6. Magical BF (`mbf`) 60 points

John learned at school that in many East Asian languages texts can be written from left to right as well as from top to bottom. He got especially curious whether it's possible to write a text such that it can simultaneously be read both ways. John kept puzzling in programming class as well and now wants to write code such that it would solve the tasks when read by rows (top to bottom and left to right), as well as when read by columns (left to right and top to bottom). Naturally, he needed a suitable language for this and John chose a language called BF for experimenting.

The memory of a BF program is an infinite array M with cells numbered from left to right (M_0, M_1, …). Each cell contains a nonnegative integer that can be arbitrarily large. Additionally there is a data pointer that in the beginning of execution points to the leftmost cell (M_0).

The execution of a program starts from its first command and in general after the execution of each command the program moves on to the next command in the sequence. Altogether there are six commands in the language, each one denoted by one character:

Command	Meaning
`>`	Moves the data pointer to the right by one cell.
`<`	Moves the data pointer to the left by one cell if the data pointer does not point to the leftmost cell; otherwise does nothing.
`+`	Increases the cell at the data pointer by one.
`-`	Decreases the cell at the data pointer by one if the cell is currently positive; otherwise does nothing.
`[`	If the cell at the data pointer is zero, then jumps forward to the corresponding '`]`' symbol; otherwise does nothing.
`]`	If the cell at the data pointer is nonzero, then jumps back to the corresponding '`[`' symbol; otherwise does nothing.

Help John write magical BF programs for solving the five tasks listed below.

As a solution to each subtask submit an $N \times N$ grid of BF program code (where $1 \leq N \leq 1000$). The code does not have to be the same when read by rows and by columns, but must solve the task correctly in both cases. The solution text can only contain the characters '`>`', '`<`', '`+`', '`-`', '`[`' and '`]`' and must completely fill the grid. For any allowed input, the code must not execute more than 10 million commands.

Example. Task: Add the contents of the cells M_0 and M_1 and put the result into the cell M_0. The rest of the cells contain the value 0 in the beginning of the execution.

Solution:
```
>[-<+>]
[++++<>
<---->< 
+++++<<
>---]><
-<><>[>
]><<<><
```

Reading by lines: `>[-<+>][++++<><---->
<+++++<<>---]><-<><>[>]><<<><`
Reading by columns: `>[<+>-][+-+-<>-+-+-><<+-+-<<++-+]><><><>[>]><<<><`
In both cases the resulting code solves the given task.

Grading. A correct solution to each subtask is given

$$12 \cdot \frac{N_{\min}}{N}$$

points, where N is the side length of the solution's grid and $N_{\min}$ is the least side length among the working solutions of all participants ($N_{\min}$ can change during the contest and the final score of each solution is thus only determined in the end of the competition). A solution that violates restrictions in the task statement always gets 0 points.

Subtasks

John's subtasks are as follows:

1. The cells M_0 and M_1 of the array contain the integers x and y ($0 \leq y \leq x \leq 200$), all other cells contain zeros. Write the difference $x - y$ into the cell M_0.

   Input example:

7	5	0	0	...

   Output example:

2	1500	0	42	...

   In the output, it only matters that the cell M_0 contains the difference of x and y (in this case, $7 - 5 = 2$), other cells may contain any numbers (for example, 1500, 0, 42, ...).

2. The cell M_0 contains an integer x ($0 \leq x \leq 200$), all other cells contain zeros. Write the remainder of dividing x by 7 into the cell M_0. By the end of the execution of the program cells other than M_0 may contain any numbers.

   Input example:

33	0	0	0	...

   Output example:

5	0	0	0	...

3. Find the number of the leftmost cell with zero in it and write this number into the cell M_0. By the end of the execution all cells other than M_0 may contain any integers. You can assume that the answer will not exceed 100 and that at the start of the execution no cell will contain value greater than 50.

   Input example:

35	15	24	37	48	31	0	13	2	...

   Output example:

6	14	24	37	...

4. The cell M_0 contains an integer k ($3 \leq k \leq 10$) and the cells M_1 and M_2 contain the integers F_1 and F_2 ($1 \leq F_1 \leq 3$, $1 \leq F_2 \leq 3$). The elements of the sequence F_i for $i > 2$ are defined by the rule $F_i = F_{i-1} + F_{i-2}$. Find the value of F_k and write it into the cell M_0. By the end of the execution all cells other than M_0 may contain any numbers.

   Input example:

7	1	3	0	0	...

   Output example:

29	0	0	0	...

5. The cell M_0 contains a zero, one or several consecutive cells after it contain positive integers and the rest of the cells all contain zeros again. Find the greatest integer that is contained in the array at the start of the execution and write the value of this integer into the cell M_0. You can assume that the answer does not exceed 50 and there are no more than 100 cells with nonzero contents in the input array. By the end of the execution all cells other than M_0 may contain any numbers.

   Input example:

0	23	50	31	32	12	0	0	...

   Output example:

50	0	0	0	...