The 2017 ACM-ICPC Asia Hong Kong Regional Contest

November 19, 2017

Clarifications:

- For Problem A, there may be multiple values of x and y that minimize $F(x, y, z, n)$, and you may output any of them.
- For Problem H, when $n = 3$, the input satisfies that $k \leq 100000$.

Organizer:

Sponsors:

Problem A. Fermat‘s Optimization Problem

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2s

Consider the error function $F(x, y, z, n) = |x^n + y^n - z^n|$, where $|v|$ means the absolute value of v. Given two positive integers n and z, our problem is to find two positive integers x and y such that $x < y < z$ and the error value $F(x, y, z, n)$ is minimized.

For example, if we are given $n = 3$ and $z = 9$, then the solution is: $x = 6$ and $y = 8$. This solution yields the error value 1.

Input

The first line contains the number of test cases T ($T < 10$). Each subsequent line corresponds to a test case, which contains two positive integers n ($2 < n < 10$) and z ($1 < z < 100000$).

Output

For each test case, output the value of x, y, and $F(x, y, z, n)$ in a line, separated by spaces.

Example

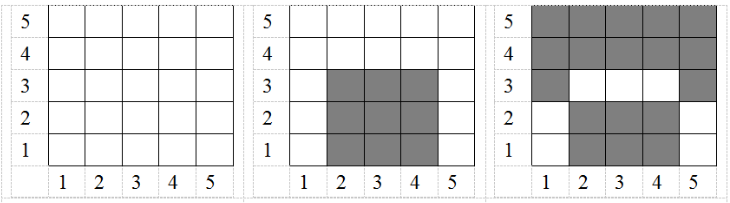
standard input	standard output

2
3 9
3 7
	6 8 1
	5 6 2

Problem B. Black and White

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1s

Consider a square map with $N \times N$ cells. We indicate the coordinate of a cell by (i, j), where $1 \leq i, j \leq N$. Each cell has a color either white or black. The color of each cell is initialized to white. The map supports the operation $flip([x_{low}, x_{high}], [y_{low}, y_{high}])$, which flips the color of each cell in the rectangle $[x_{low}, x_{high}] \times [y_{low}, y_{high}]$. Given a sequence of flip operations, our problem is to count the number of black cells in the final map. We illustrate this in the following example. Figure (a) shows the initial map. Next, we call $flip([2, 4], [1, 3])$ and obtain Figure (b). Then, we call $flip([1, 5], [3, 5])$ and obtain Figure (c). This map contains 18 black cells.



(a) initial map | (b) after flipping the rectangle [2,4]×[1,3] | (c) after flipping the rectangle [1,5]×[3,5]

Input

The first line contains the number of test cases T ($T < 10$). Each test case begins with a line containing two integers N and K ($1 < N, K < 10000$), where N is the parameter of the map size and K is the number of flip operations. Each subsequent line corresponds to a flip operation, with four integers: $x_{low}, x_{high}, y_{low}, y_{high}$.

Output

For each test case, output the answer in a line.

Example

standard input	standard output

`1`
`5 2`
`2 4 1 3`
`1 5 3 5`
	`18`

Problem C. Equivalence of Sudoku

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1s

A Sudoku solution is one 9×9 matrix in which each digit from 1 to 9 occurs only once every row, every column and every 3×3 square. Given one Sudoku solution, S, it is easy to generate many other equivalent solutions, by performing one or more of the elementary transformations: R for a rotation by 90, 180 or 270 degrees, M for a mirror image by flipping along the horizontal or vertical axis, B for a bijective substitution of each of the digits 1 to 9 being mapped to another set of digits 1 to 9. It can be seen that S_1, S_2, and S_3 are all equivalent to S in the following example.

