Mergesort is a quick sorting algorithm invented by John Von Neumann in 1945. In the heart of the algorithm lies a procedure that combines two already sorted sequences into a new sorted sequence. In this task you need to write a programme which does a similar thing.

Let A and B be sequences of **integers sorted in ascending order**, the sequence A is called *small sequence* and B *large sequence*. As it can be derived from these names, the sequence A contains less or equal number of elements than sequence B, and often the sequence A is **a lot shorter**. The sequence C is obtained by combining sequences A and B in such a way that first we sequentially take all the numbers from sequence A and then, after them, sequentially take all the numbers from sequence B. We use the notation $C[X]$ to denote the X^{th} element of sequence C where X is an integer in the interval from 1 to total number of elements in sequence C.

In the beginning, your programme knows only the lengths of sequences A and B, but not the sequence elements themselves. The programme must **sort the sequence C in ascending order** using only the following interactive commands:

- '`cmp` X Y' is a command which compares $C[X]$ and $C[Y]$ returns -1 if $C[X]$ is less than $C[Y]$, 1 if $C[X]$ is greater than $C[Y]$ and 0 if they are equal.
- '`reverse` X Y' is a command which reverses a subsequence of sequence C between the X^{th} and Y^{th} element (inclusive). For example, if sequence C is equal to $(2, 5, 2, 3)$ then the '`reverse` 2 4' command will alter it so it becomes $(2, 3, 2, 5)$.
- '`end`' is a command that denotes the end of interaction and your programme should call it when the sequence C is sorted in ascending order.

The cost of command '`reverse` X Y' is equal to the length of the reversed subsequence (in other words, $Y - X + 1$), whereas the rest of the commands do not have a cost. Write a programme that will sort the sequence C obtained by combining sorted sequences A and B as described above using **at most** 100 000 **commands in total** (including the command '`end`') and additionally, so that the total cost of all '`reverse`' commands is **at most** 3 000 000).

INTERACTION

Before interacting, your programme must read two integers from the standard input, N_A and N_B ($1 \leqslant N_A \leqslant 1\ 000$, $N_A \leqslant N_B \leqslant 1\ 000\ 000$) - lengths of sequences A and B.

Afterwards, your programme can give commands by outputting using the standard output. Each command must be output in its own line and has to be in the exact form of one of the three commands described in the task. Regarding commands '`cmp`' and '`reverse`', X and Y have to be integers less than or equal to $N_A + N_B$, and regarding the command reverse, it additionally must hold that X is less than or equal to Y.

After each given command, your program needs to flush the standard output.

After command ‘`cmp`’, your programme must input one integer using the standard input - the command result. After other commands, your program must not input anything. After outputting the command ‘`end`’, your programme needs to flush the standard output and finish executing.

Please note: Code samples that interact correctly and flush the standard output are available for download through the evaluation system.

SCORING

In test cases worth 30 points, it will hold that N_A, $N_B \leqslant 50$.

In test cases worth an additional 30 points, it will hold that N_A, $N_B \leqslant 500$.

INTERACTION EXAMPLE

Output	**Input**	**Sequence C**	**Cost**
	`2 3`	`-1 3 2 5 5`	
`cmp 1 4`	`-1`	`-1 3 2 5 5`	
`reverse 2 5`		`-1 5 5 2 3`	`4`
`cmp 2 3`	`0`	`-1 5 5 2 3`	
`reverse 4 5`		`-1 5 5 3 2`	`2`
`reverse 2 5`		`-1 2 3 5 5`	`4`
`end`			

TESTING

You can test your solution in two ways, locally or through the evaluation system. In any case, you first need to create a file that will contain test cases for testing your solution.

The first line of file must contain two integers, N_A and N_B, lengths of sequences A and B. The following line must contain exactly N_A integers separated by a single space - elements of sequence A. The following line must contain exactly N_B integers separated by a single space - elements of sequence B. Both sequences must be sorted in ascending order and their elements must fit in a 32-bit signed integer data type.

For example, an input corresponding to the interaction example above is:

```
2 3
-1 3
2 5 5
```

In order to test using the evaluation system, you must first submit the source code of your solution using the page SUBMIT and then submit the test data using the page TEST.
Local testing (Unix only) is done using `nizovi_test` file which can be downloaded through the evaluation system. You need to write the following command:

```
./nizovi_test ./your_solution input_file
```

Whatever method of testing selected, the output will provide you with information whether your programme solved your test case correctly and information about the commands your programme has given and the answers your programme obtained. In case of local testing, these additional information will be output in the file `nizovi.log` in the current directory.