2011 ACM-ICPC Asia Phuket Regional Programming Contest

4 November 2011

Hosted by Prince of Songkla University, Phuket Campus
Thailand

- There are **10** problems (A-J) to solve within 5 hours.
- Solve as many problems as you can, in an order of your choice.
- Use C or C++ or Java to program at your convenience for any problems.
- **Input and output** of each program are **standard input** and **output**.

Problem A	Water Gate Management
Problem B	Binary Search Tree
Problem C	Fun Coloring
Problem D	Twin Apparent Primes!!
Problem E	Queen Game
Problem F	Spelling Suggestion
Problem G	Coalescing Continents
Problem H	Fence Making
Problem I	Paths in a Tree
Problem J	Consecutive Sums

Problem A
Water Gate Management

A dam has n water gates to let out water when necessary. Each water gate has its own capacity, water path and affected areas in the downstream. The affected areas may have a risk of flood when the water gate is open. The cost of potential damage caused by a water gate is measured in number calculated from the number of people and areas estimated to get affected.

Suppose a water gate G_i has the volumetric flow rate of F_i m^3/hour and the damage cost of C_i. In a certain situation, the dam has the volume V m^3 of water to flush out within T hours. Your task is to manage the opening of the water gates in order to get rid of *at least* the specified volume of water within a limited time in condition that the damage cost is minimized.
For example, a dam has 4 water gates and their properties are shown in the following table.

Water Gate	G_1	G_2	G_3	G_4
Flow rate (m^3/hour)	720,000	50,000	130,000	1,200,000
Cost	120,000	60,000	50,000	150,000

Case 1: You have to flush out the water 5 million m^3 within 7 hours. The minimum cost will be 120,000 by letting the water gate G1 open for 7 hours.
Case 2: You have to flush out the water 5 million m^3 within 30 hours. The minimum cost will be 110,000 by letting the water gates G_2 and G_3 open, for example, G_2 is open for 29 hours and G_3 is open for 28 hours.

Note that each water gate is independent and it can be open only in a unit of whole hour (no fraction of hour).

Input
The first line includes an integer n indicating number of water gates ($1 \le n \le 20$). Then the next n lines contain, in each line, two integers: F_i and C_i corresponding to the flow rate (m^3/hour) and the damage cost of the water gate G_i respectively. The next line contains the number m which is the number of test cases ($1 \le m \le 50$). The following m lines contain, in each line, two integers: V and T corresponding to the volume (m^3) of water to let out within T hours. $1 \le F_i, V, C_i \le 10^9$, $1 \le T \le 1000$)

Output
For each test case, print out the minimum cost in the exact format shown in the sample output below. If it is **not** possible to let out the water of volume V in T hours from the dam, print out "`IMPOSSIBLE`" (without quotation marks).

Sample input	Sample output
4 720000 120000 50000 60000 130000 50000 1200000 150000 3 5000000 7 5000000 30 63000000 24	Case 1: 120000 Case 2: 110000 Case 3: IMPOSSIBLE

B

Problem B
Binary Search Tree

A binary search tree is a binary tree that satisfies the following properties:

- The left subtree of a node contains only nodes with keys less than the node's key.
- The right subtree of a node contains only nodes with keys greater than the node's key.
- Both the left and right subtrees must also be binary search trees.

