Albert's Dream

Albert has a dream: he wants to create his own dictionary (Merriam Webster is his idol and he hopes to have his own name on a thick book one day). Albert realizes that he does not know enough words to create a complete dictionary. But he has many many books and if he could only get all of the unique words out of those, he would be happy.

You are Albert's best friend and majoring in computer science, so you offer to help with processing all the texts and selecting only the unique words. You are going to write a program that processes text and produces a list of unique words in alphabetical order. Before you do that, Albert gives you his definition of a word. A *word* is a sequence of one or more consecutive alphabetic characters in upper or lower case. Albert also wants your program to be case insensitive, that is words like "APPLE", "apple", "appLE" should be treated as the same.

Input

The input is a text with up to 5,000 lines. Each line has at most 200 characters. The input is terminated by EOF.

Output

A list of different/unique words that appear in the input text, one per line. The output should be in alphabetical order and in lower case. You are guaranteed that the number of unique words in the text is no more than 5,000.

Example 1

```
Input:
Albert has a dream: he wants to create his own dictionary (Merriam Webster is his idol and he hopes
to have his own name on a thick book one day).

Output:
a
albert
and
book
create
day
dictionary
dream
has
have
he
his
hopes
idol
is
merriam
name
on
one
own
thick
to
wants
webster
```

Note: the above input is all on one line. Wrapping done simply to make it visible.

Example 2

```
Input:
121()* &* abc *awre@#

Output:
abc
awre
```

Reverse Polish notation

Figure 1: Jan Łukasiewicz was a Polish logician and philosopher best known for Polish notation. Source: Wikimedia

Reverse Polish notation (RPN), also known as Polish postfix notation or simply postfix notation, is a mathematical notation in which operators follow their operands, in contrast to Polish notation (PN), in which operators precede their operands. It does not need any parentheses as long as each operator has a fixed number of operands.

You need to write a program that transforms an infix expression to a equivalent RPN according to the following specifications.

1. The infix expression is in the input file in the format of one character per line, with a maximum of 50 lines. For example, `(1+1)*(4*5+1)-4` would be in the form:

```
(
1
+
1
)
*
(
4
*
5
+
1
)
-
4
```

2. There will be only one infix expression in the input file, and it will be an expression with a valid syntax.
3. All operators are binary operators `+, -, *, /`.
4. The operands will be one digit numerals: 0, 1, 2, ... , 9.
5. The operators `*` and `/` have the highest precedence. The operators `+` and `-` have the lowest precedence. Operators at the same precedence level associate from left to right. Parentheses act as grouping symbols that over-ride the operator precedence.

Input

There will be multiple lines in the input file as specified above.

Output

The output file will have a postfix expression all on one **line** with no whitespace between symbols and a single newline character at the end.

Example

```
Input:
(
1
+
1
)
*
(
4
*
5
+
1
)
-
4

Output:
11+45*1+*4-
```

Olympic Average

Some sports, such as gymnastics, use a *trimmed mean*, in which the highest and lowest scores are discarded and the remaining ones averaged. This is also known as the Olympic average, due to its use in the Olympic events, to make the score robust to a single outlier judge.

You are a game developer. Your game Quarter Life 3 is now available on Sbeam. You have released multiple updates and patches for this game to add new features, improve balance, fix bugs, etc. Since it has been very popular among game players, you received lots of reviews on Sbeam each time you release a new patch. Your boss asked you to evaluate the total feedback for each update. And you decided to use *trimmed mean* to calculate the average review score after each update.

Each time you release an update, you will receive some new reviews. Then you will find two reviews on Sbeam, one with the largest score and one with the smallest score. You will remove those two reviews from Sbeam as they are outliers. Finally you can calculate the *trimmed mean* from the remaining reviews. All the remaining reviews will be kept for next update.

Calculating mean is easy. Your task is to calculate the difference between the removed scores for each update.

Input

The first line contains an integer n, 1 <= n <= 5000, the number of released patches.

