Joint Password Storage

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 megabytes

Johnny is the developer of Joint Password Storage (JPS). Storing passwords in plaintext is a bad idea. JPS splits each password into several parts and stores them separately.

A password is a string consisting of digits and English letters. Bitwise XOR of all parts should be equal to the password. To attract less attention, Johnny decided that each part should look like something ordinary, like an arithmetic equality. What could be more ordinary than things like "$2 + 2 = 4$"?

Formally, a valid split of a password is a set of correct arithmetic equalities of the same length as the password, for which bitwise XOR of ASCII codes of characters on each position is equal to the ASCII code of the corresponding password character. A correct arithmetic equality is a string described by the following grammar, with both expressions evaluating to the same value. Operator precedence is standard: expressions in brackets of any type take precedence over all other operations, multiplication takes precedence over addition and subtraction, and operators with the same level of precedence are evaluated from left to right.

⟨*equality*⟩ ::= ⟨*expression*⟩ '`=`' ⟨*expression*⟩
⟨*expression*⟩ ::= ⟨*term*⟩ | ⟨*expression*⟩ '`+`' ⟨*term*⟩ | ⟨*expression*⟩ '`-`' ⟨*term*⟩
⟨*term*⟩ ::= ⟨*multiplier*⟩ | ⟨*term*⟩ '`*`' ⟨*multiplier*⟩
⟨*multiplier*⟩ ::= ⟨*number*⟩ | '`(`' ⟨*expression*⟩ '`)`' | '`[`' ⟨*expression*⟩ '`]`' | '`{`' ⟨*expression*⟩ '`}`'
⟨*number*⟩ ::= '`0`' | ('`1`' | ... | '`9`') ('`0`' | ... | '`9`')*

For your convenience, ASCII codes of all related characters are provided below:

`(`	`)`	`*`	`+`	`-`	`0-9`	`=`	`A-Z`	`[`	`]`	`a-z`	`{`	`}`
40	41	42	43	45	48-57	61	65-90	91	93	97-122	123	125

Your task is to write a splitting module which converts a password into bitwise XOR of several correct arithmetic equalities.

Input

The first line contains a single integer P ($1 \le P \le 50$) — the number of passwords to split. Each of the next P lines contains a single string — password s ($10 \le |s| \le 50$). Passwords can contain digits and both lowercase and uppercase English letters.

Output

For each password, if there is no valid split, output a line with a single word "`NO`".

Otherwise, the first line should contain a single word "`YES`". The next line should contain one integer k ($1 \le k \le 1000$) — the number of equalities in your split. Each of the next k lines should contain one equality. It can be proven that if a solution exists, there is a solution where k doesn't exceed 1000.

Example

standard input
3 1915090454 CanIAlwaysSplitIt 2020NorthwesternRussiaRegionalContestTaskJ

standard output
YES 3 9+91*9=828 1+19+0=4*5 999=1008-9 NO YES 7 420+[1*2*3*4*5*6]=140+[1*10*1*[20-5-5]*10] 10-1-{1}-{1}-{1}-{1}={1}+{1}+{1}+{{1}+{1}} 739=1+{3}*{2}*{9}*{9}-3+{2}*7*11*1+{100}+1 602211592866240={54321*67890}*9*{8*7*6*54} 51-188*1*0*600198090+5=[0]*(1039-25-4)+8*7 0*990-5127*11590*740*0*[3]*90=8*0*4885*2*8 3*0-0*818*39=0*(6+10*(64))*200*(93+6+8+19)