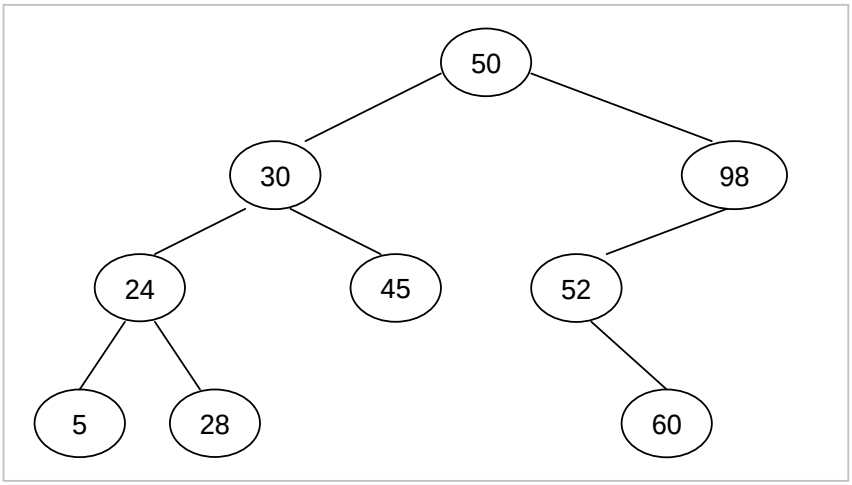


Figure 1. Example binary search tree

Pre-order traversal (Root-Left-Right) prints out the node's key by visiting the root node then traversing the left subtree and then traversing the right subtree. *Post-order traversal* (Left – Right-Root) prints out the left subtree first and then right subtree and finally the root node. For example, the results of pre-order traversal and post-order traversal of the binary tree shown in Figure 1 are as follows:

Pre-order: `50 30 24 5 28 45 98 52 60`
Post-order: `5 28 24 45 30 60 52 98 50`

Given the pre-order traversal of a binary search tree, you task is to find the post-order traversal of this tree.

Input

The keys of all nodes of the input binary search tree are given according to pre-order traversal. Each node has a key value which is a positive integer less than 10^6. All values are given in separate lines (one integer per line). You can assume that a binary search tree does not contain more than 10,000 nodes and there are no duplicate nodes.

Output

The output contains the result of post-order traversal of the input binary tree. Print out one key per line.

Sample input	Sample output
50 30 24 5 28 45 98 52 60	5 28 24 45 30 60 52 98 50

Problem C
Fun Coloring

Consider the problem called FUN COLORING below.

FUN COLORING PROBLEM

INSTANCE: A finite set U and sets $S_1, S_2, S_3, \ldots, S_m \subseteq U$ and $|S_i| \leq 3$.

PROBLEM: Is there a function $f : U \mapsto \{\text{RED, BLUE}\}$ such that at least one member of each S_i is assigned a different color from the other members?

Given an instance of FUN COLORING PROBLEM, your job is to find out whether such function f exists for the given instance.

Input

In this problem $U = \{x_1, x_2, x_3, \ldots, x_n\}$. There are k instances of the problem. The first line of the input file contains a single integer k and the following lines describe k instances, each instance separated by a blank line. In each instance the first line contains two integers n and m with a blank in between. The second line contains some integers i's representing x_i's *in* S_1, each i separated by a blank. The third line contains some integers i's representing x_i's *in* S_2 and so on. The line m+2 contains some integers i's representing x_i's *in* S_m. Following a blank line, the second instance of the problem is described in the same manner and so on until the k^{th} instance is described. In all test cases, $1 \leq k \leq 13$, $4 \leq n \leq 22$, and $6 \leq m \leq 111$.

Output

For each instance of the problem, if f exists, print a Y. Otherwise, print N. Your solution will contain one line of k Y's (or N's) without a blank in between. The first Y (or N) is the solution for instance 1. The second Y (or N) is the solution for instance 2, and so on. The last Y (or N) is the solution for instance k.

Sample input	Sample output
2 5 3 1 2 3 2 3 4 1 3 5 7 7 1 2 1 3 4 2 4 3 2 3 1 4 5 6 7	YN

D

Problem D
Twin Apparent Primes!!

Prime numbers have very interesting applications in Computer Science. This problem is obviously related with prime numbers but you need to know the following two definitions to solve this problem.

Apparent Prime: A positive number that is not divisible by all integer numbers greater than 1 and less than or equal to t is called apparent prime. The value of t will be supplied for you.

Twin Apparent Primes: If the difference of two apparent primes is 2 then they are called twin apparent primes.

Given the value of n and t you will have to find two n-digit apparent prime numbers p and (p+2).

Input

The input file contains at most 1001 lines of inputs. Each line contains two positive integers n ($3500 \leq n \leq 5000$) and t ($t \leq 8000$).

Input is terminated by a line containing two zeroes. These two numbers are of course invalid input and should not be processed.

Output

For each line of input produce one line of output. This line contains an n-digit apparent prime number p, such that (p+2) is also an apparent prime. If here is more than one such number, anyone will do.

Sample input	Sample output
2 6 0 0	17

Although n>=3500, in the sample n=2 so that the output fits in reasonable space.

