Problem A: Decoding ASCII

John wishes to send a confidential message to Paul. To protect the content of his message, John breaks his message into two pieces, encrypts each piece and then sends each one separately by courier to Paul (the details of the encryption process are given below). Write a program for Paul to decode messages from John.

John's messages consist of printable ASCII characters; that is, the characters with ASCII values in the range 32, 33, ..., 125, 126. The successor of a printable ASCII character is defined to be the printable ASCII character that comes immediately after it, with the exception that the successor of character 126 is 32 (i.e. we wrap around at the end). The predecessor of a printable ASCII character is defined to be the ASCII character that comes immediately before it, with the exception that the predecessor of character 32 is 126.

To encrypt his message, John first partitions the characters of the message into two rows. The first row contains all the odd characters (i.e. the first, third, fifth, ... characters), and the second row contains all the even characters (i.e. the second, fourth, sixth, ... characters). Next, each character in the first row is replaced with its successor, and each character in the second row is replaced with its predecessor. The resulting two rows are then sent to Paul.

Example of Encryption:
Suppose the message is "Hello, Paul. This is John."
row 1 is "Hlo al hsi on"
row 2 is "el,Pu.Ti sJh."
row 1 is encrypted as "Imp!bm!itj!po"
row 2 is encrypted as "dk+Ot-Sh rIg-"

Input

a.in

The first line of input contains a positive integer N specifying the number of data sets that follow. Each data set consists of two lines of printable ASCII characters. The first line contains no more than 40 characters (it is the first encrypted piece). The second line also contains no more than 40 characters (it is the second encrypted piece).

Note that each encrypted piece might contain SPACE characters since SPACE has an ASCII value of 32. Make certain that when your program reads in the two lines that it does not omit any SPACE characters.

Output

a.out

For each data set, print one line of text consisting of the decrypted message.

Sample Input

```
2
Imp!bm!itj!po
dk+Ot-Sh~rIg-
d
d
```

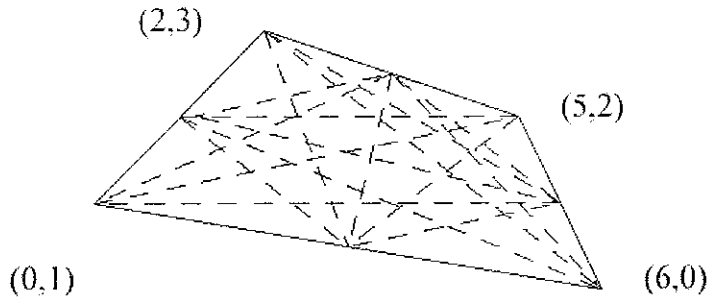
Sample Output

```
Hello, Paul. This is John.
ce
```

Problem B: This Takes the Cake

In the kingdom of Polygonia the royal family consists of the king, the queen, and the 10-year-old twins, Prince Obtuse and Prince Trisect. The twins are fiercely competitive, and on their birthday they always vie with each other for the biggest portion of the cake. The wise king and queen have devised the following way to prevent squabbles over the cake. One prince is allowed to cut the cake into two pieces, then the other prince gets to choose which of the two pieces he wants.

Cakes in Polygonia are always in the shape of a convex quadrilateral (a four-sided polygon with each internal angle less than 180 degrees). Furthermore, local custom dictates that all cake cutting must be done using a straight cut that joins two vertices, or two midpoints of the sides of the cake, or a vertex and a midpoint. For instance, the following figure shows all the possible legal cuts in a typical cake.



Your problem is to determine, for a number of different cakes, the best cut, i.e., the one that divides the cake into two pieces whose areas (we are disregarding the thickness of the cake) are as nearly equal as possible. For instance, given a cake whose vertices (when the cake is viewed from above) are located, in counterclockwise order, at the points (0, 1), (6, 0), (5, 2) and (2, 3), the best possible cut would divide the cake into two pieces, one with area 4.375, the other with area 5.125; the cut joins the points (1, 2) and (5.5, 1) (the midpoints of two of the sides).

Input b.in

Input consists of a sequence of test cases, each consisting of four (x, y) values giving the counterclockwise traversal of the cake's vertices as viewed from directly above the cake; the final test case is followed by a line containing eight zeros. No three points will be collinear, all quadrilaterals are convex, and all coordinates will have absolute values of 10000 or less.

Output b.out

For each cake, the cake number followed by the two areas, smaller first, to three decimal places of precision.

