Problem A. Awful Program

Input file:	awful.in
Output file:	awful.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

During his work at the Research Institute of Given Strings, Vasya faced huge problems, as the program for password hashing had a source in a language he didn't know.

```
[define[reversed_range n][if [< n 1] '[] [cons n [reversed_range [- n 1]]]]]
[define [range n] [reverse [reversed_range n]]]
; [range n] is a list of integers from 1 to n
[define [f l]
  [if [null? [cdr l]]
      [car l]
      [f [append [cdr[cdr l]] [list [car l]]]]
  ]
]
[define [hash n] [f [range n]]]
```

Vasya is in a great necessity of being able to calculate `[hash n]`. Help him to deal with *Awful* programming language. You may assume that the interpreter has enough stack memory to calculate `[hash n]` for every n.

Expression	*Result*	*Expression*	*Result*
`[- 10 1]`	9	`[list 1]`	`[1]`
`[car '[1 2 3]]`	1	`[cdr '[1]]`	`[]`
`[cdr '[1 2 3]]`	`[2 3]`	`[cons 1 '[2 3]]`	`[1 2 3]`
`[append '[1 2] '[3 4]]`	`[1 2 3 4]`	`[reverse[range 5]]`	`[5 4 3 2 1]`
`[cdr[cdr[range 5]]]`	`[3 4 5]`	`[if [< 0 1] 1 0]`	1
`[if [null? '[1]] 1 0]`	0	`[if [null? '[]] 1 0]`	1
`[range 5]`	`[1 2 3 4 5]`	`[reversed_range 5]`	`[5 4 3 2 1]`

Input

The input consists of no more than a thousand test cases. Each test case consists of a single line with a single positive integer n ($1 \leqslant n \leqslant 10^{100}$). Input is terminated with a test case where $n = 0$. This test case should not be processed.

Output

For each test case, write a single line with a single integer — the value of `[hash n]`.

Examples

awful.in	awful.out
1 2 3 0	1 1 3

Problem B. Count Solutions

Input file:	count.in
Output file:	count.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

Vasya got an assignment to count the number of non-negative integer solutions of the equation:

$$x_1 + 2x_2 + 3x_3 + \ldots + nx_n = n$$

Vasya composed a table for small values of n:

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Number of solutions	1	2	3	5	7	11	15	22	30	42	56	77	101	135	176

Help him to calculate the number of solutions for large values of n. The answer should be output modulo m.

Input

The input consists of no more than 10 000 test cases. Each test case consists of a single string with two integer numbers n and m ($1 \leqslant n \leqslant 50\,000$, $2 \leqslant m \leqslant 10^9$). Also, m will be square-free, i. e. there will be no such integer $a > 1$ that m is a multiple of a^2. You may also assume that the maximal prime divisor of m will not exceed 100. The input will be terminated with a test case where $n = m = 0$. This test case should not be processed.

Output

For each test case, write a single line with a single integer — the number of solutions modulo m.

Examples

count.in	count.out
1 3 4 3 15 210 0 0	1 2 176

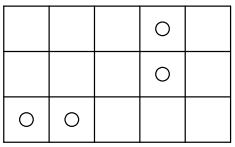
Problem C. Fleas and a Dog

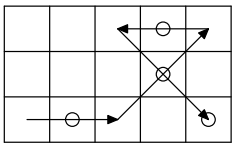
Input file:	jumps.in
Output file:	jumps.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

Vasya got a new assignment: he has to write a simulator for the game "Fleas and a Dog".

There are n fleas and one sleeping dog on an infinite chessboad. There can be more than one flea on a single cell. A move consists in choosing one flea and either moving it to a neighbouring cell or making one or several jumps with it.

Cells (x_1, y_1) и (x_2, y_2) are said to be neighbouring iff $\max(|x_1 - x_2|, |y_1 - y_2|) = 1$. A jump can be made from cell (x_1, y_1) to cell $(2x_2 - x_1, 2y_2 - y_1)$, if cell (x_2, y_2) is occupied and is a neighbour of (x_1, y_1), and cell $(2x_2 - x_1, 2y_2 - y_1)$ is free. See figure for examples of possible jumps.





