Problem LINKIT: Just Link It

The usual process of creating binary code involves compiling and linking. Compiling only means to translate code from a source language to a destination language. This means in our case that machine executable code is created out of our programming language. The second step in creating a machine executable program is to link this executable code to external libraries. Since linking is a separate work step, there often exists a linker in addition to the compiler. For us - as we are old fashioned Linux hackers - the linker of our choice is the GNU linker, invoked by the command 'ld'.
The pesky part of that linker are the little details:
If we want to link our program P against a library A and a library B, but library B also depends on library A, we have to take care in which order we list the libraries A and B.
Example:
"ld -lA -lB P" would link our program P correctly, because B depends on A and A is already included when B is treated.
But
"ld -lB -lA P" would fail, because at the time B is processed, it still contains unknown symbols from A, which cannot be resolved.
Circular dependencies can be resolved by repeating the entries:
"ld -lB -lA -lB P" would also work in our previous example, and so could these be resolved.
Since it is quite annoying to work this out with more than just two libraries, it is your job - given the library dependencies - to print the correct linker command line for that problem. Since circular dependencies are more complicated, we simply want an error message, if you detect a circular dependency. Print those include libraries at first which depend on no other libraries in alphabetical order, afterwards those which depend on nothing but the already printed include libraries (again in alphabetical order), and so on.

Input

The input consists of the number of test cases c, $0 < c <= 1000$. Each test case starts with a line containing the program name, the number of involved libraries l, $0 < l <= 26$, and the number of dependencies d, $0 <= d <= 1000$. You can assume that the names of the libraries correspond to the uppercase letters of the Latin alphabet, starting with "A", so the existence of three libraries means there are the libraries "A", "B" and "C". This line is followed by d lines describing one dependency, given by two letters separated by a space, where the first one denotes the library that is dependent on the second.
You can assume that the program name will not be longer than 20 characters. There won't be a reflexive dependency of the kind "A" depends on "A", and there are only dependencies between libraries given that occur in the current linking process, i.e. if there are only two libraries, no dependency with a library "C" will occur. The lines describing the dependencies are not necessarily unique, but there are never more than 1000 lines of dependencies per test case.

Output

For each test case your program should print the linker command line for the given problem, in the order as described above. Every linker command line should be written in a separate line.
If there is a circular dependency, output a single line containing "error: circular dependency!"
Sample Input

Sample Input 1

```
3
icpc_solution.o 1 0
hal_core.o 3 2
A B
C B
everyday_code.o 2 2
A B
B A
```

Sample Output 1

```
ld -lA icpc_solution.o
ld -lB -lA -lC hal_core.o
error: circular dependency!
```