Round 2

This section consists of 4 problems.
You have 1 hour and 45 minutes to work on these problems.
Your output must EXACTLY match the output given in the samples.

Decoding Messages

You are a member of the Resistance fighting the International Evil Empire of Evilness. You have intercepted a number of transmissions from the IEEE. Though you have been able to translate much of the text, they are using a clever cipher to disguise their numbers. Unless you are able to decode these numbers, there is no way to properly sabotage the Empire's plans!

Description

Each Imperial Cipher consists of 5 integers, ranging from 1 to 6. You must convert each group of numbers into the value it represents, using one of the following rules:

Number Pattern	Example	Value
Five of one number	4 4 4 4 4	50
Four of one number	3 1 3 3 3	Sum of all values
Three of one number	6 1 6 6 4	Sum of all values
Three of one number, two of another number	3 3 5 3 5	25
Five sequential values	4 5 3 1 2	40
Four sequential values	6 3 4 1 5	30
Any numbers	2 2 1 5 6	Sum of all values

Note that order does not matter for any pattern.

In the case where the set of integers would fulfill multiple of these patterns, the largest possible value is chosen.

Input will be space-separated 5 integers, ranging from 1 to 6. Output will be the value those integers represent.

Examples

```
2 4 1 3 5
40
```

```
6 6 5 5 6
28
```

Catching Spies

You are sure that 2 Imperial spies have infiltrated the Resistance (consisting of 6 members total), but you are not sure who to suspect. The only clue you have is which recent missions have been sabotaged, as well as which members of the Resistance went on each mission.

Description

Write a program that takes in details on recent sabotaged missions, then determine how many possible spy teams each member of the Resistance could be. A spy team is defined as a pair of members of the Resistance such that every mission known to fail has at least one of those members on it.

For example, assume members 1, 2, and 3 went on the first failed mission while members 4 and 5 went on the second failed mission. The valid spy teams that member 1 can be on consist of 1 and 4, and 1 and 5. 1 cannot be on a spy team with 2, 3, or 6, as that would leave the second failed mission with no spies on it.

Each mission will be input as a list of space-separated integers ranging 1 to 6. An input line of 0 will terminate input. You may assume that no more than 5 missions will be input.

Output will consist of each member of the Resistance's number, followed by how many valid spy teams they are a member of.

Examples

```
1 2 3
4 5 6
0

1:  3
2:  3
3:  3
4:  3
5:  3
6:  3
```

```
3 4
2 4 5
2 3
3 6
0

1:  0
2:  1
3:  3
4:  1
5:  1
6:  0
```

```
1 2
2 3 4
5 6
0

1:  0
2:  2
3:  0
4:  0
5:  1
6:  1
```

Protecting Secrets

The next order of business for the Resistance is to develop our own secret code for communication - after all, those are basically unbreakable, right?

Description

Our super secret code, is defined as a simple series of commands that modify a list of 8 integers. Each of the 8 integers starts with a value of 0. Initially, the "current integer" is the first integer in the list. Each of the commands are as follows:

+	Increase the value of the current integer.
-	Decrease the value of the current integer.
>	Move to the next integer.
<	Move to the previous integer.
[	If the current integer is 0, skip to after the matching].
]	If the current integer is not 0, skip back to the matching [.

Consider the list to wrap fully. That is, if a > occurs while the current integer is the final integer, the new current integer will be the first integer. Integers should be positive and range from 0 to 255, wrapping appropriately.

The input will consist of a sequence of these commands no longer than 20 in length. Output will consist of the values of the 8 integers, space-separated.

You can assume the given code will eventually terminate.

Example

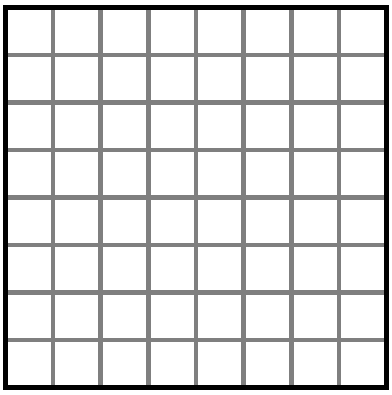
```
+++[>+>+>++>+++<<<<-]
0 3 3 6 9 0 0 0
```

Planning Operations

With our trusty zip-lines, there's no way the Evil Empire of Evil can beat us!

In order to infiltrate the EEE's facilities, our elite agents use zip-lines to travel from building to building. However, this means that each leg of our journey must be in a straight line. In order to achieve maximum stealth, we must carefully plan our route to include as few course changes as possible.

Given an 8x8 grid, write a program to find the route from the bottom left corner (0, 0) to the top right corner (7, 7) with the fewest number of course changes. Only the minimum number of course changes possible is required, not the actual path (as their may be several "best" solutions).



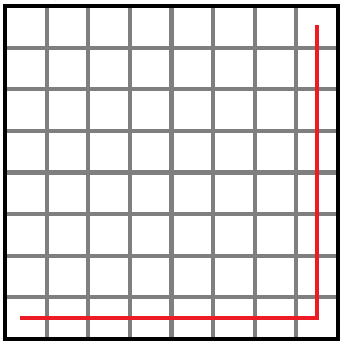
Input will be given as a series of coordinates defining tiles that are impassable. These coordinates will be given as two space-separated integers ranging from 0 to 7. You can assume that no more than 20 tiles will be impassable and that there will always be a valid route from (0, 0) to (7, 7).

Output should be the minimum number of turns required to reach (7, 7) from (0, 0), expressed as an integer.

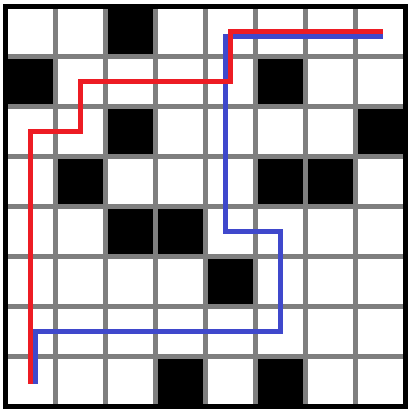
Examples

Examples below include the grid represented by the given tiles, as well as at least one valid shortest path.

```
0
1
```



```
0 3
0 5
2 4
3 2
3 3
4 1
4 5
4 6
5 2
5 7
6 0
6 5
7 2
0
5
```



The Language of Time

As you sneak into the IEEE's headquarters, you hear a strange sound — Z-BWOMPH! In front of you appears a mysterious device. A door opens up and out comes two men: one wearing a robe and slippers, the other a scowl and the skins of his enemies. "Can you help us fix our time machine?" Seymore asks.

Description

The language synthesizer on Seymore's time machine is broken. It's failing to recognize which phrases are valid expressions of time.

A proper expression of time consists of a list.

A list is defined as one of the following: (element), (element element>, or (element element element

An element is defined as one of the following: list, equation, or][

An equation is defined as one of the following: element operator element, or element operator

The valid operators are: +, -, *, /, } #, %, &

All characters not mentioned above are invalid. Spaces will not occur in any sequences.

The input will be a sequence of characters. Output will be true if the sequence is a valid expression of time, or false if it is not.

Examples

```
(][][][{
true
```

```
((][][+(][)][{})
true
```

```
((][][+(][)][{}>
false
```