The goal of the game is to reach the cell with the dog with one of the fleas.

Your task is to determine the minimal number of moves required to achieve the goal.

Input

The input consists of one or more test cases. Each test case starts with a single line containing three integer numbers n, x_d, y_d — the number of fleas and the position of the dog ($1 \leqslant n \leqslant 100\,000$). The following n lines contain pairs of integers — the coordinates of fleas. The input will be terminated with a test case with $n = x_d = y_d = 0$. This test case should not be processed.

Sum of n in the whole input will not exceed 100 000. All coordinates are integer numbers not exceeding 10^9 by absolute value.

Output

For each test case, write a single integer in a single line — the minimal number of moves required to achieve the goal.

Examples

jumps.in	jumps.out
4 0 -2 0 0 1 0 3 1 3 2 1 0 0 0 0 0 0 0	2 0

Problem D. Oriental Puzzle

Input file:	`oriental.in`
Output file:	`oriental.out`
Time limit:	1 second
Memory limit:	256 mebibytes

Vasya works for RIO (Research Institute of Oriental puzzles). He is studying a new puzzle consisting of several nodes and tubes partially filled with some liquid.

There are several nodes, plastic tubes connect some pairs of nodes. Each tube is made of a fabric that shines when the liquid passes through it. Two of the nodes are special: initially, first of them (source) contains all the liquid, and the other one (sink) is empty, but it should contain all the liquid after the puzzle will be solved (then one can swap their functions).

The goal is to arrange the tubes in such a way that after the liquid will be released from the source, all tubes will shine for some time (i. e. the liquid will be able to come through them), and finally, the liquid be collected in the sink.

Each tube should have some fixed orientation: one node will be higher, the other one will be lower. The liquid moves only from higher nodes to lower ones.

Help Vasya to solve that puzzle: find such orientation of tubes.

Input

The input consists of one or more test cases. Each test case consists of a single line with containing two integers n and k — the number of nodes and tubes ($2 \leqslant n \leqslant 10^5$, $0 \leqslant k \leqslant 10^5$). The following k lines contain descriptions of tubes: pair (a_i, b_i) means a tube connecting nodes a_i and b_i. The source will have number 1 and the sink will have number n. It is guaranteed that no tube connects a node with itself; nevertheless, there can be more than one tube between a pair of nodes.

The input will be terminated with a test case with $n = k = 0$ which should not be processed.

The total sums of n's and k's in all test cases will not exceed 100 000 each.

Output

For each test case, write a line with a word "`Yes`", if the orientation is possible and "`No`" otherwise. In the first case, write k more lines with descriptions of tubes in the same manner as in the input. The higher node in each description should go first.

Example

oriental.in	oriental.out
6 8 1 3 3 6 1 2 2 4 4 1 4 5 5 1 2 3 0 0	Yes 1 3 3 6 1 2 4 2 1 4 5 4 1 5 2 3

Problem E. Polygon

Input file:	polygon.in
Output file:	polygon.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

Vasya has a articulated polygon. It means that the polygon has hard edges that are able to rotate relatively to the vertices. Vasya wants to make the polygon circumscribed. A polygon is called circumscribed if there is a circumference touching every edge of the polygon. A touch in a vertex is forbidden.

Input

The input contains one or more test cases. Each test case consists of a single line with integer numbers. The first number is the number $3 \leqslant N \leqslant 1000$ — the number of edges. The following N numbers describe lengths of the edges in clockwise order. The lengths do not exceed 100.

The input is terminated with a test case where $N = 0$. This test case should not be processed.

Sum of N's in the whole input does not exceed 100 000.

Output

For each test case, write a line with either "YES" if Vasya can manage to achieve his goal, or "NO" otherwise. If the answer is "YES", write two more lines: one should contain the radius of the circumference, other should contain $2N$ real numbers — coordinates of polygon vertices. Vertices should be listed in clockwise order. The first edge from the input should correspond to the vertices 1 and 2. The centre of the circumference should be in $(0, 0)$. If the answer is "NO", leave two next lines empty.

