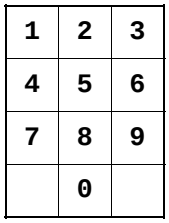
task	alarm	kaskade	tunel	toplista
source file	alarm.pas alarm.c alarm.cpp	kaskade.pas kaskade.c kaskade.cpp	tunel.pas tunel.c tunel.cpp	toplista.pas toplista.c toplista.cpp
input data	stdin			
output data	stdout			
time limit (Athlon MP 2x2.1 GHz)	1 sec			
memory limit (heap)	32 MB			
memory limit (stack)	8 MB			
points	30	50	50	70
	200			

When we set the alarm on our mobile phone, we use the keyboard to enter four digits – the hours and the minutes. For example, if the desired time is 12:30, we press the digits 1, 2, 3 and 0. One-digit numbers are entered with the leading zero (for example, the number 9 is entered by punching in 09).

Some time ago, our friend Mirko entered the wrong time and noticed that the hours displayed on the screen are actually the remainder of the hours entered divided by 24. Similarly, minutes displayed on the screen are the remainder of the minutes entered divided by 60. For example, if he punches in 66:79, the displayed time will be 18:19.

Mirko is very lazy so he wants to use as little effort as possible to obtain the required time. Write a program that finds what time should be entered so that the correct time is displayed, and **the minimum possible total effort is used**.



The effort needed for the finger to move from the key **a** to the key **b** is:

$$\textbf{effort(a,b)} = |x_a - x_b| + |y_a - y_b|$$

where (x_a, y_a) and (x_b, y_b) are the row-column coordinates of the keys **a** and **b** in the keyboard layout given in the above figure. The **total effort** is defined as the sum of three values: effort to move from the first to the second digit, effort to move from the second digit to the third digit, and effort to move from the third digit to the forth digit.

For example, total effort for to enter 22:45 is effort(2,2)+effort(2,4)+effort(4,5)=0+2+1=3.

Note: If there are multiple solutions, output **the earliest time**.

input data

The first and only line of input contains the desired time in the format HH:MM. One-digit numbers will be written with the leading zero.

The desired time is valid and between 00:00 i 23:59, inclusive.

output data

The first and only line of output should contain the correct solution in the format HH:MM. One-digit numbers should be written with the leading zero.

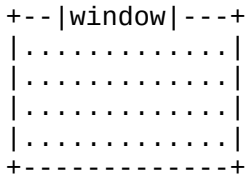
examples

input	**input**	**input**
14:19	00:11	12:34
output	**output**	**output**
14:79	24:11	12:34

A number of windows are displayed on the text-mode screen.

One window consists of the border represented by the characters '-' (minus), '|' (vertical line) and '+' (plus), of the interior represented by the characters '.' (dot), and the title of the window that is located in the middle of the upper border. The title is centered or a little to the left if exact centering is impossible. More precisely, the distance between the first letter of the title and left border will be equal to or one less than the distance between the last letter and the right border.

Each window is wide enough to contain the title, together with the adjacent strings '-|' on the left, and '|-' on the right (more precisely, the width of the window is at least 6 columns greater than the length of the title). The height of each window is at least 3 rows, and no two different windows have the same title.



We are given the layout of a screen with a number of windows on it such that no two windows overlap. We have to arrange the windows in the so called "cascade mode" defined as follows:

- the height and width of the windows must not be changed
- the upper left corner of the first window has to be in the **upper left** corner of the screen
- each next window should overlap with the previous one and its upper left corner has to be shifted exactly **one row down** and **one column right**
- windows should be **alphabetically sorted** by title – the first window in the first row, the second window in the second row etc.

input data

The first line of input contains two integers M and N, $10 \leq M,N \leq 100$, the numbers of rows and columns of the screen.

Each of the following M lines contains N characters representing the layout on the screen.

The length of the title of each window will be at least 1 and at most 10 characters. The only characters that are allowed are lowercase letters of the English alphabet ('a'-'z'). The numbers M and N will be large enough that all of the windows, after the rearranging, completely fit onto the screen.

output data

The output should contain M lines with N characters each – the layout of the screen after the windows are rearranged.

examples

input

```
12 12
............
.+-|ana|-+..
.|.......|..
.|.......|..
.|.......|..
.+-------+..
..+-|jozo|-+
..|........|
..|........|
..|........|
..|........|
..+--------+
```

output

```
+-|ana|-+...
|+-|jozo|-+.
||........|.
||........|.
+|........|.
.|........|.
.+--------+.
............
............
............
............
............
```

input

```
15 22
................+-|a|-+
.+-|winamp|--+.|.....|
.|...........|.|.....|
.|...........|.|.....|
.|...........|.|.....|
.|...........|.|.....|
.+-----------+.|.....|
...............|.....|
+----|kit|----+|.....|
|............||.....|
|............||.....|
|............||.....|
|............|+-----+
|............|.......
+------------+.......
```

output

```
+-|a|-+..............
|+----|kit|----+......
||+-|winamp|--+|......
|||...........||......
|||...........||......
|||...........||......
|||...........||......
|++-----------++......
|.....|...............
|.....|...............
|.....|...............
|.....|...............
+-----+...............
......................
......................
```

input

```
14 15
+---|a|---+....
|.........|....
|.........|....
+---------+....
...............
...............
+--|ab|---+....
|.........|....
|.........|....
+---------+....
+--|abc|--+....
|.........|....
|.........|....
+---------+....
```

output

```
+---|a|---+....
|+--|ab|---+...
||+--|abc|--+..
+||.........|..
.+|.........|..
..+---------+..
...............
...............
...............
...............
...............
...............
...............
...............
```

A tunnel is drawn in the rectangular coordinate system.

