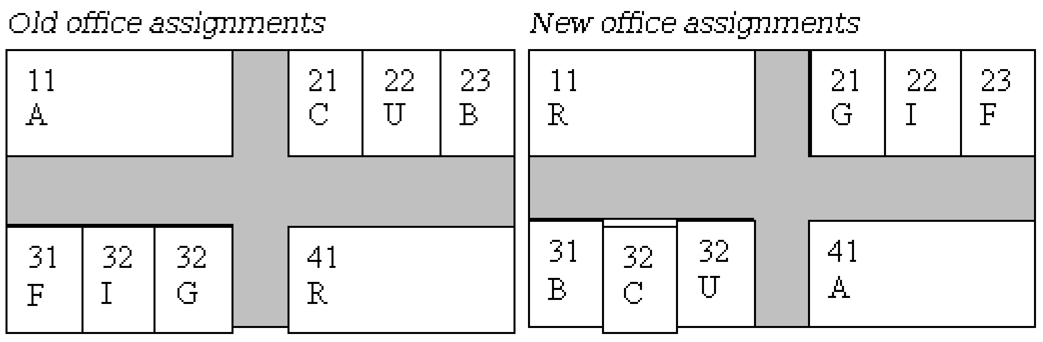
2265 Shuffle off to...Milledgeville

MTU, Inc., is a multi-national software developer with office buildings in several cities. As part of a corporate reorganization, the company's leaders are changing employee office assignments. Employees will remain in the same office building. Consider the Milledgeville office with employees A, B, C, F, G, I, R, and U as depicted below (the hallways are shaded, and the office numbers are indicated above the employee names).



Input

The input file will contain a series of building specifications. Each building specification begins with a line containing a single nonnegative integer, $n \leq 50$. The next line contains a string of between 1 and 25 uppercase or lowercase letters, indicating the name of the city where the building is located. Each of the next n lines contains a single office specification. Each office specification is of the form:

$office\_number$ $old\_occupant$ $new\_occupant$

where $office\_number$ is an integer, $old\_occupant$ and $new\_occupant$ are strings of between 1 and 25 uppercase or lowercase letters each, and these three fields are separated by whitespace. For each building specification, an employee name will appear exactly once as an $old\_occupant$ and exactly once as a $new\_occupant$. No employees share an office. Your program should stop processing building specifications when it reaches a building where n is 0.

Output

Begin the output for each building with a line containing only the name of the city where the building is located. On the next line, output the length of the longest dependency chain. Output all dependency chains meeting this length on succeeding lines, one chain per line. Begin the output for a chain with the name from the chain that comes first when the names are sorted in increasing order.

The order of succeeding names in the chain should reflect the office swaps (for example, "`C I U G`" is the correct output for the Milledgeville building, '`C`' moves to the office originally occupied by '`I`', '`I`' moves to the office originally occupied by '`U`', etc.; note that "`C G U I`" is unacceptable output in this case because '`C`' does not move to the office originally occupied by '`G`'). If there are multiple dependency chains with the same length, list them in the output in increasing sorted order of the first name in each list.

Separate the output between buildings with a blank line. Follow the format illustrated in the Sample Output.

Sample Input

```
8
Milledgeville
11 A R
21 C G
22 U I
23 B F
31 F B
32 I C
33 G U
41 R A
4
IowaCity
1  Newman     Allen
2  Allen      Newman
3  Chalmers   Granger
4  Granger    Chalmers
0
```

Sample Output

```
Milledgeville
The longest dependency chain length is 4.
Chain 1:  C I U G

IowaCity
The longest dependency chain length is 2.
Chain 1:  Allen Newman
Chain 2:  Chalmers Granger
```