Problem E
Queen Game

Queen game is played in a chessboard of size R rows and C columns. Rows are numbered from 1 to R and columns are numbered from 1 to C. The topmost square is in row 1 and column 1. The game is a 2 player game. Initially there are N queens placed in various squares of the chessboard. In his turn, the player picks a queen and moves it either towards the top vertically, or towards the left horizontally or towards the top-left diagonally and the queen should always stay on the board. When the queen reaches square (1, 1) it is removed from the board. The player who gives the last move wins. Each square is big enough to accommodate infinite number of queens. The players give their moves by turns. You are given the size of the chessboard and the initial positions of the N queens. Assuming that both of the players play perfectly your task is to determine who will win this game.

Input

First line of the input contains T the number of test cases. Each test case starts with a line containing 3 integers R($1 \le R \le 25$), C($1 \le C \le 10^{15}$) and N($1 \le N \le 1000$). Each of the next N lines contains the positions of N queens. The position is denoted by two integers. The first integer is the row number and the second integer is the column numbers.

Output

For each test case output "YES" if the first player has a winning strategy and "NO" otherwise.

Look at the output for sample input for details.

Sample Input	Sample output
3 5 5 1 2 3 5 5 2 4 4 4 4 5 5 3 1 2 2 1 2 2	NO NO YES

F

Problem F
Spelling Suggestion

A spelling suggestion is a part of spelling correction program that generates a set of plausible replacements for words that are likely to be misspelled. One way to measure the plausibility of these replacements is to compute their edit distance against a given misspelled word. The edit distance between two words is the total number of edit operations that have to be done in order to transform one word into the other. Normally these edit operations are insertion, deletion and substitution of a single character including transposition of 2 consecutive characters.

For example, for a word “wonder”, if the deletion is applied at the character 'o', this word will transform into “wnder”. And if the substitution with 'a' is applied at 'o', it will become “wander”. And if the transposition is applied at “er”, it will become “wondre”.

In this edit distance strategy, the degree of similarity between two words is up to their minimum edit distance. If the minimum edit distance between $word_1$ and $word_2$ is lower than the distance between $word_1$ and $word_3$, then $word_1$ is more similar to $word_2$ than to $word_3$. So the $word_2$ is a better spelling suggestion for $word_1$, comparing with $word_3$.

Suppose that you are an employee of a software company which needs to build up a prototype of spelling suggestion program. This prototype tends to be a part of word processing software. The requirement is that it has to use the edit distance strategy for their spelling suggestion. But the substitution operation has to be redefined to match the behavior of mistyping. The cost of substitution of a character with another character depends on the position of them on the keyboard layout. If they are close to each other, the cost is only half of the normal one. For this purpose, the substitution is categorized into near-substitution and far-substitution. Their costs are defined as 1 and 2 respectively. And the costs for insertion, deletion and transposition are 2. In addition, this program must run fast enough to pass the time limit that is set by your manager. By the way for this prototype, an English QWERTY keyboard layout is chosen to be used.

Goal

To generate optimum spelling suggestions for each input word, where each optimum spelling suggestion is a word in dictionary that has the least minimum edit distance from the given input word. The time limit for 5,000 misspelled words is less than or equal to 5 minutes.

Input

Input is a standard input which contains 3 parts of data. Each of these three parts ends with a blank line.

- The first part is a set of near-substitution rules, where each rule is kept in one line. Each line has two fields. Each filed is separated by a space. The total no. of rules is less than or equal to 150
  - The first field is a character where it can be near-substituted with other characters.
  - The second field is a sequence of characters which can be near-substituted for the character in the previous field. There is no space in this field.
  - The characters that may be contained in this part are characters that can be typed in using a generic English keyboard layout, which are alpha-numeric characters and some punctuations without space or tab. They are listed as the following.

```
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789`~
!@#$%^&*()-_=+\|[{]};:'",<.>/?
```

- The second part is a sequence of words in dictionary, where each word is kept in one line. The total no. of words is less than or equal to 150,000.
  - The characters in dictionary are alphabetical characters with an apostrophe punctuation.
    - `abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ'`
- The third part is a sequence of words that need to be checked for their spellings. Each of these words is also kept in one line. The total no. of words is less than or equal to 5,000.
  - The characters that may be contained in this part are the same as characters in the first part, which are alpha-numeric characters and some punctuation. (See the first part.)
- Since each of these three parts end with a blank line, the third blank line is the termination of the input.

Output

For each word in the third part, write a line which contains 3 parts of information, separated with a colon.

- The first part is the given input word.
- The second part is the minimum edit distance between the input word and suggestion word(s).
- The third part is an ascending sorted sequence of suggestion word(s), separated with a space. There is no space left after the last word.

