Problem A - Crypto Columns

The columnar encryption scheme scrambles the letters in a message (or plaintext) using a keyword as illustrated in the following example: Suppose BATBOY is the keyword and our message is MEET ME BY THE OLD OAK TREE. Since the keyword has 6 letters, we write the message (ignoring spacing and punctuation) in a grid with 6 columns, padding with random extra letters as needed:

```
MEETME
BYTHEO
LDOAKT
REENTH
```

Here, we've padded the message with NTH. Now the message is printed out by columns, but the columns are printed in the order determined by the letters in the keyword. Since A is the letter of the keyword that comes first in the alphabet, column 2 is printed first. The next letter, B, occurs twice. In the case of a tie like this we print the columns leftmost first, so we print column 1, then column 4. This continues, printing the remaining columns in order 5, 3 and finally 6. So, the order the columns of the grid are printed would be 2, 1, 4, 5, 3, 6, in this case. This output is called the ciphertext, which in this example would be EYDEMBLRTHANMEKTETOEEOTH. Your job will be to recover the plaintext when given the keyword and the ciphertext.

Input

There will be multiple input sets. Each set will be 2 input lines. The first input line will hold the keyword, which will be no longer than 10 characters and will consist of all uppercase letters. The second line will be the ciphertext, which will be no longer than 100 characters and will consist of all uppercase letters. The keyword THEEND indicates end of input, in which case there will be no ciphertext to follow.

Output

For each input set, output one line that contains the plaintext (with any characters that were added for padding). This line should contain no spacing and should be all uppercase letters.

Sample Input

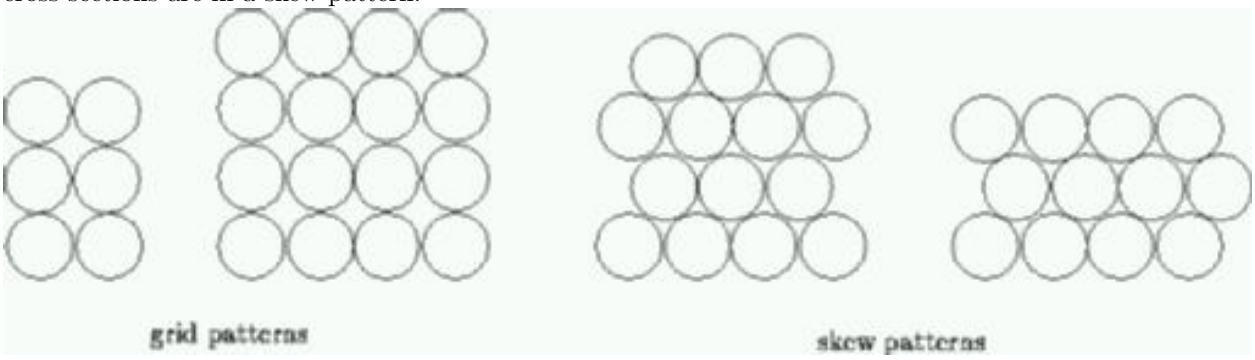
```
BATBOY
EYDEMBLRTHANMEKTETOEEOTH
HUMDING
EIAAHEBXOIFWEHRXONNAALRSUMNREDEXCTLFTVEXPEDARTAXNAARYIEX
THEEND
```

Sample Output

```
MEETMEBYTHEOLDOAKTREENTH
ONCEUPONATIMEINALANDFARFARAWAYTHERELIVEDTHREEBEARSXXXXXX
```

Problem B - Pipe Fitters

Filters, or programs that pass "processed" data through in some changed form, are an important class of programs in the UNIX operating system. A pipe is an operating system concept that permits data to "flow" between processes (and allows filters to be chained together easily.)
This problem involves maximizing the number of pipes that can be fit into a storage container (but it's a pipe fitting problem, not a bin packing problem).
A company manufactures pipes of uniform diameter. All pipes are stored in rectangular storage containers, but the containers come in several different sizes. Pipes are stored in rows within a container so that there is no space between pipes in any row (there may be some space at the end of a row), i.e., all pipes in a row are tangent, or touch. Within a rectangular cross-section, pipes are stored in either a grid pattern or a skew pattern as shown below: the two left-most cross-sections are in a grid pattern, the two right-most cross-sections are in a skew pattern.



