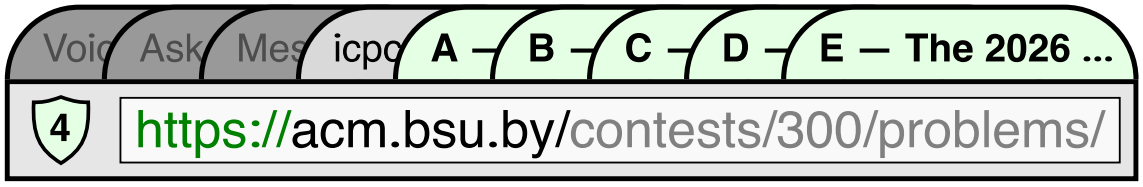
Problem A. Additional Tabs

Time limit: 1 second
Memory limit: 1024 megabytes



Alice has n tabs open in a browser. Exactly k of them are active, and the remaining $n - k$ tabs are in the background. Each tab consumes RAM: an active tab consumes exactly a megabytes, and a background tab consumes exactly b megabytes.

Alice wants to open ℓ more tabs. Each new tab is active and also requires a megabytes of memory. The previously existing tabs remain in their initial state.

The browser is allocated m megabytes of RAM in total, and it cannot use more memory than that. Determine whether the allocated memory is enough for all $n + \ell$ tabs to stay open at the same time.

Input

The only line contains six integers n, k, a, b, ℓ, m ($0 \le k \le n \le 10^4$, $0 \le a, b \le 10^4$, $0 \le \ell \le 10^4$, $0 \le m \le 2 \cdot 10^8$).

It is guaranteed that, before opening new tabs, the allocated memory was enough to fit all the already existing tabs.

Output

Print `YES` if the allocated memory is enough; otherwise, print `NO`. You may print the words in any case.

Examples

standard input	standard output
3 2 10 5 1 100	YES
5 5 2 100 0 10	YES
0 0 7 1 3 20	NO
4 1 10 67 5 340	YES

Note

In the first example, Alice has 2 active tabs consuming 10 megabytes each, and 1 background tab consuming 5 megabytes. Adding 1 more active tab raises the usage to $(2 + 1) \cdot 10 + 1 \cdot 5 = 35$ megabytes, which is less than the 100 megabytes Alice has at her disposal.

In the second example, no new tabs are added, and the previously existing tabs consume exactly the amount of available memory.

In the third example, all tabs are new, and they do not fit into the available memory.

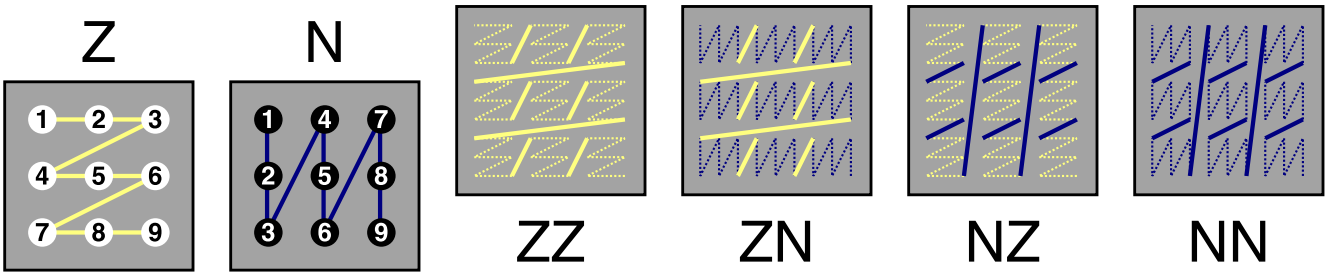
The fourth example is illustrated by the image above. The amount of available memory after subtracting previously opened tabs is 129 megabytes, so Alice can comfortably open 5 new tabs, but, for example, opening 13 new tabs would require Alice to close a few older tabs.

Problem B. Block Encoding

Time limit: 1 second
Memory limit: 1024 megabytes

For any matrix, we can define two ways to enumerate its elements in non-decreasing order of their indices: either the row number takes priority over the column number, or vice versa. Let us call these orders `Z` and `N`, respectively (taking inspiration from the shape of the polyline connecting the elements in the order).