S Sudoku matrix

6	7	8	5	3	2	4	9	1
3	1	5	9	4	7	6	8	2
9	4	2	6	1	8	7	5	3
2	8	3	7	9	6	5	1	4
4	9	7	2	5	1	3	6	8
5	6	1	4	8	3	9	2	7
8	3	9	1	7	5	2	4	6
7	5	6	8	2	4	1	3	9
1	2	4	3	6	9	8	7	5

S_1 Rotation of S by 180 degrees

5	7	8	9	6	3	4	2	1
9	3	1	4	2	8	6	5	7
6	4	2	5	7	1	9	3	8
7	2	9	3	8	4	1	6	5
8	6	3	1	5	2	7	9	4
4	1	5	6	9	7	3	8	2
3	5	7	8	1	6	2	4	9
2	8	6	7	4	9	5	1	3
1	9	4	2	3	5	8	7	6

S_2 Mirror image of S (flip along y)

1	9	4	2	3	5	8	7	6
2	8	6	7	4	9	5	1	3
3	5	7	8	1	6	2	4	9
4	1	5	6	9	7	3	8	2
8	6	3	1	5	2	7	9	4
7	2	9	3	8	4	1	6	5
6	4	2	5	7	1	9	3	8
9	3	1	4	2	8	6	5	7
5	7	8	9	6	3	4	2	1

S_3 Bijection of S: $1 \leftrightarrow 2, 3 \leftrightarrow 4, 5 \leftrightarrow 6, 7 \leftrightarrow 8$

5	8	7	6	4	1	3	9	2
4	2	6	9	3	8	5	7	1
9	3	1	5	2	7	8	6	4
1	7	4	8	9	5	6	2	3
3	9	8	1	6	2	4	5	7
6	5	2	3	7	4	9	1	8
7	4	9	2	8	6	1	3	5
8	6	5	7	1	3	2	4	9
2	1	3	4	5	9	7	8	6

A partial Sudoku is one in which not all of the 81 cells are filled up. We say that a partial Sudoku P_1 is subsumed by (less than) another one P_2 if there exists P which is equivalent to P_2, and P is obtained from P_1 by filling in one or more cells. In the following example, P_1 is subsumed by P_2 with respect to P, where P can be obtained from P_2 via a bijective mapping $\{1, 2, 3, 4, 5, 6, 7, 8, 9\}$ $\{2, 9, 3, 7, 8, 6, 1, 5, 4\}$, after rotating P_2 clockwise by 90 degrees. Similarly, we say that P_2 subsumes P_1 or P_2 is greater than P_1 with respect to P. Note that it is not necessary that one of the matrices is completely filled.

P_1

9	1	3	5		6	2		
5	2	8		1	7		9	4
6			2		8	1	5	3
8	9	1				3		5
4	3	6	1	5	9	7	2	8
		5	8	3	4	9	1	6
1	5	4	7		3	8	6	
	6		4	8	1	5	3	
	8	2		6	5	4	7	1

P_2

4	9	3	8	5	6	2	1	7
5	2	8	9	1	7	6	3	4
1	6	7	3	4	2	5	8	9
6	4	5	1	2	9	3	7	8
9	7	2	4	8	3	1	5	6
8	3	1	6	7	5	4	9	2
3	5	4	7	6	8	9	2	1
7	1	9	2	3	4	8	6	5
2	8	6	5	9	1	7	4	3

P

9	1	3	5	4	6	2	8	7
5	2	8	3	1	7	6	9	4
6	4	7	2	9	8	1	5	3
8	9	1	6	7	2	3	4	5
4	3	6	1	5	9	7	2	8
2	7	5	8	3	4	9	1	6
1	5	4	7	2	3	8	6	9
7	6	9	4	8	1	5	3	2
3	8	2	9	6	5	4	7	1

To summarize, between every two Sudoku matrices S_1 and S_2, there are four possible relationships: S_1 is equivalent to S_2 (E), S_1 is less than S_2 (L), S_1 is greater than S_2 (G), S_1 is incomparable to S_2 (I). Write a program to read in a list of n Sudoku matrices and determine the pairwise relationship between them. The output is an $n \times n$ matrix showing the relationships (E, L, G, I). It is obvious that the diagonal elements are all E (every matrix is equivalent to itself trivially).