Note that although it may not be apparent from the diagram, there is no space between adjacent pipes in any row. The pipes in any row are tangent to (touch) the pipes in the row below (or rest on the bottom of the container). When pipes are packed into a container, there may be "left-over" space in which a pipe cannot be packed. Such left-over space is packed with padding so that the pipes cannot settle during shipping.

Input

The input is a sequence of cross-section dimensions of storage containers. Each cross-section is given as two real values on one line separated by white space. The dimensions are expressed in units of pipe diameters. All dimensions will be less than 2^7. Note that a cross section with dimensions a x b can also be viewed as a cross section with dimensions b x a.

Output

For each cross-section in the input, your program should print the maximum number of pipes that can be packed into that cross section. The number of pipes is an integer – no fractional pipes can be packed. The maximum number is followed by the word "grid" if a grid pattern results in the maximal number of pipes or the word "skew" if a skew pattern results in the maximal number of pipes. If the pattern doesn't matter, that is the same number of pipes can be packed with either a grid or skew pattern, then the word "grid" should be printed.

Sample Input

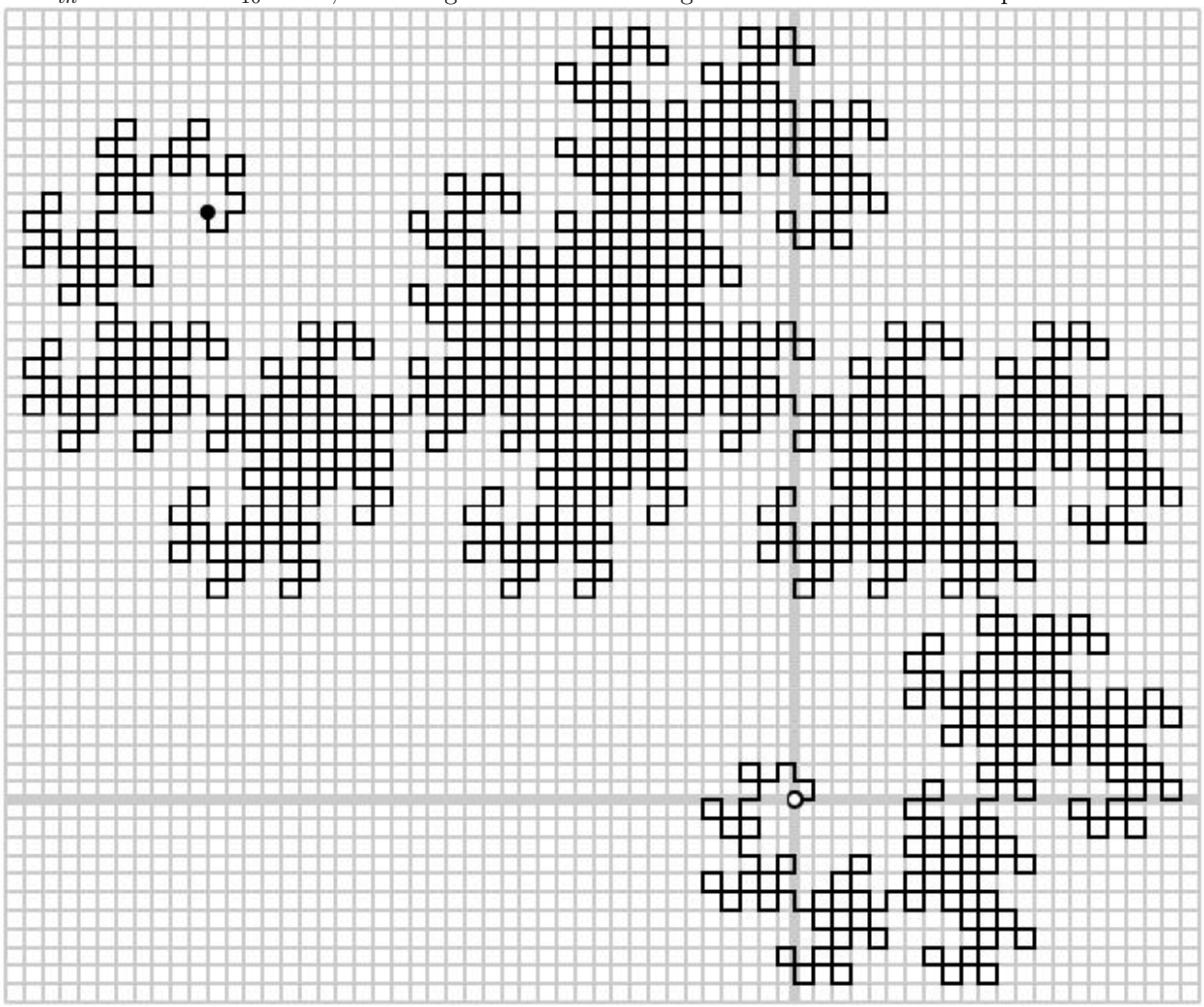
```
3 3
2.9 10
2.9 10.5
11 11
```

Sample Output

```
9 grid
29 skew
30 skew
126 skew
```

Problem C - Fractal

Define r(s) to be the complement of the reverse of the binary string s. i.e. Reverse s and then convert all 1's to 0's and all 0's to 1's. Further define a sequence of binary string as follows: $s_0 = 1$ and $s_n = s_{n-1}1\mathrm{r}(s_{n-1})$. i.e.
$s_0 = 1$
$s_1 = 11\mathbf{1}0$
$s_2 = 110\mathbf{1}1100$
$s_3 = 1101100\mathbf{1}1100100$
...
We then program a robot to move at a steady speed of 1 unit per second and make a right-angle turn according to the characters of s_{10} after every unit of movement. At the k_{th} turn, the robot turns to left if the k_{th} character of s_{10} is a 1, and to right otherwise. The figure below shows the whole path of the robot.



The robot is placed at the origin (the small circle) and face east originally. It ends up at the coordinates (-32,32) (the small spot) after 2048 seconds. The path of the robot is known as a dragon curve, a pretty well-known pattern of fractal.
If the robot is now programmed with input string s_{30} (with identical initial conditions as above), it will keep moving and then stop after 2^{31} seconds. We want to know the location of the robot at any given time.

