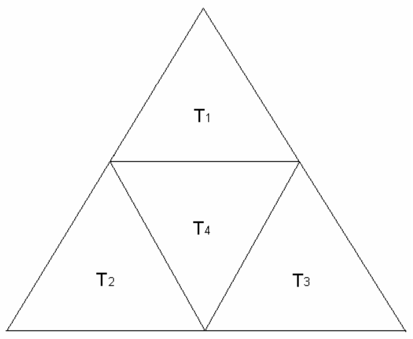
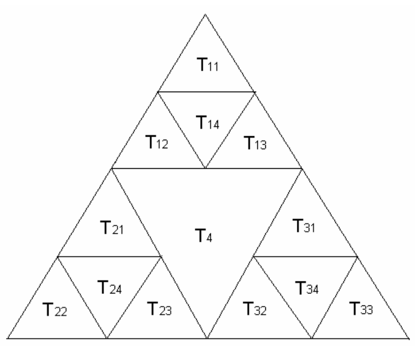
problem	waclaw	toplist	lights	matches
source file	`waclaw.pas` `waclaw.c` `waclaw.cpp`	`toplist.pas` `toplist.c` `toplist.cpp`	`lights.pas` `lights.c` `lights.cpp`	`matches.pas` `matches.c` `matches.cpp`
input data	stdin			
output data	stdout			
time limit (Athlon MP 2x2.1 GHz)	1 sec			
memory limit (heap)	32 MB			
memory limit (stack)	8 MB			
points	**50**	**60**	**70**	**70**
	250			

waclaw

Waclaw Sierpinski was a Polish mathematician who liked playing with triangles. One day he started drawing triangles using the following procedure:

- Draw an equilateral triangle T.
- Connect the midpoints of its sides with line segments. Denote the new equilateral triangles with T1, T2, T3 and T4, as illustrated in the first figure below.
- Repeat the previous step on triangles T1, T2 and T3. New triangles are: T11, T12, T13, T14, T21, T22, T23, T24, T31, T32, T33, T34.
- Continue the procedure on all triangles ending in 1, 2 or 3. The resulting fractal is called the Sierpinski triangle.





We say that a triangle A is **leaning** on the triangle B if B **does not contain** A and if one **entire** side of A **is a part** of some side of B. For example, the triangle T23 is leaning on T24 and T4, but not on T2 or T32. Note that A leaning on B does not imply that B is leaning on A.

Given a triangle A, which is a part of the Sierpinski triangle, write a program that finds **all** triangles B such that A is leaning on B.

input data

The first and only line of input contains a sequence of characters representing the given triangle, as described above. The sequence will contain between 2 and 50 characters, inclusive.

output data

Output all triangles that the given triangle is leaning on, each on a separate line, in any order.

examples

input

```
T4
```

output

```
T1
T2
T3
```

input

```
T11
```

output

```
T14
```

input

```
T312
```

output

```
T4
T314
T34
```

toplist

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 ≤ N ≤ 500, the number of statements.

Each of the following N lines contains a statement of the form "A of B song1 song2 ... songA", 1 ≤ A ≤ B ≤ 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 ('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 of 3 tidabu
2 of 2 trebami jasekonja
```

output

```
3 tidabu
```

input

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

output

```
3 pjesma4
4 pjesma3
```

input

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

output

```
1 cetiri
4 jedan
```

lights

2*N light bulbs are arranged in two rows and N columns. Each light bulb can be either off or on, and all lights are initially off.

We want to turn some of them on so that they form a beautiful pattern. In one step we can **change the state** of a sequence of (one or more) **consecutive** light bulbs in the same row or column.

Given the desired pattern, write a program that finds the **minimal** number of steps required to form the pattern.

The following figure illustrates the seven steps needed to obtain the pattern given in the third example:

0 00000000000000000000 00000000000000000000	**1** **111**00000000000000000 00000000000000000000	**2** 111000**1**0000000000000 000000**1**0000000000000	**3** 11100010000000000000 0**111110110**0000000000
4 1110110**1111**000000000 01111101100000000000	**5** 11101101111000**11111**0 01111101100000000000	**6** 111011011110001**0**1110 011111011000000**1**0000	**7** 11101101111000101**0**10 01111101100000010**1**00

input data

The first line of input contains an integer N, 1 ≤ N ≤ 10,000, the number of columns.

Each of the following two lines contains a sequence of N characters representing the desired final pattern.

Character '1' indicates a light bulb that should be on in the final state, while the character '0' indicates a light bulb that should be off.

output data

Output the minimal number of steps required.

examples

input

```
3
100
000
```

output

```
1
```

input

```
5
11011
11011
```

output

```
3
```

input

```
20
11101101111000101010
01111101100000010100
```

output

```
7
```

matches

24 matches are arranged to form a 3x3 grid as illustrated in the figure below.

```
+--+--+--+
|..|..|..|
|..|..|..|
+--+--+--+
|..|..|..|
|..|..|..|
+--+--+--+
|..|..|..|
|..|..|..|
+--+--+--+
```

Two consecutive '-' (minus) characters represent a horizontal match, whereas two consecutive '|' (vertical bar) characters represent a vertical match. '+' (plus) characters represent positions where two or more matches' ends touch. The interior of the board is filled with '.' (dot) characters.

Write a program that, given integers N and K, **removes exactly N matches** in such a way that the remaining matches form **exactly K squares** and that **each remaining match is part of some square**.

input data

The first and only line of input contains two integers N and K, 1 ≤ N < 24, 1 ≤ K < 14, the number of matches to be removed and the number of squares wanted.

output data

Output the resulting board in a format identical to the figure from the problem description, with dot characters in place of the removed matches.

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

examples

input

```
20 1
```

output

```
+--+..+..+
|..|......
|..|......
+--+..+..+
..........
..........
+..+..+..+
..........
..........
+..+..+..+
```

input

```
5 4
```

output

```
+--+--+--+
|..|..|..|
|..|..|..|
+--+--+..+
|.....|..|
|.....|..|
+--+--+..+
|........|
|........|
+--+--+--+
```

input

```
4 6
```

output

```
+--+--+--+
|..|..|..|
|..|..|..|
+--+--+..+
|..|..|..|
|..|..|..|
+--+--+..+
|........|
|........|
+--+--+--+
```