Your answer will be considered correct if the lengths of edges will differ from the given ones by no more than 10^{-6}.

Examples

polygon.in	polygon.out
4 1 1 1 2 4 1 1 1 1 0	NO YES 0.5 -0.5 -0.5 -0.5 0.5 0.5 0.5 0.5 -0.5

Problem F. Polynomial Hash

Input file:	polyhash.in
Output file:	polyhash.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

Vasya uses polynomial hashing. Given a string of characters $s_0 s_1 s_2 \ldots s_n$, its hash can be calculated as $h = (\text{ord}(s_0) + \text{ord}(s_1)p + \text{ord}(s_2)p^2 + \ldots + \text{ord}(s_n)p^n) \bmod q$, where p and q are distinct primes and $\text{ord}(c)$ means ASCII code of character c.

Given the value of hash, find any string having this hash. The string should consist of lowercase Latin letters (ASCII codes 97–122).

Input

Input consists of a single line containing three integers q, p and h ($3 \leqslant q \leqslant 10^7$, $0 \leqslant h < q$, $2 \leqslant p < q$, $2 \leqslant p \leqslant 10^6$). It is guaranteed that p and q will be prime.

Output

Write any string of length not exceeding 10^6 that has hash value of h.

Examples

polyhash.in	polyhash.out
533009 239 244670	spbsuchampionship
53309 239 6636	abacaba

Problem G. Setting up the Monument

Input file:	`roll.in`
Output file:	`roll.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

Head of Research Institute of Given Strings decided to set up a monument in the lobby. Vasya was assigned to estimate the budget.

The monument is a massive ball. The lobby is a rectangle $h \times w$. Initially, the ball is in the lower-left corner of the lobby, i. e. in the cell $(1, 1)$. The goal is to place it at cell $(h - 1, w - 1)$. Loaders can push the ball either to the top or to the right. After that, the ball starts movement until it meets some barrier. To place the ball to its designated place, loaders should build some temporary walls in the lobby. The prices of placing additional walls in different cells differ.

For each cell, Vasya knows the price of placing a wall in that cell. Help him to determine the minimal price of moving the ball to its destination.

Input

The input consists of one or more test cases. Each test case starts with a line containing two integers h and w ($3 \leqslant h, w \leqslant 100$). The following h lines contain the matrix of prices. The prices are given as integers from 0 to 10^9; negative integers are placed in cells where it's impossible to build a wall. You may assume that it's impossible to build a wall in $(1, 1)$.

The input will be terminated with a test case where $h = w = 0$. This test case should not be processed.

The sum of h's and w's in the whole input does not exceed 1000.

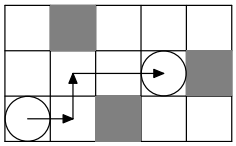
Output

For each test case, write a single line. It should contain either the minimal price or the message "`Mission impossible`" if the destination is unreachable.

Examples

roll.in	roll.out
3 5 5 1 5 5 5 5 5 5 5 1 -1 5 1 5 5 3 3 -1 -1 -1 -1 -1 -1 -1 -1 -1 0 0	3 Mission impossible

The figure shows an optimal way of placing walls in the first sample test.



Problem H. Christmas Tree

Input file:	spruce.in
Output file:	spruce.out
Time limit:	1 second
Memory limit:	256 mebibytes

People of Manersk city are in preparation of celebrating the New Year. A big and beautiful Christmas Tree should be installed on the main square. The task of picking a tree was given to Vasya, a famous researcher from some research institute.

Forest warden Andrey knows the algorithm Vasya uses to pick a tree. Surely, he wants to find the highest spruce in the forest. However, the forest is large, and Vasya does not want to get lost in it. So before going to the forest, he will come to Andrey and ask which spruce should he pick. Then, Vasya will climb up that spruce and look around. From a spruce h meters high, he will see every tree in the square $2h$ meters $\times$ $2h$ meters centered at that spruce.