Input

The first line contains the number of matrices n. Each subsequent set of 9 lines corresponds to a (partial) Sudoku matrix. An unfilled cell is represented as 0 (instead of blank). Assume that $2 < n \leq 400$, all inputs are correct and all matrices are correct (partial) Sudoku matrices.

Output

For each pair of input matrices, determine the relationship between them and output that relationship as a single letter $O \in \{E, L, G, I\}$, one line for each input matrix.

Example

standard input	standard output
`3`	`EGI`
`0 7 4 5 2 3 0 9 8`	`LEL`
`0 0 0 7 9 8 1 4 0`	`IGE`
`5 9 8 0 4 0 2 3 7`	
`6 3 0 2 7 9 5 8 0`	
`8 4 7 0 5 6 9 2 3`	
`0 2 5 0 8 4 0 0 1`	
`4 8 6 9 1 0 3 5 2`	
`2 1 9 8 3 5 0 7 6`	
`0 5 3 4 0 2 8 0 0`	
`0 0 8 0 5 0 4 7 0`	
`6 1 5 2 0 0 9 8 0`	
`0 3 0 9 0 0 5 0 2`	
`0 0 0 6 7 1 0 9 0`	
`4 6 0 8 9 2 0 0 0`	
`9 0 0 0 0 0 0 0 1`	
`1 0 2 0 8 9 7 6 5`	
`8 0 6 5 1 4 2 3 9`	
`0 0 9 7 2 0 1 4 0`	
`1 7 4 5 2 3 0 9 8`	
`3 0 2 7 9 0 1 4 5`	
`5 9 8 0 4 0 2 3 7`	
`6 3 0 0 7 9 5 8 4`	
`8 4 7 0 5 6 9 0 3`	
`9 2 5 3 8 4 7 6 0`	
`4 8 6 9 0 0 3 5 2`	
`2 1 9 0 3 5 4 7 6`	
`0 5 3 4 0 0 8 1 9`	

Problem D. Card collection

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1s

In an online game, a player can collect different types of power cards. Each power card can enable a player to have a unique game magic. There are m power cards available in the game as $(P_1, \cdots, P_m)$. A power card can be acquired by game points or through trading with others. In order to support the trading easier, a platform has been built. The platform charges a fixed amount $C_{i,j}$ game points for trading respective power cards, P_i and P_j. Note. Trading P_i to P_j or P_j to P_i would be of the same charge.

Write a program to calculate the minimal number of game points with a given original power card (P_o) to a target one (P_t). The output of your program should be the minimal game point value.

Input

The test data may contain many test cases. Each test case contains three data sections. The first section is an integer to indicate the number of power card types, denoted by m $(1 < m \leq 50)$. The second section contains two integers representing the type o $(0 < o \leq m)$ of the original power card P_o, and the type t $(0 < t \leq m)$ of the target power card. Also, o cannot be the same as t. The third section has a set of triplets and each triplet contains two cards types i, j and the charge amount $c_{i,j}$ $(0 < c_{i,j} \leq 20)$ between 2 types of power cards (P_i, P_j). The end part of section 3 contains a single 0.

Output

The output for each test case is the minimal number of game points needed for the trading of the original power card P_o to the target power card P_t.