The subsequent n lines contains a sequence of non-negative numbers separated by whitespaces. The numbers in the (i+1)-st line give the reviews scores for the i-th update.

The first number in each of these lines, k, 0 <= k <= 100,000 , is the number of reviews and the subsequent k numbers are positive integers of the review scores.

The review scores on Sbeam are integers ranged from 1 to 1,000,000, inclusive. The total number of all review scores is no bigger than 1,000,000.

And there will be at least 2 reviews on Sbeam after receiving reviews for each update.

Output

Print n lines, the i-th line contains the difference between the largest score and the smallest score for the i-th update.

Example 1

```
Input:
2
2 1 1
2 2 5

Output:
0
3
```

Example 2

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

Output:
5
3
1
```

Example 3

```
Input:
5
2 1 1
3 2 3 1
4 5 1 5 10
0
1 2

Output:
0
2
9
3
3
```

Stack Puzzle

There are two sequences of stack operations converting the word TROT to TORT:

```
[
i i i i o o o o
i o i i o o i o
]
```

where `i` and `o` stands for *in* / push and *out* / pop operation respectively. In this problem, you are given two words and you are asked to find out all sequences of stack operations converting the first word to the second word.

Input The input consists of two lines, the first of which is the source word and the second is the target word. The length of each word does not exceed 12.

Output Your program should print a sorted list of valid `i/o` sequences. The list is delimited by

```
[
]
```

and the sequences are sorted in lexicographical order. Within each sequence, `i`'s' and `o`'s are separated by a single space and each sequence is terminated by a new line. There should be a newline character printed after the closing square bracket delimiter. Warning: there should not be any spaces after the last `o` of the output.

Process Given an input word, a valid `i/o` sequence implies that every character of the word is pushed and popped exactly once, and no attempt is ever made to pop an empty stack. For example, if the word FOO is input, then the sequence: + `i i o i o o` is valid and produces OOF + `i i o` is not valid (too short), + `i i o o o i` is not valid (illegal pop from an empty stack)

A valid sequence results in a permutation of the letters in the input word. For example, given the input word FOO, both sequences `i i o i o o` and `i i i o o o` give the word OOF.

Example 1

```
Input:
madam
adamm

Output:
[
i i i i o o o i o o
i i i i o o o o i o
i i o i o i o i o o
i i o i o i o o i o
]
```

Example 2

```
Input:
bahama
bahama

Output:
[
i o i i i o o i i o o o
i o i i i o o o i o i o
i o i o i o i i i o o o
i o i o i o i o i o i o
]
```

Example 3

```
Input:
long
```

```
short

Output:
[
]
```

Example 4

```
Input:
eric
rice

Output:
[
i i o i o i o o
]
```

Picking Flowers

Mary, Ann and Lucy decided to go to a garden to collect some flowers for their art project. They collected x, y and z number of flowers respectively. Since the garden had variety of flowers, they decided to number each type of flower with a distinct non negative integer.

Once they are done collecting flowers, Ann made 2 observations:

- Ann has picked some type of flowers that Mary and Lucy did not.
- Both Mary and Lucy had some kind of flowers that Ann did not pick.

Now, your task is to help each of them find the number of type of flowers that each of them has while the other two do not. And also the number of types of flowers that each one does not have while the other two have.

Input

The first line of input contains 3 integers: x, y, z which are the flowers picked by each of them. x, y, z <= 1000

Each of the next 3 lines contains x, y and z integers respectively, each representing the type of flowers picked by Mary, Ann and Lucy respectively.

Output

The output should consist of 3 lines each, first for Mary, second for Ann and last for Lucy. Each line should have 2 integers, separated by a single space. First integer for each line represents the number of flowers the respective person has and the second integer represents the number of flowers the respective person does not have while the other two does.

Example 1

```
Input:
2 2 2
1 2
3 4
0 0

Output:
2 0
2 0
1 0
```

Example 2

```
Input:
6 4 3
11 22 33 44 55 66
11 22 77 88
33 44 55

Output:
1 0
2 3
0 2
```