Line no.	**Sample Input**
1	a AqQsSzZ
2	b BgGvVnN
3	p P0);:oO[{
4	r R4$fFeEtT
5	z ZaAxX
6	
7	a
8	A
9	b
10	B
11	Z
12	angel
13	angle
14	anger
15	angry
16	ABC
17	
18	x
19	s
20	z
21	xx

22	xxx
23	angre
24	angri
25	angrt
26	angel
27	ACB
28	BAC
29	CAB
30	

Line no.	**Sample Output**
1	x:2:A B Z a b
2	s:1:a
3	z:1:A Z a
4	xx:4:A B Z a b
5	xxx:6:A ABC B Z a b
6	angre:2:anger angle angry
7	angri:2:angry
8	angrt:2:anger angry
9	angel:0:angel
10	ACB:2:ABC
11	BAC:2:ABC
12	CAB:4:A ABC B

More Explanations

In this sample input, there are 5 near-substitution rules (line no. 1-5), 10 words in dictionary (line no. 7-16) and 12 words looking for their suggestions (line no. 18-29).
In the sample output, there are 12 lines for each corresponding 12 words from the input.
For the 1st word, the minimum edit distance between `x` and its suggestions (`A B Z a b`) is 2. All of them are the (far) substitution costs.
For the 2nd word, the minimum edit distance between `s` and its suggestion (`a`) is 1, which is a near-substitution cost, guided by the 1st substitution rule.
For the 3rd word, the minimum edit distance between `z` and its suggestion (`A Z a`) are 1, which is a near-substitution cost, guided by the 1st or 5th substitution rule.
For the 4th word, the minimum edit distance is 4, which are summed from one far-substitution cost and one deletion cost.
For the 5th word, the minimum edit distance is 6. The costs between `xxx` and (`A B Z a b`) are from two deletion and one far-substitution costs. The cost between `xxx` and `ABC` re from three far-substitution costs.
For the 6th word, the cost between `angre` and `anger` is from one transposition costs. The costs between `angre` and (`angle angry`) are from one far-substitution costs.
For the 7th word and 8th word, seem to be similar. But the cost of 7th word is from one far-substitution costs. But the cost between `angrt` and `anger` is from 2 near-substitution costs (`r` substitutes with `e` and `t` substitutes with `r`) according to the near-substitution rule no. 4.
For the 9th word, the word is exactly matched within dictionary. So the cost is 0.
For the 10th and 11th words, each cost is from one transposition costs.
For the 12th words, the cost between `CAB` and (`A B`) is from two deletion costs. The cost between `CAB` and (`ABC`) is from one deletion and one insertion costs. Please be notify that it is not from 2 transpositions of `CAB` to `ACB` and then `ACB` to `ABC`.

Problem G
Coalescing Continents

Continental drift refers to the idea that once the earth's continents were all in one piece and with time they drifted away from each other and arrived at their present positions.

In this problem, we are concerned with the reverse process – *Continental Coalesce*.

For the sake of brevity, let's assume the continents were initially merged together forming a square. With time, the square disintegrated into **K** different continents in the shape of rectangles and drifted away from its source.

You are provided with the current locations of the continents (i.e. rectangles). These locations were outlined with the help of satellites. Your first job is to figure out whether the data provided is a valid one. The data is valid if you can move the rectangles and form a square. In one move, you can pick any rectangle and push it one unit in one of the four directions (north, south, east or west). The continents can slide underneath one another during movement – meaning, more than one continent can occupy a particular position simultaneously. If it is possible to form a square then you are also required to find the minimum number of moves to do that. The final position of the square is not the primary concern here. Note that in the final position, two continents cannot overlap and the desired square cannot have any holes in it.

Input

The first line of input is an integer **T** (**T** <= 200) that indicates the number of test cases. Each case consists of 20 lines containing 20 characters. Each character can either be a dot(.) representing an empty space or an ***x*** (ASCII value 120). Every ***x*** character will be part of some continent.

Notes:

- In the input grid, it's guaranteed that, ***x*** characters from different continents will not be adjacent to each other. Any ***x*** character will be part of some rectangle
- Two cells are adjacent if they share a common edge.
- There will be exactly 25 ***x*** characters in the grid.
- The number of continents, **K**, will be at most 5.