You are given an $n \times m$ matrix that can be split into blocks of size $r \times c$ (in other words, it is a $\frac{n}{r} \times \frac{m}{c}$ matrix of blocks). We can define 4 encoding formats `ZZ`, `ZN`, `NZ`, and `NN` for it as follows: the first character defines the order of blocks in the enumeration, the second character defines the order of cells in a single block, and all cells of a block are enumerated sequentially. Refer to the image to obtain a clearer understanding using a 9×9 matrix example with 3×3 blocks.



Another, "n-dimensional" (`ND`) encoding format corresponds to `Z`-ordering and lists elements in the "usual" order: cells with a lower row number, or with an equal row number but a lower column number, go first.

You are given a block matrix stored as a 1-dimensional array in the order of format s. Reorder it to adhere to format f.

Input

The first line contains a single integer t ($1 \le t \le 10^4$): the number of test cases. Then, t test cases follow.

The first line of a test case contains four integers n, m, r, c ($1 \le r \le n \le 3 \cdot 10^5$, $1 \le c \le m \le 3 \cdot 10^5$, r divides n, c divides m): the dimensions of the matrix and the dimensions of the blocks.

The second line contains two formats s and f. Each of them is one of `ND`, `ZZ`, `ZN`, `NZ`, and `NN`.

The third line contains $n \cdot m$ integers $a_1, a_2, \ldots, a_{nm}$ ($-10^9 \le a_i \le 10^9$): the encoded matrix in format s.

It is guaranteed that the sum of $n \cdot m$ over all test cases does not exceed $3 \cdot 10^5$.

Output

For each test case, print $n \cdot m$ integers on a separate line: the elements of the same matrix in format f.

Example

standard input	standard output
3 4 3 2 1 ND ZZ 1 2 3 4 5 6 7 8 9 10 11 12 4 4 2 2 ZN ND 1 5 2 6 3 7 4 8 9 13 10 14 11 15 12 16 4 4 2 2 NZ NN 1 2 -7 -6 5 6 -3 -2 3 4 -5 -4 7 8 -1 0	1 4 2 5 3 6 7 10 8 11 9 12 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 1 -7 2 -6 5 -3 6 -2 3 -5 4 -4 7 -1 8 0

Problem C. Coloring of a Permutation

Time limit: 1 second
Memory limit: 1024 megabytes

For the purposes of this problem, a *permutation* of length n is an array of n **distinct** integers from 1 to n.

You are given a permutation $p = (p_1, p_2, \ldots, p_n)$ of the integers 1 through n.

You are to color its elements using the minimum possible number of colors so that there is no pair of indices (i, j) such that the following conditions hold at the same time:

- $i < j$;
- $p_i < p_j$;
- the i-th and the j-th elements have the same color.

If there are several valid colorings with the minimum possible number of colors, find any of them.

Input

The first line contains a single integer t ($1 \le t \le 10^4$): the number of test cases. Then, t test cases follow.

The first line of a test case contains a single integer n ($1 \le n \le 2 \cdot 10^5$): the length of the permutation.

The second line of a test case contains n integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$). It is guaranteed that each integer from 1 to n appears exactly once.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print two lines.

In the first line, print a single integer k ($1 \le k \le n$): the minimum possible number of colors.

In the second line, print n integers $c_1, c_2, \ldots, c_n$ ($1 \le c_i \le k$), where c_i is the color of the i-th element of the permutation. Each color from 1 to k must be used at least once. Any optimal valid coloring will be accepted.