Example

standard input	standard output

```
5
2 4
1 2 1
2 3 4
5 4 2
3 4 1
2 5 2
0
7
6 7
1 2 4
1 3 2
1 6 1
2 7 1
3 4 2
4 7 1
4 5 1
5 6 2
0
```
	```
	4
	4
	```

Problem E. Base Station Sites

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1s

5G is the proposed next telecommunications standards beyond the current 4G standards. 5G planning aims at higher capacity than current 4G, allowing a higher density of mobile broadband users, and supporting device-to-device, reliable, and massive wireless communications. A telecommunication company would like to install more base stations to provide better communication for customers. Due to the installation cost and available locations, the company can only install S $(2 \leq S \leq L)$ base stations at L $(2 \leq L \leq 100,000)$ candidate locations. Since the base stations work in the same frequency band, they will interfere and cause severe performance degradation. To provide high quality communication experience to customers, the company would like to maximize the distance between the base stations so as to reduce the wireless interference among the base stations. Suppose the L candidate locations are in a straight line at locations $P_1, P_2, \cdots, P_L$ $(0 \leq P_i \leq 1,000,000)$ and the company wants to install S base stations at the L candidate locations. What is the largest minimum distance among the S base stations?

Input

The input data includes multiple test sets.

Each set starts with a line which specifies L (i.e., the number of candidate locations) and S (i.e., the number of base stations). The next line contains L space-separated integers which represent $P_1, P_2, \cdots, P_L$.

The input data ends "0 0".

Output

For each set, you need to output a single line which should be the largest minimum distance among the base stations.

Example

standard input	standard output

`5 3`
`2 3 9 6 11`
`4 3`
`1 4 9 10`
`0 0`
	`4`
	`3`

For the first set, the 3 base stations can be installed at locations $2, 6, 11$.

Problem F. Nearby Bicycles

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1s

With fast developments of information and communication technology, many cities today have established bicycle sharing systems. The key component of the system is to provide information on nearby bicycles to potential users.

Consider m bicycles and n customers, where each bicycle is located at coordinate (c_j, d_j) for $j = 1, 2, \cdots, m$, and each user i is located at coordinate (a_i, b_i) for $i = 1, 2, \cdots, n$. The distance between two coordinates (x, y) and (x', y') is measured by $\sqrt{(x - x')^2 + (y - y')^2}$. For each user $i = 1, 2, \cdots, n$, you are given a threshold s_i, your task is to return the total number of bicycles that are within a distance of s_i from user i.

Input

The test data may contain many test cases. Each test case contains four lines. The first line of each case contains two integers, m and n $(0 < m, n \leq 1000)$. The second line contains the coordinates, $(c_1, d_1), (c_2, d_2), \cdots, (c_m, d_m)$, of bicycles $1, 2, \cdots, m$, respectively, which are separated by a space. The third line contains the coordinates, $(a_1, b_1), (a_2, b_2), \cdots, (a_n, b_n)$, of users $1, 2, \cdots, n$, respectively, which are separated by a space. The fourth line contains the thresholds, $s_1, s_2, \cdots, s_n$, of the n users. The last test case is followed by a line of two 0s. All the number of coordinate in the input is in the range $[-100000, 100000]$.

Output

The output for each test case contains a line of n integers, $k_1, k_2, \cdots, k_n$, where each k_i represents the total number of bicycles that are within a distance of s_i from user i, for $i = 1, 2, \cdots, n$.

Example

standard input	standard output

`4 2`
`(0,0) (0,1) (1,0) (1,1)`
`(0,0) (1,1)`
`1 1`
`0 0`
	`3 3`

Problem G. Optimal Coin Change

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1s

In a 10-dollar shop, everything is worthy 10 dollars or less. In order to serve customers more effectively at the cashier, change needs to be provided in the minimum number of coins.

In this problem, you are going to provide a given value of the change in different coins. Write a program to calculate the number of coins needed for each type of coin.

The input includes a value v, a size of the coinage set n, and a face value of each coin, $f_1, f_2, \cdots, f_n$. The output is a list of numbers, namely, $c_1, \cdots, c_n$, indicating the number of coins needed for each type of coin. There may be many ways for the change. The value v is an integer satisfying $0 < v \leq 2000$, representing the change required in cents. The face value of a coin is less than or equal to 10000. The output of your program should take the combination with the least number of coins needed.