If Vasya sees a spruce higher than the one he just climbed, he will go to the highest of such spruces and climb up again. Vasya will continue doing so until he climbs up a spruce from which he can see no higher spruce. After that, he will climb down, cut that spruce down and carry it to the main square.

Forest warden Andrey knows everything about the forest and the spruces. In particular, he knows the coordinates and height of each of the n spruces in the forest. Help him save the forest, that is, show Vasya such spruce that Vasya will cut a spruce of height as small as possible.

Input

Input consists of one or more test cases. The first line of each test case contains n, the number of spruces in the forest. Next n lines contain three numbers each; i-th of these lines contains integers x_i, y_i and h_i — the coordinates and height of i-th spruce. All these integers do not exceed 10^9 by absolute value. There are no spruces of the same height in each particular test case.

Input is terminated by a line containing a single zero. The sum of n over all test cases does not exceed 50 000.

Output

For each test case, write two numbers on a single line: the number of spruce Andrey should show to Vasya and the height of the spruce which will be cut down as a result. In each test case, spruces are numbered from one to n in the order they are given in the input.

Example

spruce.in	spruce.out
3 -1 1 2 -1 -1 3 0 3 4 0	2 3

Problem I. New Movies Free Download

Input file:	`torrent.in`
Output file:	`torrent.out`
Time limit:	2 seconds
Memory limit:	256 mebibytes

Anton and friends decided to watch a good old movie once more. Oh no! Old DVD disk is so scratched that it's unreadable. Time to download the movie over Internet.

It is well-known that transferring files over torrent protocol (it's called BitTorrent, same as the original client software) works in the following way: file is divided into segments, and each segment has to be downloaded separately. Different segments could be downloaded from different places to improve overall download speed.

Anton knows the size of the segments into which the file with the movie is divided. He also knows the download speed and playback speed of that file: one segment takes t_d seconds to download and t_v seconds to watch. Anton knows quite well the internals of many network protocols, so he just happens to know in advance the order in which segments will be downloaded. He also noted a particular feature of his BitTorrent client: it downloads each segment into RAM, and only when a segment is fully downloaded, it is saved to the hard drive where it can be then used by the media player.

Anton now wants to know the earliest moment when he can start watching the movie. Obviously, it can happen that at such moment, the film is not yet fully downloaded, but by the time Anton and his friends watch the first few available segments, other ones are already saved to the hard drive.

Input

The first line of input contains three integers n, t_d and t_v ($1 \leqslant n \leqslant 100\,000$, $1 \leqslant t_d, t_v \leqslant 10^9$). The next line contains a permutation of n numbers which denotes the order in which segments are downloaded.

Output

Write one number to the output — the time in seconds from start of the download to the earliest moment Anton and his friends can start watching the movie.

Examples

torrent.in	torrent.out
3 10 20 1 3 2	10
3 10 20 2 3 1	30
3 10 15 1 3 2	15

Problem J. Photocopy

Input file:	xerox.in
Output file:	xerox.out
Time limit:	2 seconds
Memory limit:	256 mebibytes

Research Institute of Given Strings (RIGS) is in preparation of celebrating the New Year. Vasya is honored with the task of hanging a congratulation banner in the main hall. For historical reasons, the banner should use a rather uncommon font. Fortunately, Vasya has a photohraph of the banner RIGS had last year. So, he decided to make the new banner by photocopying some parts of the old banner and then glueing the copies together. More precisely, each photocopy should be made of a substring of the old banner which has length between k_1 and k_2 characters, inclusive; otherwise, the sizes of the letters will be too different. Each substring will then be photocopied to one page of size A4, and these pages will be glued together to form the new banner.

The problem is that photocopying one page of size A4 costs 2 roubles. Help Vasya to get the required banner by making the minimal possible number of photocopies. If required, the photocopied parts could overlap or even be the same parts of the string.