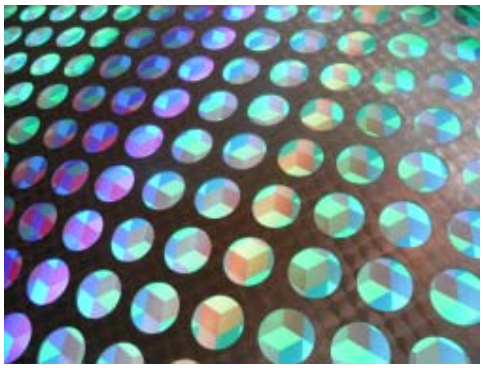
- There is a blank line before the start of every case.
- The rectangles and the desired square are axis parallel.
- The earth here is flat. That is, the first and last row is NOT adjacent to each other. Same thing holds for the first and last column.

Output

For each case, output the case number first. If it's not possible to form a square by moving the rectangles around, output "invalid data", quotes for clarity. If it is possible, output the minimum number of moves required to complete the task.

Sample Input	Sample Output
2 ....xxxxx..... ..xxxxx..... ....xx.....xxx... ....xxxxx..... ..xxxxx..... ....x.. ....xx..... ..xx..... ..xx..... ..xx..... ..xx..... ..xx..... .....xxxxxx..... .....xxxxxx.....	Case 1: 13 Case 2: invalid data

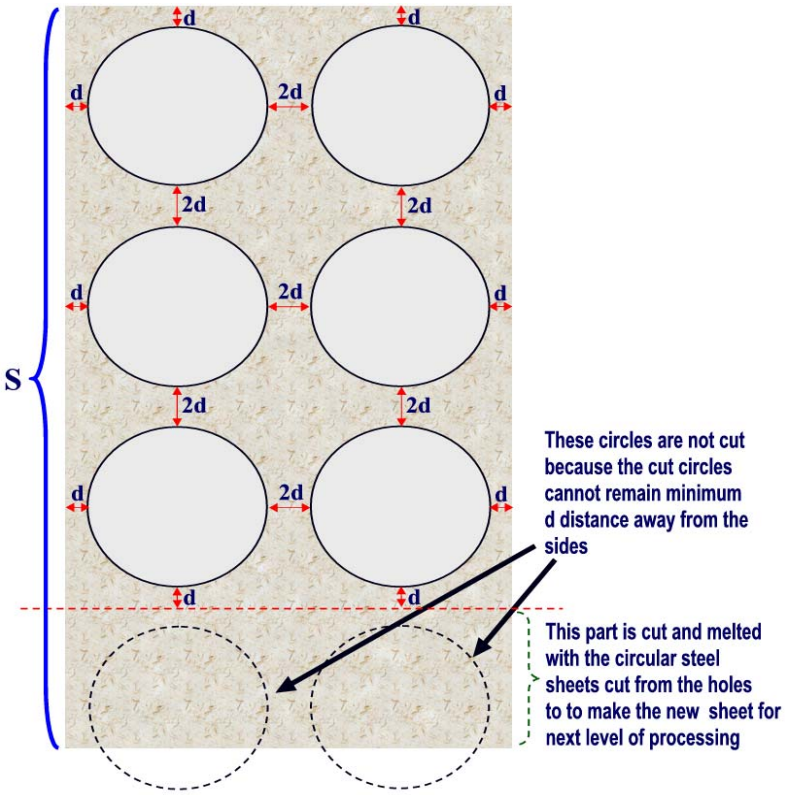
Problem H
Fence Making



The City of Haka is very famous for its traffic jam. This has given birth to many problemsetters (Those who pose questions for programming contests) as in traffic jam some people have nothing meaningful to do other than thinking about new problems. Most wide roads in Haka have fence along the divider. Perforated steel sheets (As shown in the figure on the left) are often used in these fences.

In this problem we will discuss the manufacturing process of one kind of perforated steel strips and ask you to solve a problem related to this building process.

