Sample Input

```
4 5
1 3 5
3 2 1
2 4 6
1 4 10
3 4 3
3 4
1 2 1
1 2 2
2 3 1
2 3 2
0 0
```

Output for Sample Input

```
Case 1: 2 12
4 2
5 4
Case 2: No solution
```