Problem I. Mahdi

Time limit	8000 ms
Mem limit	262144 kB
Input file	mahdi.in
Output file	standard output
OS	Windows

Time goes so quickly; remember **Bakkar**'s & **Maymona**'s anniversary two years ago when **Bakkar** bought a smartphone as a surprise gift for **Maymona**? Well **Maymona** had also a joyful surprise for **Bakkar**. **Maymona** was pregnant.

Months later **Maymona** gave birth to a beautiful baby and they decided to call him **Mahdi**. They were really so excited about their first child and decided to do their best to raise him up well. **Mahdi** is now about one year old and started to learn how to talk. As **Bakkar** & **Maymona** are geeks they had a very creative idea to teach **Mahdi** how to talk and evaluate his progress.

Simply they programmed a little teddy bear to speak so **Mahdi** can repeat after him. When **Mahdi** repeats what the teddy bear just said it recognises the word and analyse it. And as we all know its very natural that kids don't spell all letters at the beginning. So what **Bakkar** & **Maymona** thought to evaluate the correctness and progress of what **Mahdi** says is to check if the work **Mahdi** repeats after the teddy bear is a subsequence of the original word the teddy bear just said. To evaluate **Mahdi**'s progress the teddy bear defines a subsequence of a string **y** if it can be produced by deleting some of the characters in **y**. For example, "ace" is a subsequence of "adcbe" but not a subsequence of "bcae". If **Mahdi** spells a subsequence it is considered to be a positive indicator that **Mahdi** is progressing. To motivate **Mahdi** when he spells something correctly according to what we mentioned; if he spells a subsequence of the original word the teddy bear would respond with 'bravo' and start clapping, Otherwise he would repeat the word again several times until so **Mahdi** can take his time practicing.

The teddy bear is programmed in a very simple and creative way. He says random words based on an equation that **Bakkar** & **Maymona** invented. They specify a starting character to start the word, the length of the word, adder and multiplier. Simply the word is generated as follows:

s_0 = starting character

s_i = char((pos(s_{i-1}) * multiplier + i * adder) % 26)

Where:

pos(a) = 0, pos(b) = 1, ...pos(z) = 25

char(0)= a, char(1)= b,.....char(25)= z

Hmmmmm, **Bakkar** & **Maymona** are so busy; **Bakkar** at his work and **Maymona** taking care of the house and **Mahdi**. And as you are one of their best friends and still single and have some spare time you proposed to program the teddy bear for them. Besides you are the voice recognition geek among their friends ;).

For simplicity the words **Mahdi** repeats will be generated using the same equation the teddy bear uses.

Input

The first line will be an integer T ($1 \leq T \leq 10$) the number of test cases. Each test case will start by a line containing 1 character and 3 integers (length, multiplier, adder) for the string $\boldsymbol{S}$ as described above ($1 \leq \text{length}(S) \leq 10^7$). The next line will contain a single integer $\boldsymbol{N}$ ($1 \leq \boldsymbol{N} \leq 10^4$) the number of strings **Mahdi** repeats. The next $\boldsymbol{N}$ lines each contain 1 character and 3 integers (length, multiplier, adder) for the string L_i as described above. The starting character will be a lowercase character, ($1 \leq \text{length}(L_i) \leq 100$),

$1 \leq$ multiplier, adder ≤ 1000.

Output

For each test, For each test case print a single line containing "Case n:" (without the quotes) where n is the test case number (starting from 1) followed by N lines each containing either "BRAVO" (without quotes) if L_i is a subsequence of S and **Mahdi** is making a progress or "REPEAT" (without quotes) otherwise.

Sample 1

Input	Output
1 a 100 13 37 3 a 11 11 311 b 9 13 22 d 5 31 12	Case 1: REPEAT REPEAT BRAVO

Note

for input d 5 31 12, the string will be

s_0 = d

s_1 = char((pos(d) * 31 + 1 * 12)%26) = char((3 * 31 + 1 * 12)%26) = char(105%26) = char(1) = b

s_2 = char((pos(b) * 31 + 2 * 12)%26) = char((1 * 31 + 2 * 12)%26) = char(55%26) = char(3) = d

s_3 = char((pos(d) * 31 + 3 * 12)%26) = char((3 * 31 + 3 * 12)%26) = char(129%26)= char(25) = z

s_4 = char((pos(z) * 31 + 4 * 12)%26) = char((25 * 31 + 4 * 12)%26) = char(823%26) = char(17) = r

so the string is "dbdzr"

Warning: large Input/Output data, be careful with certain languages.