Sample Input

```
2 3 0 1 6 0 5 2
0 0 100 0 100 100 0 100
0 0 0 0 0 0 0 0
```

Sample Output

```
Cake 1: 4.375 5.125
Cake 2: 5000.000 5000.000
```

Problem C: ACM (ACronym Maker)

The sadists who design problems for ACM programming contests often like to include the abbreviation "ACM" somewhere in their problem descriptions. Thus, in years past, the World Finals has had problems involving "Apartment Construction Management," the "Atheneum of Culture and Movies," the "Association of Cover Manufacturers," "ACM Airlines," the "Association for Computational Marinelife," and even an insect named "Amelia Cheese Mite." However, the number of word combinations beginning with A, C, and M that make sense is finite and the problem writers are starting to run out of ideas (it's hard to think of problems about "Antidisestablishmentarianistic Chthonian Metalinguistics"). Fortunately, modern culture allows more flexibility in designing abbreviations - consider, for example:

GDB - Gnu DeBugger
LINUX - either "LINus's UniX" or "LINUs's miniX" or "Linux Is Not UniX"
SNOBOL - StriNg Oriented symBOlic Language
SPITBOL - SPeedy ImplemenTation of snoBOL
The rules used in these examples seem to be:

- Insignificant words (such as "of", "a", "the", etc.) are ignored.
- The letters of the abbreviation must appear, in the correct order, as an ordered sublist of the letters in the significant words of the phrase to be abbreviated.
- At least one letter of the abbreviation must come from every significant word (multiple occurrences of a letter are, of course, treated as distinct).

Of course these rules are often broken in real life. For instance, RADAR is an abbreviation for "RAdio Detecting And Ranging". Under our rules (assuming that "and" is an insignificant word), this would not be a valid abbreviation (however, RADR or RADRAN or DODGING would be valid). You have been asked to take a list of insignificant words and a list of abbreviations and phrases and to determine in how many ways each abbreviation can be formed from the corresponding phrase according to the rules above.

Input C.in

The input file consists of multiple scenarios. Each scenario begins with an integer $100 \geq n \geq 1$ followed by n insignificant words, all in lower case, one per line with no extra white space. (A line containing 0 indicates end of input.) Following this are one or more test cases for this scenario, one per line, followed by a line containing the phrase "LAST CASE". Each line containing a test case begins with an abbreviation (uppercase letters only) followed by a phrase (lowercase letters and spaces only). The abbreviation has length at least 1 and the phrase contains at least one significant word. No input line (including abbreviation, phrase, and spaces) will contain more than 150 characters. Within these limits, however, abbreviations and phrase words may be any length.

Output

C, out

For each test case, output the abbreviation followed by either `is not a valid abbreviation` or `can be formed in` i `ways` where i is the number of different ways in which the letters of the abbreviation may be assigned to the letters in the phrase according to the rules above. The value of i will not exceed the range of a 32-bit signed integer.

Sample Input

```
2
and
of
ACM academy of computer makers
RADAR radio detection and ranging
LAST CASE
2
a
an
APPLY an apple a day
LAST CASE
0
```

Sample Output

```
ACM can be formed in 2 ways
RADAR is not a valid abbreviation
APPLY can be formed in 1 ways
```

Problem E: Perfect Numbers

A positive integer is said to be a perfect number if it is equal to the sum of its positive divisors less than itself. For example, 28 is perfect, because

$$28 = 1 + 2 + 4 + 7 + 14$$

On the other hand, 12 is not perfect, because

$$12 \neq 1 + 2 + 3 + 4 + 6$$

You are to write a program that prompts the user to enter a positive integer and responds by reporting whether or not the given number is perfect. The program should terminate when the user enters 0.

Input

e.in

The input consists of a sequence of integers between 0 and $2^{31} - 1$. End of input will be indicated when the number is 0.

Output

e.out

For each nonzero input N, you should print either
N IS perfect.
or N IS NOT perfect.

Do not print anything for the 0 line.

Sample Input

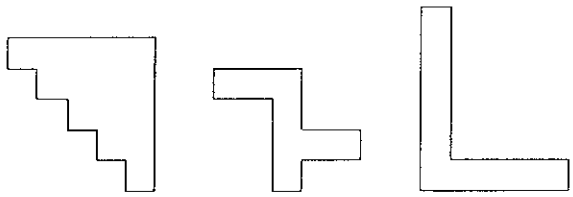
```
12
28
0
```

Sample Output

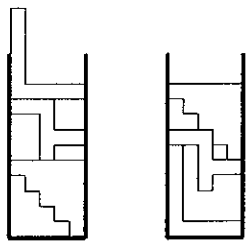
```
12 IS NOT perfect.
28 IS perfect.
```

Problem G: Plaque Pack

The *Knick Knack Plaque Shack* designs plaques of unusual shapes. All the plaques are 1 inch deep, and have a wide variety of shapes, some of which are shown below.



Ben Fitt is one of several workers in the shipping department (part of the Knick Knack Plaque Shack Pack, as they like to call themselves). Each day he is assigned the task of shipping all the plaques of a certain width to the various department stores which sell them. He has at his disposal boxes with a depth of 1 and a width equal to the plaques' width. As the plaques come off the assembly line, he fits them into the boxes one at a time. When placed in a box, each plaque will slide down until some part of it touches the topmost plaque already in the box (or the bottom of the box if it is the first plaque). For example, if the leftmost plaque above came off the assembly line first, followed by the middle and then the rightmost, they would stack up one on top of the other as shown on the left. If they came off the assembly line in reverse order, they would stack up as shown on the right.