Input

The input consists of one or more test cases. Each test consists of three lines. The first of these lines contains the length of the old banner n and the length of the new banner m ($1 \leqslant n, m \leqslant 10\,000$) followed by integers k_1 and k_2 ($1 \leqslant k_1 \leqslant k_2 \leqslant n$). The second line contans the text of the old banner. The third line contains the text of the new banner. Each text consists of lowercase Latin letters. The total length of all banner texts in the input does not exceed 10 000 characters. Input terminates by a line with four zeroes; this line should not be processed.

Output

For each test case, write the partition which should be photocopied. Adhere to the format of sample output below. If the problem cannot be solved for a particular test case, write "IMPOSSIBLE" instead.

Examples

xerox.in
51 12 1 2 snovymgodomsnovymschastiemsprazdnikomdorogiekollegi pozdravlyaem 51 12 1 2 snovymgodomsnovymschastiemsprazdnikomdorogiekollegi happynewyear 0 0 0 0
xerox.out
Case #1: p|o|zd|ra|v|l|y|a|em Case #2: IMPOSSIBLE

Problem K. Coin Stacks (Division 2 Only!)

Input file:	coin.in
Output file:	coin.out
Time limit:	2 seconds
Memory limit:	256 Mebibytes

You have several stacks of identical coins. You wonder how you can combine them all into one big stack, using a sequence of the following two operations:

- Combine — For two stacks of exactly equal height, combine them together to form a single stack.
- Difference — For two stacks of unequal height, take out the difference and treat it as a new stack. The rest of the coins are combined into one single stack. In other words, if there are two piles with a and b coins each, where $a > b > 0$, replace them with piles of size $2b$ and $a - b$.

For example, if you have piles with sizes 1, 3, 4, you could perform the difference operation on the first two piles (getting piles with sizes 2, 2, 4), then combine the two piles with size 2, and finally combine the remaining two piles to get a single pile with size 8.

Input

The first line contains the integer $T \leqslant 100$, followed by the data for T test cases. Each test case begins with a line containing integer N , $2 \leqslant N \leqslant 200$, which represents the number of stacks. The next line contains N positive integers, separated by spaces, which represent the sizes of the stacks, such that the total number of coins equals 2^K for some integer $K \leqslant 30$.

Output

For each test case, give a correct sequence of no more than 10^4 steps that combine all the coins. For each step, put a pair of space-separated integers describing the sizes of the stacks on which you perform the operation. If the integers are equal, it is a combine operation; otherwise it is a difference operation. After the last operation of each test case, output two zeroes. If for some test case is impossible to combine coins in such a way, print $-1\ -1$ as last operation of this test case.

Example

coin.in	coin.out
1 3 1 3 4	1 3 2 2 4 4 0 0

Problem L. Martial arts (Division 2 Only!)

Input file:	martial.in
Output file:	martial.out
Time limit:	2 seconds
Memory limit:	256 Mebibytes

In martial arts athletics, the two finalists are determined by a single elimination tournament. The winner of the final wins the gold, the loser wins the silver.

To qualify for bronze medal tournament, a competitor must have lost to one of the finalists.

For each case, you shall output the winner of the tournament, the runner-up and the qualifiers for bronze medal tournament.

Input

The input are a sequence of not more than 50 test cases. Each test case begins with the integer N . When N is 0, that is the signal that the input has ended and this line never should be processed. Otherwise, $2 \leqslant N \leqslant 6$. Next come $2^N - 1$ lines with pairs of names separated by whitespace, i.e., $name_1$, $name_2$.

Names are composed of alphabetic characters only, and will never exceed 10 characters. The meaning of each line of input is that $name_1$ defeats $name_2$.

The data always describe a perfect single elimination tournament bracket, in which half of the competitors are eliminated in round 1, half of those remaining are eliminated in round 2, and so forth, until a single winner remains. There are always exactly 2^N distinct names.

Output

For each case, output the winner of the `'Gold: '`, the `'Silver: '` and the qualifiers for the `'Bronze round: '`, in a space-separated, alphabetical ordered list.

