Nasty Calculations

Source code: `nasty.c/nasty.cpp/nasty.pas`
Time limit: 1 s
Memory limit: 64 MB

Your friend Joe forgot to write his math homework, and his teacher got angry (guess why). He ordered Joe to stay at school after the classes and evaluate a long list of expressions. The teacher is, however, quite lazy to think up so many different expressions, therefore he gave Joe a single formula f of a single variable x and a large number of values for x (as you understand, those are much easier to generate). Joe's task is to compute the value of $f(x)$ for each of the given values of x.

In fact, Joe is not required to compute the exact value of $f(x)$, since it might be quite large. Instead, he should write only the last digit of the value of $f(x)$ (Joe thinks that the reason is that the teacher can count only up to 100, but who knows...). Since the topic of the last class were numeral systems with different bases, all calculations and expressions (in particular, the expression describing f, the values for x, and the corresponding last digits of $f(x)$) are written in the B-base numeral system, where B is a given integer.

Task

As the number of values for x is really large and the formula is by no means short, Joe asked you to help. He knows that you are an excellent programmer, thus he thinks you could use the computer to solve the problem. In order to make your task simpler, he has rewritten the formula f to the postfix notation (explained below), since he is aware of the fact that it is easier for computers to evaluate such formulas. Your task is to determine the last digit of the value of $f(x)$ for the given formula f and for given values of x in the B-base numeral system.

Postfix notation

The *postfix notation* (also known as the *Reverse Polish notation*, RPN) is an alternative way of writing mathematical expressions. Using the most common infix notation, an operator is placed *between* its operands, such as $1+2$, $1+2*3$, or $(1+2)*3$. Unlike that, using the postfix notation, the operator is placed directly *after* its operands: the above expressions are encoded as $1\,2\,+$, $1\,2\,3\,*\,+$, and $1\,2\,+\,3\,*$.

This notation seems to be quite hard to read for a human eye (at least when compared with the infix notation), however, it is much easier to write a program which evaluates expressions in RPN than to write one which "understands" the infix notation. In addition, the use of RPN avoids the need of parentheses, since the sequence of operands and operators already gives a unique way of decoding the expression.

Positional base-B numeral systems

Numbers are usually presented in the *decadic numeral system*. In this system, a sequence of digits $d_k d_{k-1} \ldots d_1 d_0$ encodes the number $d_k 10^k + d_{k-1} 10^{k-1} + \cdots + d_1 10^1 + d_0 10^0$. If the number 10 is replaced by an integer B, $B \geq 2$, and d_i's are allowed to be numbers between 0 and $B-1$, we get a *base-B-system*. The value of B is the *base* of the system and d_i's are the *digits*. More precisely, in the base-B numeral system, the sequence $d_k d_{k-1} \ldots d_1 d_0$ encodes the number $d_k B^k + d_{k-1} B^{k-1} + \cdots + d_1 B^1 + d_0 B^0$. Of course, since we have only ten decadic digits, the set of "digits" must be enhanced to represent digits $d_i > 9$. The standard way is that decadic digits preserve their meaning and the remaining digits are represented by letters: A stands for 10, B for 11, etc. E.g., the number 29 in the decadic (base-10) system is written as 45 in the base-6 system, and as $1D$ in the hexadecimal (base-16) system.

Description of input

The first line of the input contains exactly two integers—B and N separated by a single space. B ($2 \leq B \leq 36$) is the base of the numeral system and N ($1 \leq N \leq 100\,000$) is the number of values for x given in the input.

The second line contains the description of the formula f in the postfix notation—it consists of several elements separated by single spaces. Those elements can be:

- A sequence of digits and upper case letters. Such a sequence describes a number written in a base-B numeral system as described earlier. You may assume that the represented number will not exceed $2\,000\,000\,000$.
- The lower case letter x. This symbol should be replaced with the appropriate value of x in every calculation.
- The addition operator $+$.
- The subtraction operator $-$.
- The multiplication operator $*$.

Each of the next N lines contains a single sequence of digits and upper case letters which encodes the corresponding value for x written in the base-B system in the same fashion as above. Again, you may assume that none of the values exceeds $2\,000\,000\,000$.

You may also expect that the input data is correct, i.e., the content of the second line is a valid expression written in the postfix notation and every sequence of digits and upper case letters is a valid integer in the base-B notation. Further, you may assume that the length of the second line will not exceed $100\,000$ characters.

Description of output

The output should contain exactly N lines. The i-th line should then contain exactly one character, either a digit or an upper case letter that is the last digit of the value of $f(x)$ (in the base-B numeral system) for x given by the $(i+2)$-th line of the input. You may assume that the value of $f(x)$ will be non-negative for all values of x given in the input.

Example

Input:

```
15 4
2 x * 123A +
1
2
3
4
```

Output:

```
C
E
1
3
```