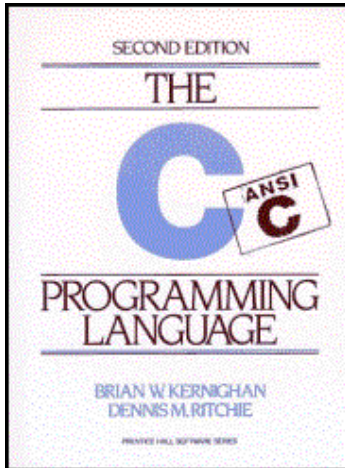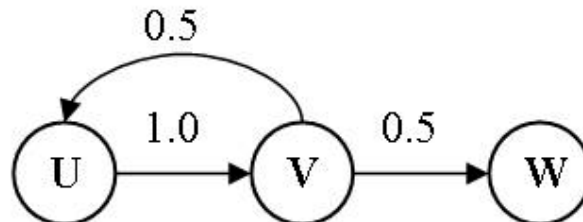# Problem I
Back to Kernighan-Ritchie
Input: Standard Input
Output: Standard Output

You must have heard the name of **Kernighan** and **Ritchie**, the authors of *The C Programming Language*. While coding in **C**, we use different control statements and loops, such as, *if-then-else*, *for*, *do-while*, etc. Consider the following fragment of pseudo code:

```
//execution starts here
do {
    U;
    V;
} while(condition);
W;
```

In the above code, there is a bias in each conditional branch. Such codes can be represented by control flow graphs like below:

Let the probability of jumping from one node of the graph to any of its adjacent nodes be equal. So, in the above code fragment, the expected number of times U executes is 2. In this problem, you will be given with such a control flow graph and find the expected number of times a node is visited starting from a specific node.

Input

Input consists of several test cases. There will be maximum 100 test cases. Each case starts with an integer: n (n ≤ 100). Here n is the number of nodes in the graph. Each node in the graph is labeled with 1 to n and execution always starts from 1. Each of the next few lines has two integers: start and end which means execution may jump from node start to node end. A value of zero for start ends this list. After this, there will be an integer q (q ≤ 100) denoting the number of queries to come. Next q lines contain a node number for which you have to evaluate the expected number of times the node is visited. The last test case has value of zero for n which should not be processed.

Output

Output for each test case should start with "Case #i:" with next q lines containing the results of the queries in the input with three decimal places. There can be situations where a node will be visited

forever (for example, an infinite `for` loop). In such cases, you should print "`infinity`" (without the quotes). See the sample output section for details of formatting.

Sample Input	Output for Sample Input
3	Case #1:
1 2	2.000
2 3	2.000
2 1	1.000
0 0	Case #2:
3	infinity
1	infinity
2	infinity
3	
3	
1 2	
2 3	
3 1	
0 0	
3	
3	
2	
1	
0	

Problem setter: Mohammad Sajjad Hossain
Special Thanks: Shahriar Manzoor