The ceiling of the tunnel starts in (0,1) and ends in (N,1).

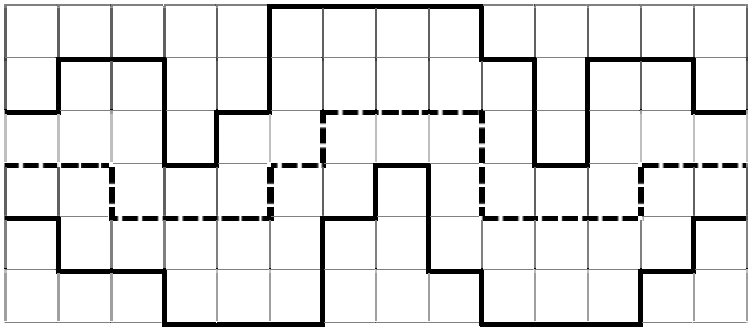
The floor of the tunnel starts in (0,-1) and ends in (N,-1).

The path through the tunnel starts in (0,0) and ends in (N,0).

The ceiling, floor and path are sequences of horizontal and vertical segments such that each corner has integer coordinates, and the x-coordinate of each corner is greater than or equal to the x-coordinate of the previous corner.

All of the y-coordinates of the ceiling and the floor will be integers between -1000 and 1000, inclusive. The path must not touch the ceiling or the floor of the tunnel, even on corners.

The tunnel from the third example is given in the figure below. The ceiling and the floor are represented with solid lines and the path is represented with the dashed line.



Write a program that finds some **shortest path** through the tunnel.

input data

The first line of input contains an integer N, $1 \leq N \leq 100{,}000$, the width of the tunnel.

The second line contains N integers – y-coordinates of the horizontal segments of the ceiling, from left to right.

The third line contains N integers – y-coordinates of the horizontal segments of the floor, from left to right.

output data

The first and only line of output should contain N integers – y-coordinates of the horizontal segments of the path, from left to right.

Note: the test data will be such that a solution, although not necessarily unique, always exists.

examples

input	**input**	**input**
9 1 4 4 4 4 4 4 4 1 -1 -1 -1 -1 2 -1 -1 -1 -1	9 1 1 1 0 -1 0 1 1 1 -1 -4 -4 -4 -4 -4 -4 -4 -1	14 1 2 2 0 1 3 3 3 3 2 0 2 2 1 -1 -2 -2 -3 -3 -3 -1 0 -2 -3 -3 -3 -2 -1
output	**output**	**output**
0 0 0 3 3 3 0 0 0	0 0 -1 -2 -2 -2 -1 0 0	0 0 -1 -1 -1 0 1 1 1 -1 -1 -1 0 0

At the end of the year, a popular radio station publishes a list of songs, ranked by listeners' votes throughout the year.

The station keeps the list confidential for a while, and organizes a guessing competition for the listeners. They announce certain hints about the placement of some songs and ask the listeners to deduce the exact positions of as many songs as possible.

For example, consider the following two statements:

- The song "Ti Da Bu Di Bu Da" is one of the top three songs.
- Songs "Treba mi nešto jače od sna" and "Ja se konja bojim" are among the top two songs.

They don't reveal anything directly, but one can still deduce that the song "Ti Da Bu Di Bu Da" comes in third on the list.

Write a program that, given a number of statements, outputs **all** songs whose **exact** position on the list can be deduced.

input data

The first line of input contains an integer N, $1 \leq N \leq 500$, the number of statements.

Each of the following N lines contains a statement of the form "A od B song1 song2 ... songA", $1 \leq A \leq B \leq 100$, meaning that the songs "song1", ..., "songA" are among the top B songs on the list.

Each song name is a string, consisting of at most 20 lowercase letters of English alphabet ('a'-'z'). The total number of different songs across all statements will be at most 500.

Note: the statements will not contradict each other and there will be at least one song whose exact position can be deduced.

output data

Output all songs whose position on the list can be deduced. The result should be printed in the form "position song", sorted in ascending order by position, each song on its own line.

examples

input

```
2
1 od 3 tidabu
2 od 2 trebami jasekonja
```

output

```
3 tidabu
```

input

```
3
2 od 2 pjesma1 pjesma2
3 od 4 pjesma1 pjesma3 pjesma4
1 od 3 pjesma4
```

output

```
3 pjesma4
4 pjesma3
```

input

```
4
1 od 4 jedan
2 od 3 dva tri
1 od 1 cetiri
1 od 4 dva
```

output

```
1 cetiri
4 jedan
```