For example, the Hong Kong coinage issued by the Hong Kong Monetary Authority consists of 10 cents, 20 cents, 50 cents, 1 dollar, 2 dollars, 5 dollars and 10 dollars would be represented in the input by $n = 7$, $f_1 = 10$, $f_2 = 20$, $f_3 = 50$, $f_4 = 100$, $f_5 = 200$, $f_6 = 500$, $f_7 = 1000$.

Input

The test data may contain many test cases, please process it to the end of the file.

Each test case contains integers $v, n, f_1, \cdots, f_n$ in a line. It is guaranteed that $n \leq 10$ and $0 < f_1 < f_2 < \cdots < f_n$.

Output

The output be n numbers in a line, separated by space. If there is no possible change, your output should be a single -1. If there are more than one possible solutions, your program should output the one that uses more coins of a lower face value.

Example

standard input	standard output
`2000 7 10 20 50 100 200 500 1000`	`0 0 0 0 0 0 2`
`250 4 10 20 125 150`	`0 0 2 0`
`35 4 10 20 125 150`	`-1`
`48 4 1 8 16 20`	`0 1 0 2`
`40 4 1 10 13 37`	`3 0 0 1`
`43 5 1 2 21 40 80`	`1 1 0 1 0`

Problem H. Sets

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1s

For a fixed number n, we define $S_1(n)$ as the set of integers $1, 2, \cdots, n$. And for any $i > 1$, we define $S_i(n)$ as a set containing all sums of two different numbers in $S_{i-1}(n)$, e.g. e.g., $S_i(n) = \{x+y : \exists x, y \text{ in } S_{i-1}(n) \text{ such that } x \neq y\}$.

For example, if $n = 3$, we have

$$\begin{aligned} S_1(n) &= \{1, 2, 3\}, \\ S_2(n) &= \{3, 4, 5\}, \\ S_3(n) &= \{7, 8, 9\}, \\ S_4(n) &= \{15, 16, 17\}. \end{aligned}$$

Then, we sort each set respectively, and combine them in order into a sequence L. In above case, we have the sequence $L = 1, 2, 3, 3, 4, 5, 7, 8, 9, 15, 16, 17, \cdots$.

Now, given integers n and k, what is the k-th number in sequence L?

Input

The input file contains several test cases, please handle it to the end of file.

For each case, there is only one line containing two integers n and k ($n \leq 1000, k \leq 10^{1000}$).

Output

For each case, output one integer indicating the k-th number of the sequence. Output -1 if it does not exist.

Example

standard input	standard output
4 6 2 20	4 -1

Problem I. Count the Even Integers

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1s

Yang Hui's Triangle is defined as follow.

In the first layer, there are two numbers $A_{1,1}$ and $A_{1,2}$ satisfying $A_{1,1} = A_{1,2} = 1$.

Then for each $i > 1$, the i-th layer contains $i + 1$ numbers satisfying $A_{i,1} = A_{i,i+1} = 1$ and $A_{i,j} = A_{i-1,j-1} + A_{i-1,j}$ for $1 < j \le i$.

1 1

1 2 1

1 3 3 1

1 4 6 4 1

1 5 10 10 5 1

1 6 15 20 15 6 1

1 7 21 35 35 21 7 1

1 8 28 56 70 56 28 8 1

Now, given an integer N, you are asked to count the number of even integers in the first N layers.

Input

The input file contains multiple cases, please handle it to the end of file.

For each case, there is only one line containing an integer N ($0 < N \le 10^{50}$).

Output

For each case, output the number of the even integers in the first N layers of Yang Hui's Triangle.

Example

standard input	standard output
4 8 12	4 16 42

Problem J. Triangle

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2s

Bob draws N black points and N white points in the plane, and draws directed edges between each pair of them.

Given a distance threshold D, for each pair of points p and q, Bob can draw an edge according to the following rules:

- If these two points share the same color, put an edge $p \to q$ or $q \to p$ optionally.
- If they have different colors, suppose that p is white and q is black, then draw the edge $p \to q$ if $dist(p, q) > D$, or $q \to p$ if not.