Example

martial.in	martial.out
3 A aB CC D F E H G A CC H F A H 0	Gold: A Silver: H Bronze Round: CC F G aB

Problem M. Paperboy (Division 2 Only!)

Input file:	paperboy.in
Output file:	paperboy.out
Time limit:	2 seconds
Memory limit:	256 Mebibytes

A paperboy wants you to help him optimize his delivery route, which is on one side of a [very] long, linear street. He spends time delivering papers, and time moving between houses. The time to deliver a paper is 1.

The time to move between a pair of houses equals the difference in the house numbers multiplied by the number of papers he carries plus 1. The paperboy is allowed to carry as many or as few papers as he wishes, picking up or leaving papers behind as he sees fit. The papers are dropped off in front of house number x. What's the smallest amount of time it will take him to deliver the papers?

Input

The first line contains the positive integer $T \leqslant 60$, followed by the data for T test cases. Each test case is composed only of positive integers, one test case per line. Each line will begin with the number $n < 100$, the number of houses, followed by $n + 1$ house numbers. The first n of these are where the paperboy delivers his papers, the last house number is where all n papers are dropped off. (This is the house where he starts his paper route.) House numbers never exceed 10^5. The input ends on EOF.

Output

For each test case, output the minimum amount of time it takes for the paperboy to finish his route.

Example

paperboy.in	paperboy.out
2 3 10 20 30 10 4 10 20 40 80 30	53 184

Problem N. Qualifiers (Division 2 Only!)

Input file:	qual.in
Output file:	qual.out
Time limit:	2 seconds
Memory limit:	256 Mebibytes

In track athletics, the finals are determined by two semifinal heats. The runners with the best 3 times from each heat advance to the finals, as well as the best 2 times of all nonqualifiers.

Your job is to write a program that displays the qualifiers from a pair of semi-final heats.

Input

The input begins with a positive integer $T \leqslant 1000$ — the number of test cases, on a line by itself. T cases follow.

Each test case is composed of two lines with 8 numbers on each line. Each line represents one of the semi-finals. Each number is between 10.0 and 55.0, and is presented with no more than 3 decimal places of precision. All numbers are distinct from the others.

Output

For each case, output the race number, followed by the two lines of times in the order in which they were input, displayed to three decimals. Each qualifying time will be prefixed by a -.

Example

qual.in	qual.out
2 10 12 14 16 18 20 22 24 11 13 15 17 19 21 23 25 10.0 10.1 10.6 10.9 11.5 11.8 12.5 22.7 10.2 10.3 10.4 10.5 10.7 10.8 11.0 12.2	1 -10.000 -12.000 -14.000 -16.000 18.000 20.000 22.000 24.000 -11.000 -13.000 -15.000 -17.000 19.000 21.000 23.000 25.000 2 -10.000 -10.100 -10.600 10.900 11.500 11.800 12.500 22.700 -10.200 -10.300 -10.400 -10.500 -10.700 10.800 11.000 12.200

Problem O. Splitting (Division 2 Only!)

Input file:	split.in
Output file:	split.out
Time limit:	2 seconds
Memory limit:	256 Mebibytes

There are 3 camps of fans of FC «Spartak» and 3 camps of fans of FC CSKA standing at distinct points of a very large field. To prevent a conflict between fans, is decided to build a straight line wall that separates them.

Input

The first line contains the integer $T \leqslant 1000$, followed by the data for T test cases. The first line of each test case contains $x_1, y_1, x_2, y_2, x_3, y_3$ the coordinates of the Spartak fan camps, and the second (and last) line $x_4, y_4, x_5, y_5, x_6, y_6$ the coordinates of CSKA fan camps. All coordinates are integers between 1000 and -1000, and no three camps are collinear.

Output

Output `Yes` if a wall can be built to separate them and `No` if no such wall exists.

Example

split.in	split.out
2 1 1 2 2 2 3 1 0 5 1 4 5 1 0 -2 1 -2 -1 -1 0 2 1 2 -1	Yes No