A certain computer has 10 registers and 1000 words of RAM. Each register or RAM location holds a 3-digit integer between 0 and 999. Instructions are encoded as 3-digit integers and stored in RAM. The encodings are as follows:

- `100` means *halt*
- `2`*dn* means *set register d to n (between 0 and 9)*
- `3`*dn* means *add n to register d*
- `4`*dn* means *multiply register d by n*
- `5`*ds* means *set register d to the value of register s*
- `6`*ds* means *add the value of register s to register d*
- `7`*ds* means *multiply register d by the value of register s*
- `8`*da* means *set register d to the value in RAM whose address is in register a*
- `9`*sa* means *set the value in RAM whose address is in register a to the value of register s*
- `0`*ds* means *goto the location in register d unless register s contains 0*

All registers initially contain 000. The initial content of the RAM is read from standard input. The first instruction to be executed is at RAM address 0. All results are reduced modulo 1000.

Input

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs.

The input to your program consists of up to 1000 3-digit unsigned integers, representing the contents of consecutive RAM locations starting at 0. Unspecified RAM locations are initialized to 000.

Output

For each test case, the output must follow the description below. The outputs of two consecutive cases will be separated by a blank line.

The output from your program is a single integer: the number of instructions executed up to and including the *halt* instruction. You may assume that the program does halt.

Sample Input

```
1

299
492
495
399
492
495
399
283
279
689
078
100
000
000
000
```

Sample Output

```
16
```