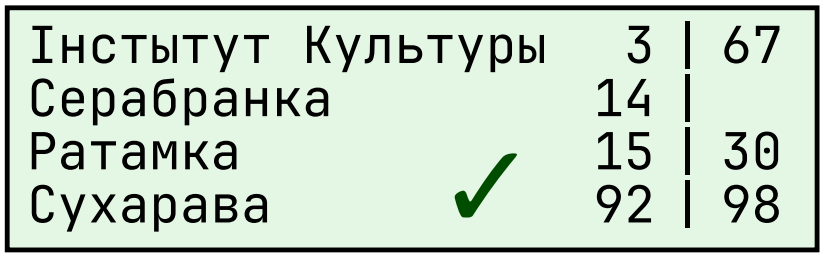
Example

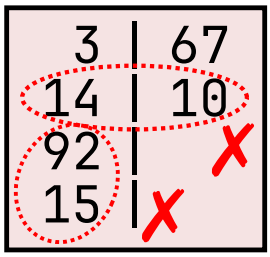
standard input	standard output
3 5 3 1 4 2 5 4 4 3 2 1 3 1 2 3	3 1 1 2 2 3 1 1 1 1 1 3 1 2 3

Problem D. Digital Display

Time limit: 3 seconds
Memory limit: 1024 megabytes

In Minsk, a bus stop is often equipped with a display showing how soon buses on certain routes will arrive:





The numeric part of the display consists of n rows and two columns. At any moment in time, the display shows upcoming buses of k ($0 \le k \le n$) different routes. In such a case, the first k rows are filled, and the remaining $n - k$ rows are completely empty.

The i-th filled row contains a positive integer a_i in the first column and may contain a positive integer b_i in the second column (otherwise, that cell is empty). It means that the time until the next bus of the described route is a_i minutes, and if b_i is present, the time until the subsequent bus is b_i minutes (otherwise, there is no subsequent bus). Of course, if b_i is present, then $b_i \ge a_i$.

The routes are placed in non-decreasing order of next bus waiting time, that is, $a_{i+1} \ge a_i$ for all suitable i. Note that the numbers in the second column are not necessarily sorted, nor are they necessarily present.

As columns have a fixed width of d digits, all present numbers have to be between 1 and $10^d - 1$, inclusive.

Your task, however, does not require knowing the complicated schedule of the transport system. Just find the number of distinct numeric parts of displays. Since the number can be large, find it modulo $10^9 + 7$.

Input

The only line contains two integers n and d ($1 \le n \le 10^{18}$, $1 \le d \le 6$).

Output

Print the number of distinct numeric parts of displays modulo $10^9 + 7$.

Examples

standard input	standard output
1 1	55
4 2	7155947
6767 4	886886693

Note

In the first example, there exist 1 empty table, 9 tables with one filled cell, and 45 tables with two filled cells. Thus, the answer is 55.

The illustration in the problem shows an example of a correct display (left) and an example of an incorrect display (right).

Problem E. Equivocal Logic

Time limit: 1 second
Memory limit: 1024 megabytes

One day, Edgar invented a new fancy method to quickly compute the roots of various degrees for a positive integer: take the sum of its digits, and subtract the degree of that root. Just look how easy the calculations are now: $\sqrt[2]{25} = 2 + 5 - 2 = 5$, $\sqrt[2]{196} = 1 + 9 + 6 - 2 = 14$, $\sqrt[3]{216} = 2 + 1 + 6 - 3 = 6$.

To Edgar's great disappointment, in reality, not only does his method not work, but also in multiple cases the true values of roots are not even integers. However, there are definitely a lot of roots that can be extracted that way.

Consider all pairs of positive integers (x, k), $k \geq 2$, such that the correct value of the k-th root of x can be obtained using Edgar's method, and arrange them in a list in ascending order of x (in case of equal x – in ascending order of k). Edgar wonders what the q-th pair in the list is, for various values of q. If for that pair $x \geq 10^{300}$, it is enough to say that the numbers are too large without finding them.

Input

The first line contains a single integer t ($1 \leq t \leq 10^3$): the number of test cases. Then, t test cases follow.

The only line of a test case contains a single integer q ($1 \leq q \leq 10^9$), which means that you should find the q-th pair (x, k) in the list.

Output

For each test case, print two required integers x and k ($1 \leq x < 10^{300}$, $k \geq 2$). If x is not less than 10^{300}, print two integers `-1 -1` for that test case instead.

Example

standard input	standard output
5 1 2 5 6 1000000000	4 2 25 2 196 2 216 3 -1 -1

Problem F. Faulty Machine

Time limit: 2 seconds
Memory limit: 1024 megabytes

The oldest exhibit in the museum of the Foundation of Fabulous Transformations is a probabilistic machine containing n light bulbs. On a usual day, it is turned off.

