Electric Circuit

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)
- We denote an OR gate by reversing the order (the first > the second)
- We assume that a capital letter represents the complement (inversion) of its (lower case) Boolean logic
- For example, "abBA" computes two gates: the first is AND(a,b) and the second is OR(-a,-b) where -a is the complement/inverted Boolean logic of a.
- Another example "abAced" is an implementation of the multiplexer function: mux(a,b,c) = b if a is true, c otherwise.

  Suggestion: take the time to draw the example Boolean logic circuits to ensure the notation is understood.

Task

- You receive a string logic function (similar to the above examples)
- Each gate must produce a logical output (which may or may not be used in the remaining part of the logical string)
- Each computed variable that is not further utilized in the logic string (unused Boolean logic) is output
- Your program must search for unused Boolean logic, before calculating the total amount of logical true (or 1) outputs, for all possible input combinations.

Input

- The first line is the number of tests.
- Each consecutive 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 unused variable in their alphabetical order.

<u>**Example 1:**</u>

Input:

```
3
2 abBA
3 abAced
2 abABdcCD
```

Output:

```
1,3
4
2,2
```

<u>Example 2:</u>

Input:

```
3
4 abcedf
4 dcebfa
3 ABabaAcebc
```

Output:

```
1
15
2,0,1,2
```