When a plaque comes off the assembly line which will not fit into the box (i.e., it sticks up over the top), Ben closes that box, ships it off, and starts a new box. In the above examples, the height of the boxes is only 12, so it would take two boxes for the first ordering of plaques, but only one for the second.

Input

Input will consist of multiple test cases. Each test case will start with a line containing three integers nwb, where n indicates the number of plaques to ship, w indicates the width of each plaque, and b indicates the height of each shipping box. These values will lie in the ranges $1 \ldots 100, 1 \ldots 10$ and $1 \ldots 100$, respectively. Following this line will be n specifications of plaque shapes. Each shape specification starts with a single line containing the integer height h of the plaque ($1 \leq h \leq 10$ and $h \leq b$). Following this will be h lines containing w characters each, where each character is either `'X'` (indicating a part of the plaque) or `'.'`,

indicating empty space. The order in which the plaques appear in the input is the order in which they must be packed in the boxes, and rotating or inverting the plaques is not allowed. The input file will end with the line `0 0 0`.

Output

For each test case, output a single line containing the maximum height of the plaques in each box, in the order in which they are filled.

Sample Input

```
3 5 10
5
XXXXX
.XXXX
..XXX
...XX
....X
4
XXX..
..X..
..XXX
..X..
6
X....
X....
X....
X....
X....
XXXXX
3 5 10
6
X....
X....
X....
X....
X....
XXXXX
4
XXX..
..X..
..XXX
..X..
5
XXXXX
.XXXX
..XXX
...XX
....X
0 0 0
```

Sample Output

```
9 6
10
```

Problem H: The Composite Galaxy

Super Mario takes a break from exploring the Power Square to do some exploring. He finds himself in a galaxy where, in order to earn power stars, he needs to find consecutive composite numbers. Composite numbers are numbers with more than two positive integer divisors (i.e. not prime). For example, the sequence $90 = 2 \times 45, 91 = 7 \times 13, 92 = 2 \times 46, 93 = 3 \times 31, 94 = 2 \times 47, 95 = 5 \times 19, 96 = 2 \times 48$ is a list of 7 consecutive composite numbers. Note that we also presented evidence that they are composite (since 1 and the original number are always positive integer divisors, a third positive integer divisor is all that's needed). Write a program to help Super Mario!

Input h.in

The input is a list of integer ranges for which you must find the longest consecutive sequence of composite numbers. Each line of input provides a single range, consisting of two positive integers, the lower bound and the upper bound (both inclusive), separated by a space. You may assume that all numbers in the range are less than 100,000,000. End of input will be indicated by the line 0 0.

Output h.out

The output is a list of consecutive sequences of composite numbers found for each range. Each line of output provides a consecutive sequence of composite numbers, consisting of the smallest and largest values within the sequence, separated by a space. If there are multiple longest consecutive sequences, output the sequence with larger values.

Sample Input

```
10 20
20 40
90 96
2 1000
10000 20000
987654 1234567
0 0
```

Sample Output

```
14 16
32 36
90 96
888 906
19610 19660
1098848 1098952
```

Problem I: One Person "The Price is Right"

In the game show "The Price is Right", a number of players (typically 4) compete to get on stage by guessing the price of an item. The winner is the person whose guess is the closest one not exceeding the actual price. Because of the popularity of the one-person game show "Who Wants to be a Millionaire", the American Contest Management (ACM) would like to introduce a one-person version of the "The Price is Right". In this version, each contestant is allowed $G(1 \leq G \leq 30)$ guesses and $L(0 \leq L \leq 30)$ lifelines. The contestant makes a number of guesses for the actual price. After each guess, the contestant is told whether it is correct, too low, or too high. If the guess is correct, the contestant wins. Otherwise, he uses up a guess. Additionally, if his guess is too high, a lifeline is also lost. The contestant loses when all his guesses are used up or if his guess is too high and he has no lifelines left. All prices are positive integers.

It turns out that for a particular pair of values for G and L, it is possible to obtain a guessing strategy such that if the price is between 1 and N (inclusive) for some N, then the player can guarantee a win. The ACM does not want every contestant to win, so it must ensure that the actual price exceeds N. At the same time, it does not want the game to be too difficult or there will not be enough winners to attract audience. Thus, it wishes to adjust the values of G and L depending on the actual price. To help them decide the correct values of G and L, the ACM has asked you to solve the following problem. Given G and L, what is the largest value of N such that there is a strategy to win as long as the price is between 1 and N (inclusive)?

Input

The input consists of a number of cases. Each case is specified by one line containing two integers G and L, separated by one space. The end of input is specified by a line in which $G = L = 0$.

Output

For each case, print a line of the form:
`Case` c`:` N
where c is the case number (starting from 1) and N is the number computed.

Sample Input

```
3 0
3 1
10 5
7 7
0 0
```

Sample Output

```
Case 1: 3
Case 2: 6
Case 3: 847
Case 4: 127
```