Arithmetic expressions are usually written with the operators in between the two operands (which is called infix notation). For example, $(x + y) * (z - w)$ is an arithmetic expression in infix notation. However, it is easier to write a program to evaluate an expression if the expression is written in postfix notation (also known as reverse polish notation). In postfix notation, an operator is written behind its two operands, which may be expressions themselves. For example, '$x\ y + z\ w - *$' is a postfix notation of the arithmetic expression given above. Note that in this case parentheses are not required.

To evaluate an expression written in postfix notation, an algorithm operating on a stack can be used. A stack is a data structure which supports two operations:

1. **push**: a number is inserted at the top of the stack.
2. **pop**: the number from the top of the stack is taken out.

During the evaluation, we process the expression from left to right. If we encounter a number, we push it onto the stack. If we encounter an operator, we pop the first two numbers from the stack, apply the operator on them, and push the result back onto the stack. More specifically, the following pseudocode shows how to handle the case when we encounter an operator 'O':

```
a := pop();
b := pop();
push(b O a);
```

The result of the expression will be left as the only number on the stack.

Now imagine that we use a queue instead of the stack. A queue also has a push and pop operation, but their meaning is different:

1. **push**: a number is inserted at the end of the queue.
2. **pop**: the number from the front of the queue is taken out of the queue.

Can you rewrite the given expression such that the result of the algorithm using the queue is the same as the result of the original expression evaluated using the algorithm with the stack?

Input

The first line of the input contains a number T ($T \leq 200$). The following T lines each contain one expression in postfix notation. Arithmetic operators are represented by uppercase letters, numbers are represented by lowercase letters. You may assume that the length of each expression is less than 10000 characters.

Output

For each given expression, print the expression with the equivalent result when using the algorithm with the queue instead of the stack. To make the solution unique, you are not allowed to assume that the operators are associative or commutative.

Sample Input

```
2
xyPzwIM
abcABdefgCDEF
```

Sample Output

```
wzyxIPM
gfCecbDdAaEBF
```