Input

Input consists of multiple problem instances. Each instance consists of a single non-negative integer n, where $n \leq 10^9$. The input data is terminated by a negative integer. There will be less than 5000 test cases.

Output

For each input integer n, print out the location of the robot right after n second since the robot starts its journey with input string s_{30}. The location should be printed with the format (x,y) in a single line.

Sample input

```
1
2
3
2048
1000000000
-1
```

Sample output

```
(1,0)
(1,1)
(0,1)
(-32,32)
(9648,-31504)
```

Problem D - Fabled Rooks

We would like to place n rooks, $1 \leq n \leq 5000$, on a n x n board subject to the following restrictions

- The i-th rook can only be placed within the rectangle given by its left-upper corner (xl_i, yl_i) and its right-lower corner (xr_i, yr_i), where $1 \leq i \leq n$, $1 \leq xl_i \leq xr_i \leq n$, $1 \leq yl_i \leq yr_i \leq n$.
- No two rooks can attack each other, that is no two rooks can occupy the same column or the same row.

The input consists of several test cases. The first line of each of them contains one integer number, n, the side of the board. n lines follow giving the rectangles where the rooks can be placed as described above. The i-th line among them gives xl_i, yl_i, xr_i, and yr_i. The input file is terminated with the integer 0 on a line by itself.
Your task is to find such a placing of rooks that the above conditions are satisfied and then output n lines each giving the position of a rook in order in which their rectangles appeared in the input. If there are multiple solutions, any one will do. Output IMPOSSIBLE if there is no such placing of the rooks.

Sample input

```
8
1 1 2 2
5 7 8 8
2 2 5 5
2 2 5 5
6 3 8 6
6 3 8 5
6 3 8 8
3 6 7 8
8
1 1 2 2
5 7 8 8
2 2 5 5
2 2 5 5
6 3 8 6
6 3 8 5
6 3 8 8
3 6 7 8
0
```

Output for sample input

```
1 1
5 8
2 4
4 2
7 3
8 5
6 6
3 7
1 1
5 8
2 4
4 2
7 3
8 5
6 6
3 7
```

Problem E - Luggage

Peter and his friends are on holiday, so they have decided to make a trip by car to know the north of Spain. They are seven people and they think that two cars are enough for their luggage.
It's time to leave and a heap of suitcases are awaiting out of the cars. The drivers disagree about which suitcase must be put into each trunk, because nobody wants one trunk to carry more weight than the other one. Is it possible that the two trunks load with the same weight? (Obviously without unpacking the suitcases!)
Consider m sets of numbers representing suitcase weights, you must decide for each one, if it is possible to distribute the suitcases into the trunks, and the two trunks weigh the same.

