AW

Notation:

We use the following notation:

The input to the function is given in variables ("a", "b", etc).
Each Boolean gate gets two inputs and stores its output in the next consecutive letter.
We denote an AND gate by putting its inputs in alphabetical order (the first <= the second); and an OR gate by reversing the order (the first > the second).
We assume that a capital letter represents the negation of its lower case.

For example, "abBA" computes two gates, the first is AND(a,b) and the second is OR(-a,-b) when -a is the complement of a.
Another example "abAced" is an implementation of the multiplexer function:

mux(a,b,c) = b if a is true, c otherwise.

Explanation:

d <=- a AND b; it is an AND gate since ae <=- (NOT a) AND c; A means (NOT a) and AND since a
f <=- d or e; it is an OR gate since e>d

And last example, "abABdcCD" computes the xor and xnor of two bits.

Task:

You get a function as defined above and you should compute the nummber of possible inputs which would yield an output of 1 for each computed variable which is not used.
Note the we could have computed the same two outputs for the last example by "abABdcEE", but then our definition would have given only the second output (xnor) and not both.

Input:

First line is the number of tests.
Each other line has the number of inputs bits and the function in the notation described above; separated by a single space.

Output:

For each test you should output a single line which contains a list of comma separated numbers. one for each computed, but ununsed variable in their alphabetical order.

Example 1:

Input:

3
2 abBA
3 abAced
2 abABdcCD

Output:

1,3
4
2,2

Example 2:

Input:

3
4 abcedf
4 dcebfa
3 ABabaAcebc

Output:

1
15
2,0,1,2