Circular Memory

Vlad is building a computer from scratch but he doesn't really know what he's doing. He needs your help writing a program to accomplish a certain task. In this computer, memory blocks are a circle of cells with a rotating robotic arm in the middle that can grab the value at a given cell.

The arm can rotate in either direction and can do read and write operations on the cell that it is pointing at. You are asked to write a program that will load a circular memory block with the given integer values and then execute a given list of instructions on it.

Your program must keep track of an integer, *i*, initially set to zero, which will be manipulated by the given list of instructions. After all the instructions are completed, your program must print out *i*.

Instructions

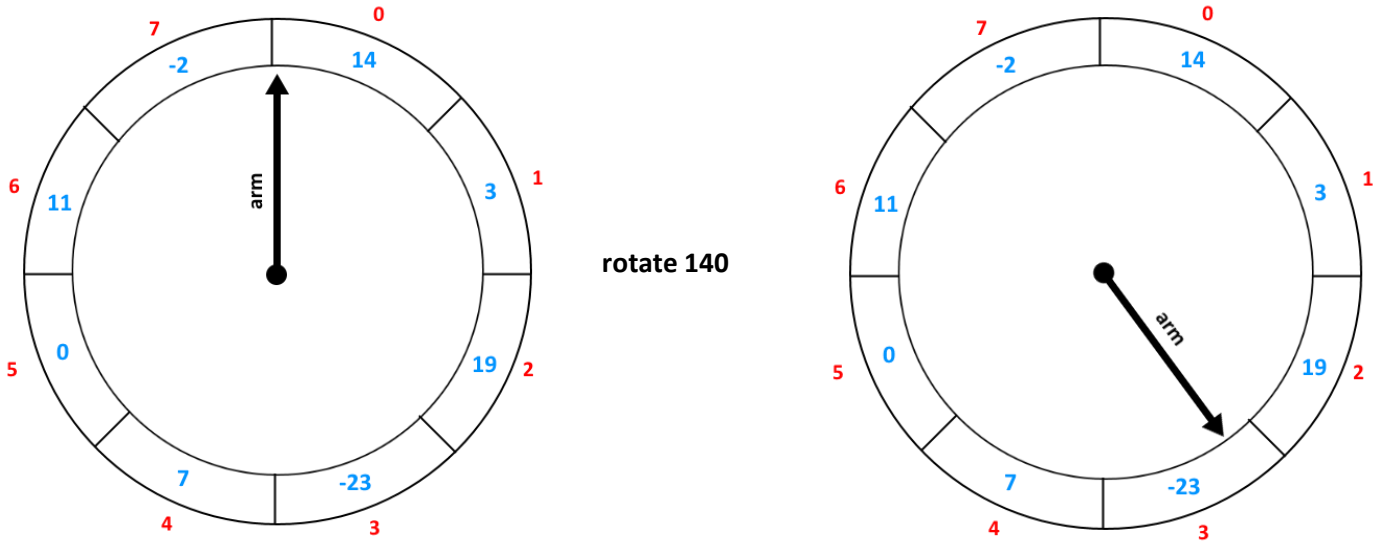
- **rotate** *degrees*
  Rotates the arm the given number of degrees, which may be either negative or positive

- **add**
  Read the inter in the memory location the arm is pointing towards and add it to *i*

- **subtract**
  Read the integer in the memory location the arm is pointing towards and subtract it from *i*

- **multiply**
  Read the integer in the memory location the arm is pointing towards and multiply *i* by it

- **store**
  Store the current value of *i* at the memory location the arm is pointing towards



red: memory cell address
blue: value in that memory cell

Notice that in the second picture, even if the arm had rotated another 5 degrees in either direction it still would have been accessing the same memory cell.

The arm starts at 0 degrees, which is the left-most boundary of cell 0. If there are *n* cells in a memory block, each cell takes up 360/*n* degrees. For example, if there were 10 cells, each cell would be 36 degrees wide and cell 0 would range from 0 to 35 degrees, cell 1 is from 36 to 71, etc…

You may assume the value of *i* will always fit within a 32-bit integer.

Input

The first line contains a single integer, *n*, where $0 < n < 36$, which indicates the numbers of cells in this memory block. The next line contains *n* integers separated by spaces. You may assume *n* will be a factor of 360.

The third line contains a single integer, *m*, where $0 < m < 100$, which is the number of instructions. The next *m* lines each contain one instruction per line. You must execute instructions in the order they appear in the input.

Output

A single line containing the value of *i* after all instructions have been executed.

Sample Input

```
9
11 -3 44 0 33 8 -89 3 7
12
add
rotate 193
subtract
store
rotate 365
multiply
rotate -13
rotate 130
multiply
multiply
rotate -112
add
```

Sample Output

```
4364
```