Problem A: Maximal number

You are given an integer number N. You can swap any two digits of number N. For example. N = 89547 You can swap first and third digits, and you will get number 59847.
You can do this operation as many times as you whant, find the maximal number you can get from given number N.

Input

The first and only line of input contains one integer number N ($1 \le N \le 30000$).

Output

Output the maximal number you can get.

Sample Input	Sample Output
5984	9854
8702	8720

Problem B: Matrix

You are given matrix A_{NxN} , each cell contains either 0 or 1.
Matrix is called antisymetric if for *any i and j* $1 \le i,j \le N$ *if* $A_{i,j} = 1$ *and* $A_{j,i} = 1$ *then* $i=j$. In other words there can't be two ones in symetric cells.
Example of antysimetric matrixs.

0 0 1 0 0 1
1 0 1 1 0 1

Example of not antysimetric matrixs.

0 1 1 1 0 1
1 0 1 0 1 1

You are given N, find number of NxN antysimetric matrixs.

Input
Input contains one integer number N, ($1 \le N \le 9$).

Output
Output number of antysimetric matrixs.

Sample Input	Sample Output
1	2
2	12

Problem C: Mail checker

You are writing some part of one internet website. At this moment you are creating registration block. And you have to chek either users mail address is correct or not ! Mail adress must satisfy following statements:

1. Mail adress must consist only of english letters ‘a’..’z’ , ‘A’..’Z’, digits ‘0’..’9’ , ‘@’ , and dot ‘.’

2.String must contain exactly one symbol ‘@’

3.String before ‘@’ must not be empty.

4.After ‘@’ there must present at least one dot (‘.’).

5. String between two dots, between ‘@’ and a dot, and string after last dot must not be empty.

Example of correct mail addres is register.contest@gmail.com , exmaple of not correct adress is ed@k@gmail.com !

Input

Input contains one non empty string. Length of this string is at most 50. String doesn’t contain any spaces.

Output

Output “Correct” if mail address is correct, and “Invalid” otherwise.

Sample Input	Sample Output
register.contest@gmail.com	Correct
ed@k@gmail.com	Invalid

Problem D: Ambiguous permutations

Some programming contest problems are really tricky: not only do they require a different output format from what you might have expected, but also the sample output does not show the difference. For an example, let us look at permutations.

A **permutation** of the integers *1* to *n* is an ordering of these integers. So the natural way to represent a permutation is to list the integers in this order. With $n = 5$, a permutation might look like 2, 3, 4, 5, 1.

However, there is another possibility of representing a permutation: You create a list of numbers where the *i*-th number is the position of the integer *i* in the permutation. Let us call this second possibility an **inverse permutation**. The inverse permutation for the sequence above is 5, 1, 2, 3, 4.

An **ambiguous permutation** is a permutation which cannot be distinguished from its inverse permutation. The permutation 1, 4, 3, 2 for example is ambiguous, because its inverse permutation is the same. To get rid of such annoying sample test cases, you have to write a program which detects if a given permutation is ambiguous or not.

Input

The first line of each test case contains an integer n ($1 \le n \le 100000$). Then a permutation of the integers *1* to *n* follows in the next line. There is exactly one space character between consecutive integers. You can assume that every integer between *1* and *n* appears exactly once in the permutation.

Output

For each test case output whether the permutation is ambiguous or not. Adhere to the format shown in the sample output.

Sample Input	Sample Output
1 1	ambiguous
4 1 4 3 2	ambiguous
5 2 3 4 5 1	not ambiguous

Problem E: Number e

Output number **e** rounded to the N-th digit after decimal point. Number **e** rounded to the 25^{th} digit after decimal point is 2.7182818284590452353602875

Input
Input contains one integer number N ($0 \le N \le 25$).

Output
Output number **e** rounded to the N-th digit after decimal point.

Sample Input	Sample Output
0	3
25	2.7182818284590452353602875
13	2.7182818284590

Problem F: Bullshit Bingo

Bullshit Bingo is a game to make lectures, seminars or meetings less boring. Every player has a card with 5 rows and 5 columns. Each of the 25 cells contains a word (the cell in the middle has always the word "BINGO" written in it). Whenever a player hears a word which is written on his card, he can mark it. The cell in the middle is already marked when the game starts. If a player has marked all the words in a row, a column or a diagonal, he stands up and shouts "BULLSHIT". After this, the game starts over again.

Sitting in a lecture, you observe that some students in the audience are playing Bullshit Bingo. You wonder what the average number of different words is until "BULLSHIT" is exclaimed. For the purpose of this problem, a word consists of letters of the English alphabet ('a' to 'z' or 'A' to 'Z'). Words are separated by characters other than letters (for example spaces, digits or punctuation). Do the comparison of words case-insensitively, i.e., "Bingo" is the same word as "bingo". When counting the number of different words, ignore the word BULLSHIT (indicating the end of the game), and consider only the words of the current game, i.e., if a word has already occurred in a previous game, you may still count it in the current game. If the last game is unfinished, ignore the words of that game.

Input

The input file consists of the text of the lecture, with "BULLSHIT" occurring occasionally. The first game starts with the first word in the input. Each occurrence of "BULLSHIT" indicates the end of one game.
You may assume, that

- the word "BULLSHIT" occurs only in uppercase letters
- every word has at most 25 characters, and each line has at most 100 characters
- there are at most 500 different words before a game ends
- the players follow the rules, so there is no need to check if a game is valid or not

Output

The output consists of one number: the average number of different words needed to win a game. Write the number as a reduced fraction in the format shown below. Reduced fraction means that there should be no integer greater than 1 which divides both the numerator and denominator. For example if there were 10 games, and the number of different words in each game summed up to 55, print "11 / 2".

Sample Input

Programming languages can be classified BULLSHIT into following types:
- imperative and BULLSHIT procedural languages
- functional languages
- logical BULLSHIT programming languages
- object-oriented BULLSHIT languages

Sample Output

9 / 2