July Fourth, Take 2

Alexandra was born on fourth of July. She considers 4 and 7 to be her favorite digits. She really likes all the numbers that are made only of these two digits. For example: 4, 7, 477447, 77744774, 7474, 474, 4747474, 747, 44, 7777 - possibilities are endless. She calls these numbers *July Fourth Numbers*.

Recently she's been wondering about other interesting numbers. She is particularly interested in the numbers that can be evenly divided (i.e., without remainder) by her favorite numbers. She even came up with the name for these numbers: they are *July Fourth Family Numbers*.

Your job is to write a program that determines if a given number belongs to the *July Fourth Family Numbers*. Note that all *July Fourth Numbers* are a subset of *July Fourth Family Numbers* since each *July Fourth Number* can be divided evenly by itself.

Input

Input is a single line containing a number 1 <= n < 2^64.

Output

A single line containing the string "July Fourth Family Number" or "Not in the family" - depending on what your program decides about the given input value.

Example 1

```
Input:
74

Output:
July Fourth Family Number
```

Example 2

```
Input:
8

Output:
July Fourth Family Number
```

Because 8 can be evenly divided by 4.

Example 3

```
Input:
47471

Output:
Not in the family
```

Daily Prize

Trader Jane's has a new promotion to attract even more customers to already crowded supermarket.

To participate in a daily drawing of the prize a customer needs to fill out a form with their name, phone number and the amount that they paid for their groceries.

At the end of the day, Trader Jane's manager picks two forms from among the completed ones: - first is the one that has the highest amount paid - second is the one that has the lowest amount paid The person who paid the highest amount gets the prize equal to the difference between their bill and the lowest bill.

Given how busy Trader Jane's supermarket is, you can be certain that at the end of each day there are at least two bills in the pool to select from (usually there are many many more).

The selected forms are discarded, but the other ones remain in the pool for the next day, so each customer has a chance to be selected as the prize winner on the day of their purchase, or any day after.

Your task is to compute how much money Trader Jane's pays out in prizes.

Input

The input contains an integer `n`, `1 <= n <= 5,000` on the first line - this is the number of days in the promotion. Each of the next `n` lines contains a sequence of non-negative integers separated by whitespace. The first number on each line indicates the number of forms submitted on that day, `0 <= k <= 100,000`. The next `k` numbers specify the bill amounts on the new forms entered for the daily drawing on that day. Each amount is guaranteed to be no larger than `1,000,000`. The total amount of all bills is no larger than `1,000,000`.

Output

Print one number that is the sum of all the prizes that Trader Jane's pays out during the promotion followed by a newline.

Example 1

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

Output:
19
```

Example 2

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

Output:
2
```

Buggy Keyboard

Gabrielle has an assignment for her Algorithmic Problem Solving course due in a couple of hours. She opened up her computer which she purchased recently and started working using her favorite editor when she discovered something strange. Generally, pressing the backspace key should erase a character to the left of the cursor. But on her new computer, pressing that key produced a character `<`. Since the assignment is due in couple of hours, she does not have time to call customer support or replace the computer so she decides to temporarily find a way around the problem with the help of a program.

Help Gabrielle write a program which takes the string written on her computer and outputs the string that Gabrielle actually intended to write. It can be assumed that she never needs to output the character `<`. Also you can be assured that she will never press backspace on an empty line.

Figure 1: Computer keyboard

Input

A string `s` containing text written on Gabrielle's computer. The length of `s` is less than 10^6 The string will contain only lower case letters, spaces, and the character `<`.

Output

A string containing text that was actually intended. Note that there should be no spaces or new line after the last character in the output

Example 1

```
Input:
a<bcd<

Output:
bc
```

Example 2

```
Input:
prind<tf

Output:
printf
```

Example 3

```
Input:
as<d<<

Output:
```

Note: there is no output in the last example.

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
```

Parsing Brackets

Sam is working on a new program that is supposed to parse and validate the use of brackets in code programs. He came up with a way to do this for one type of bracket but run into problems when there are multiple types of brackets in the program to be validated. He asks you for help in completing the more advanced program.

A valid sequence of brackets has them correctly nested and sequenced. Here are some general rules:

- valid bracket pairs are `()`, `[]`, and `{}`
- any sequence of characters that does not include any brackets is a valid program (well, a least from the point of view of validating the brackets)
- all programs below are valid:
  - `(VALID)`
  - `[VALID]`
  - `{VALID}`
  - `VALID1VALID2`

  in which `VALID`, `VALID1` and `VALID2` are any valid programs

By these rules all the one-line programs below are valid:

```
a+b
```

```
()
```

```
[a+b]
```

```
if (a + b = c) {x = 50 * y;}
```

```
if (a + b = c) { if (x < y) x = 50 * z[7];}
```

The following programs are not valid:

```
a+b)
```

```
][
```

```
if (a + b = c) { if (x < y) x = 50 * z[7];
```

```
if (a + b = c) { if (x < y} x = 50 * z[7];}
```

Input

For simplicity, the entire input is given on a single line. It may consist of any sequence of characters. Each program is at most 3000 characters.

Output

The program should print `YES` if the program is valid (from the point of view of matching brackets), or `NO` followed by the character position (1-based) at which the first problem was detected if the program is not valid.

Example 1

```
Input:
if( x < y) {print x;}

Output:
YES
```

Example 2

```
Input:
if( x < y) {print x;

Output:
NO 21
```

Example 3

```
Input:
if (a + b = c) { if (x < y} x = 50 * z[7];}

Output:
NO 27
```