Problem 9.1 **Reverse Polish Notation** (page 1 of 3)

General Statement: One day while working on your math homework, you decided to take a break and stroll through a nearby forest, calculator still in hand. While wandering through the forest, you encounter Baba Jaga, the Slavic witch of the forest and master of magic. She is angry that you have invaded her forest, and casts a spell on you. You run away and are relieved that the spell seems to have done nothing to you. However, when you start working on your math homework again, you are shocked to find that your calculator is not working properly. It is now operating in Reverse Polish Notation, and you must figure out how to use it to finish your assignment.

Reverse Polish Notation (abbreviated RPN) is a form of postfix—that is, operators appear after the operands are inputted. RPN uses the concept of a stack, in which numbers are pushed onto the stack and removed in a “last in, first out” manner. There are two modes in RPN: number input mode, and calculation mode. Here is a summary of what certain keystrokes do, in the two different modes:

Keystroke	**Number input mode**	**Calculation mode**
A digit 0 through 9 (digit *d*)	Appends the digit *d* to the end of the number in memory.	Switches to number input mode. Starts a new number in memory with the digit *d* as its only digit.
An operator `+-*/^` (operator op)	Pushes the number in memory onto the stack. Switches to calculation mode. Pops the last number off the stack into *x*, and then pops the next number off the stack into *y*. Computes *y* op x, and pushes this computed value back onto the stack.	Pops the last number off the stack into *x*, and then pops the next number off the stack into *y*. Computes *y* op x, and pushes this computed value back onto the stack.
The `ENTER` key	Switches to calculation mode. Pushes the number in memory onto the stack.	Pops the last number on the stack and pushes it onto the stack twice (duplicates the last number).

Note: Push refers to the stack operation in which an object is added to the stack. Pop refers to the stack operation in which an object from the stack is removed, according to the “last in, first out” model. That is, the last items to be pushed onto the stack will also be the first ones popped from the stack.

Problem 9.1 **Reverse Polish Notation**

Input: The first line of the input is an integer *n* that represents the number of data collections that follow where each data collection contains a list of calculator keystrokes. The first line of each data collection contains an integer *m* that represents the number of keystrokes in the data collection. The next *m* lines will each contain exactly one of the following:

- A single digit `0123456789`, representing a keystroke of that digit
- A single-character operator `+-*/^`, corresponding to the following operations in RPN respectively: addition, subtraction, multiplication, integer (truncate) division, and exponentiation
- A single all-capitalized word `ENTER`, signifying either the end of the input of the previous number (if it follows a digit) or the duplication of the last number in the stack (if it follows an operation or another `ENTER`)

Name of Data File: adv91.dat

Time Allocation: 1 second

Output: Your program should produce an indefinite number of lines of output (one or more lines for each data collection).

Output the integers in the calculator's memory stack, beginning with the lowest layer of the stack. That is, output the numbers in the reverse order of how they would be popped.

No empty lines should be outputted between the outputs for consecutive data collections.

The output is to be formatted exactly like that for the sample output given below.

Assumptions: The values for *n* and *m* each will not exceed 1000.
All numbers on the stack at any given time will be integers between -999999999 and 999999999, inclusive.
Operations will be called only when at least two numbers are on the stack.
Each data collection begins in number input mode with empty memory and no numbers on the stack.
The first keystroke of each data collection will always be a digit.
Division by 0 will not be tested, and both numbers involved in exponentiation will be positive.
All data collections will end with either an `ENTER` or an operator; no data collections will end in the number input mode.
All input will be valid.