The distance function is defined as $dist(p, q) = |p.x - q.x| + |p.y - q.y|$, where $p.x$ and $q.x$ represent the x-coordinates of p and q, and $p.y$ and $q.y$ represent the y-coordinates of p and q.

Bob thinks that it a beautiful triangle of points p, q and r (which is a triple of three points) should satisfy the the following conditions:

- At least one of them is black, and at least one of them is white.
- $p \to q, q \to r$ and $r \to p$ are all the edges between them.

Now Bob wants to know the minimum number and the maximum number of beautiful triangles could exist.

Input

The input contains several test cases, please handle it to the end of file.

For each test case, the first line contains two integers N ($N \leq 100000$) and D. In the next N lines, each line contains two integers indicating the x-coordinate and the y-coordinate of a white point. In the next N lines, each line contains two integers indicating the x-coordinate and the y-coordinate of a black point. Some points may share the same coordinate.

All numbers in input are non-negative integers and less that 2^{31}.

Output

For each test case, output two integers indicating the minimum number and the maximum number of beautiful triangles.

Example

standard input	standard output

2 1
1 2
1 1
3 1
2 2
	0 2

Problem K. Marine

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2s

In a fantastic map of Starcraft, you are required to use one marine to defeat two zerglines. As you do not think whether this mission is possible or not, you have to simulate it with a program.

The map consists of 5×5 grid cells, and each grid cell can either be passable or not passable. The marine and zerglings are always in passable grid cells. Two zerglings may be in the same grid cell, but the marine can not be in the same grid cell with any alive zergling in any time. Initially, the health point (HP) of the marine is m, and the HPs of both zerglings are z.

The game runs in turns. In each turn, the game runs in three phases.

1. The marine moves by one grid cell in horizontal or vertical direction, or do not move to shoot to one zergling. Because the size of the map is small, the marine can shoot to any zergline in any position. The shooting in each turn reduces the HP of the target zergline by 1. If the HP of a zergline is equal to or less than 0, it dies.

2. All alive zerglines move at the same time. If a zergline is in the neighbouring grid cell of the marine, it will attack the marine, otherwise it will move by one grid cell in the shortest path to the marine. If there are multiple ways, it will choose with priority of left, up, right and down (e.g. if both going left and going up are in the shortest path, it will choose going left). If both zerglings in the same grid cell and attack the marine, the HP of the marine will be reduced by 1, otherwise each zergling attacks the marine and decreases the HP of the marine by 1. If the HP of the marine is equal to or less than 0, it dies.

3. The system checks the status of the marine and zerglings. If both zerglings die, you win. If the marine die, you lose. Furthermore, if you can not win the game in 34 turns, you lose the game.

You need to figure out whether if you can win the game or not. If yes, output the minimum turns in which you win the game.

Input

The input file might contains multiple cases, please handle it to the end of file. The first five lines of each case is a map. Each line contains five characters. The meaning of each character is described as follows:

- 1: a impassable grid cell;
- M: the marin;
- Z: one zergling;
- z: another zergling;
- other characters: passable grid cells.

In the six line, there are two integers $m, z (0 < m \leq 16, 0 < z \leq 99)$, as described above.

Output

For each case, if you can win, output 'WIN' in the first line and the minimum turns in the second line, otherwise output 'LOSE'.

Example

standard input	standard output
zZ000 11110 00M10 01110 00000 15 15	WIN 30

Remark: In the example above, the best strategy is that the marine does not move, and shoots to zerglings 'Z' first, and then shoots to 'z'. The zergling 'Z' moves to the neighbouring grid of the marine after turn 14 but it is killed in turn 15. The zergling ‘z’ moves to the neighbouring grid of the marine after turn 15. And then, the zergling ‘z’ and the marine attack each other in the next 15 turns. In turn 30, because the marine moves first, it kills zergline ‘z’ and has 1 HP remained.