For this problem the rectangular perforated steel strips have two circular holes in each row. Circular holes in the same row are 2d distance apart. Two consecutive circular holes in the same column are also 2d distance apart. The distance of each circle from its nearest side is d. The radius of all the circles is r. So automatically the width of the metal sheet/strip becomes (4d+4r). The length of the initial sheet is S. Such a sheet with holes in it is shown in the picture on the right. Holes are only drilled if they can be cut according to rules mentioned above. The circular steel sheets which are obtained by cutting the holes and part of the sheet that is unused (Such as portion below the red dotted line in the figure on the right) are melted to create a new steel sheet/strip of width (4d+4r). Now holes are cut according to the rules mentioned above. This process is repeated until the newly created sheet is so small that two holes (In the same row) cannot be created in it following the above-mentioned restrictions. Suppose C(d, r, S) denote the total number of holes that are created. Now given the minimum possible value of r (r_{min}), maximum possible value of r (r_{max}), minimum possible value of d (d_{min}) and maximum possible value of d (d_{max}) you will have to find the total number of holes created.



In other words you will have to find $\sum_{r=r_{min}}^{r_{max}} \sum_{d=d_{min}}^{d_{max}} C(d,r,S)$. It should be clear to you now that the value of d and r are always integer. Your method should be quite efficient. You must assume that the initial given strip and the strips created later on have equal and uniform thickness in all places.

Input

Input file contains 1000 sets of inputs. The input for each set is given in a single line. Each line contains five integers r_{min} ($5{,}000 \le r_{min} \le 10{,}000$), r_{max} ($0 \le r_{max} - r_{min} \le 1{,}000$) , dmin ($1 \le d_{min} \le 21$), dmax ($0 \le d_{max} - d_{min} \le 100$)and S ($1{,}000{,}000 \le S \le 2{,}000{,}000{,}000$). By now it should be clear to you that value of r and d can only be a round number.

Input is terminated by a line containing five zeroes. This line should not be processed.

Output

For each set/line of input produce one line of output. This line contains an integer, which denotes the value of $\sum_{r=r_{min}}^{r_{max}} \sum_{d=d_{min}}^{d_{max}} C(d,r,S)$. You can safely assume that the value of this integer will comfortably fit in a 64-bit signed integer. Errors not exceeding 10^{-7} % will be ignored.

Sample Input	Sample Output
9682 9719 18 29 71757646 5746 5958 19 24 1942485264 0 0 0 0 0	15404518 1918408970

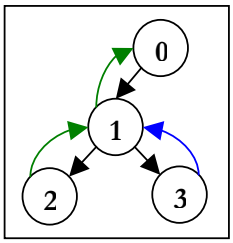
I

Problem I
Paths in a Tree

You are given a tree (a connected graph with no cycles), and the edges of the tree which are for some reason directed; your task is to add **minimum** number of special paths in the tree such that it's possible to go from any node to another. The rules for the special paths are noted below:

1. A special path consists of some continuous edges (from the tree) and nodes.
2. In a special path, the edges should be in opposite directions as they are in the tree.
3. A node or an edge can be visited at most once in a special path.
4. Multiple special paths may have common nodes or edges.

For example, in the picture below, a tree is drawn, the black arrows represent the edges and their directions, circles represent nodes. Then we need two special paths. One path is **2-1-0** (green arrow), another is **3-1** (blue arrow). Instead of the path **3-1** we can add **3-1-0**. You cannot add a path like **1-3** or **0-1-2** because of rule 2. You cannot add **0-2** or **2-3-0** because of rule 1.



Input

Input starts with an integer **T (≤ 30)**, denoting the number of test cases.

Each case starts with a line containing an integer **N (2 ≤ N ≤ 20000)**, where **N** denotes the number of nodes. The nodes are numbered from **0** to **N-1**. Each of the next **N-1** lines contains two integers **u v (0 ≤ u, v < N, u ≠ v)** meaning that there is an edge from **u** to **v**.

Output

For each case, print the case number and the minimum number of special paths required such that it's possible to go from any node to another.

Sample Input	Sample Output
2 4 0 1 1 2 1 3 5 0 1	Case 1: 2 Case 2: 3

1 2 1 3 0 4	

J

Problem J
Consecutive Sums

The sum of p (p>0) consecutive integers can often be equal to the sum of next q consecutive positive integers. For example:

9+10+11+12 = 13+14+15, Here p = 4 and q = 3
4+5+6+7+8 = 9+10+11, Here p = 5 and q = 3.

Given the value of q, how many possible values of p are there?

Input

The input file contains at most 1500 lines of inputs. Each line contains a positive integer less than 10^{14}, which denotes the value of q. Input is terminated by a line containing a single zero. This line should not be processed.

Output

For each line of input produce one line of output. This line contains an integer, which denotes the total number of possible values of p.

Sample Input	Sample Output
5 1 0	6 2