4745 Compressed String

Dealing with super long character strings is Chris's daily work. Unfortunately, the strings are so long that even the fastest computer in the world cannot work with them.

Chris does her work in a smart way by compressing the strings into shorter expressions. She does her compression for each string in the following way:

a) Find a consecutive repeated substring of the original string, e.g. "ab" in "cababab".

b) Replace the repeating part with the bracketed repetend, followed by the times the repetend appears in the original string. e.g. Write "cababab" as "`c[ab]3d`". Note she can also write it as "`c[ab]1ababd`" or "`ca[ba]2bd`" and so on, although these string are not compressed as well as the first one is.

c) Repeat a) and b) several times until the string is short enough.

Chris does her compression quite well. But as you know, the work is boring and a waste of time. Chris has written a computer program to help her do the boring work. Unfortunately, there is something wrong with the program; it often outputs an incorrect result. To help her debug the program, you are ordered to write a debugger which can compare Chris's standard compressed string against the string compressed by the program.

Input

There are multiple test cases.

The first line of the input contains an integer T, meaning the number of the test cases.

For each test case, there are two lines of character strings which the first one is Chris's standard compressed string and the second one is the program's compressed string. Both string contains only lowercase letters (`a-z`), square brackets (`[]`) and numbers (`0-9`). The brackets must be followed with an integer indicating the times the string in the brackets repeat, **note that the repeat time can be zero.** The brackets can be nested.

You can assume all the compressed strings in the input are no longer than 20. See further details in the input sample.

Output

For each test case, output case number first. And then if the two uncompressed strings are the same, output '`YES`' in a single line; otherwise, output '`NO`' followed by the first position where the uncompressed strings differ.

Hint:

For sample test case 3, the first string "`[a[ba]2b]12`" can be written as "`[ababab]12`", then "`[[ab]3]12`", finally we get "`[ab]36`".

The numbers in this task may be very large and cannot be stored in a 32 bit integer.

Sample Input

```
5
a[a]12
[[a]3]4a
[z]12
zzzzzzzzz
[a[ba]2b]12
[ab]36
[a]123123123[icpc]2
[[a]123]1001001inter
aismoreeasierthanc
gismuchharderthanj
```

Sample Output

```
Case #1: YES
Case #2: NO 10
Case #3: YES
Case #4: NO 123123125
Case #5: NO 1
```