Input

The first line of the input contains an integer, m, indicating the number of test cases. For each test case, there is a line containing n+1 integers ($1 \leq n \leq 20$) separated by single spaces. The first integer is n and the remaining n integers are the weights of each suitcase. The total sum of the weights of all the suitcases is less or equal to 200 kilograms.

Output

The output consists of m lines. The i-th line corresponds with the i-th set of suitcases weight and contains the string YES or NO, depending on the possibility that the two trunks load with the same weight for the respective test case.

Sample Input

```
3
5 1 2 1 2 1
10 2 3 4 1 2 5 10 50 3 50
20 3 5 2 7 1 7 5 2 8 9 1 25 15 8 3 1 38 45 8 1
```

Sample Output

```
NO
YES
YES
```

Problem F - Tight words

Given is an alphabet $\{0, 1, \ldots, k\}$, $0 \leq k \leq 9$. We say that a word of length n over this alphabet is tight if any two neighbour digits in the word do not differ by more than 1.
Input is a sequence of lines, each line contains two integer numbers k and n, $1 \leq n \leq 100$. For each line of input, output the percentage of tight words of length n over the alphabet $\{0, 1, \ldots, k\}$ with 5 fractional digits.

Sample input

```
4 1
2 5
3 5
8 7
```

Output for sample input

```
100.00000
40.74074
17.38281
0.10130
```

Problem G - Marks Distribution

In the last exam period, a student was examined in N subjects and got a total of T marks. He has passed in all N subjects. The minimum mark for passing in each subject is P. You have to calculate the number of ways the student can get the marks. For example, if N=3, T=34 and P=10 then the marks in the three subject could be as follows.

	Subject 1	Subject 2	Subject 3
1	14	10	10
2	13	11	10
3	13	10	11
4	12	11	11
5	12	10	12
6	11	11	12
7	11	10	13
8	10	11	13
9	10	10	14
10	11	12	11
11	10	12	12
12	12	12	10
13	10	13	11
14	11	13	10
15	10	14	10

So there are 15 solutions, i.e. F (3, 34, 10) = 15.

Input

In the first line of the input there will be a single positive integer K followed by K lines each containing a single test case. Each test case contains three positive integers denoting N, T and P respectively. The values of N, T and P will be at most 70. You may assume that the final answer will fit in a standard 32-bit integer.

Output

For each input, print in a line the value of F (N, T, P).

Sample Input

```
1
3 34 10
```

Output for Sample Input

```
15
```

Problem H - EKG Sequence

The EKG sequence is a sequence of positive integers generated as follows: The first two numbers of the sequence are 1 and 2. Each successive entry is the smallest positive integer not already used that shares a factor with the preceding term. So, the third entry in the sequence is 4 (being the smallest even number not yet used). The next number is 6 and the next is 3. The first few numbers of this sequence are given below.

1, 2, 4, 6, 3, 9, 12, 8, 10, 5, 15, 18, 14, 7, 21, 24, 16, 20, 22, 11, 33, 27

The sequence gets its name from its rather erratic fluctuations. The sequence has a couple of interesting, but non-trivial, properties. One is that all positive integers will eventually appear in the sequence. Another is that all primes appear in increasing order. Your job here is to find the position in the sequence of a given integer.

Input

Input consists of a number of test cases. Each case will be a line containing a single integer n, $1 \leq n \leq 300000$. An input of 0 follows the last test case. Note that the portion of the EKG sequence that contains all integers $\leq$ 300,000 will not contain an integer $>$ 1,000,000.

Output

Each test case should produce one line of output of the form: The number n appears in location p. where n is the number given and p is the position of n in the EKG sequence. You are guaranteed that p will be no larger than 1,000,000.

Sample Input

```
12
21
2
33
100000
299977
0
```

Sample Output

```
The number 12 appears in location 7.
The number 21 appears in location 15.
The number 2 appears in location 2.
The number 33 appears in location 21.
The number 100000 appears in location 97110.
The number 299977 appears in location 584871.
```