When it turns on during the excursions, the following happens. Within the first second, each bulb lights up independently with probability $\frac{1}{2}$. After that, at the end of each second, a subset of exactly m light bulbs is chosen uniformly at random, and each light bulb from that subset is toggled between on and off states.

Unfortunately, the machine no longer works perfectly. When exactly k light bulbs are on at the end of a second (it can be the first second as well), a bug in the machine is triggered and the process stops.

Find the expected number of seconds after the start until that happens.

We can show that after the first second, it is known whether the process can theoretically stop or not. For the latter case, **assume that** the expected number of seconds after the start for the process to stop is 0.

Input

The only line contains three integers n, m, k ($1 \le n \le 2 \cdot 10^5$, $1 \le m \le \min(n, 10)$, $0 \le k \le n$, $|2k - n| \le \max(\frac{n}{500}, 10)$).

Output

Print a real number: the expected number of seconds before the process stops, under the assumptions from the statement.

The answer will be considered correct if its absolute or relative error does not exceed 10^{-6}.

Formally, let your answer be x, and the jury's answer be y. Your answer is accepted if and only if $\frac{|x-y|}{\max(1,|y|)} \le 10^{-6}$.

Examples

standard input	standard output
3 2 1	0.625000000000
19 5 8	6.722779050519

Note

Consider the first example.

If 0 or 2 light bulbs are on initially, the process will go on forever (this has probability $\frac{1}{2}$).

If 1 light bulb is on initially, the process stops after 1 second (this has probability $\frac{3}{8}$).

If 3 light bulbs are on initially, it will change to 1 on the next second, so the process stops after 2 seconds (this has probability $\frac{1}{8}$).

Therefore, the answer is $0 \cdot \frac{1}{2} + 1 \cdot \frac{3}{8} + 2 \cdot \frac{1}{8} = \frac{5}{8}$.

Problem G. Going Along the Circle

Time limit: 1 second
Memory limit: 1024 megabytes

For the purposes of this problem, a *permutation* of length n is an array of n **distinct** integers from 1 to n.

A permutation of the integers from 1 to n is written down at n positions around a circle. From any position, you can go in one step to a neighboring position in the clockwise direction.

Start at the number 1 and go **only** clockwise: first to the number 2, then to the number 3, and so on up to the number n. The *length* of the walk from i to $i+1$ is the number of clockwise steps needed to go from the position of number i to the position of number $i+1$. Find the minimum and the maximum among these $n-1$ lengths.

Input

The first line contains a single integer t ($1 \le t \le 10^4$): the number of test cases. Then, t test cases follow.

The first line of a test case contains a single integer n ($3 \le n \le 2 \cdot 10^5$): the number of positions on the circle.

The second line of a test case contains n distinct integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$): the integers at the positions in clockwise order. These integers form a permutation of the integers from 1 to n.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print two integers from 1 to $n-1$ on a separate line: the minimum walk length, then the maximum walk length.

Example

standard input	standard output
6 6 1 4 2 6 3 5 3 1 2 3 3 2 3 1 5 1 2 5 4 3 4 4 3 2 1 4 1 3 2 4	2 4 1 1 1 1 1 4 3 3 2 3

Note

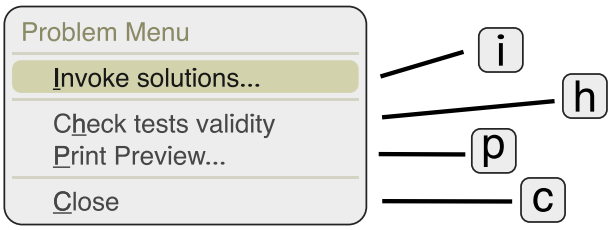
In the first test case of the example, $n = 6$ and the integers 1, 4, 2, 6, 3, 5 are written down clockwise. The lengths of the walks from 1 to 2, from 2 to 3, from 3 to 4, from 4 to 5, and from 5 to 6 are 2, 2, 3, 4, and 4, respectively, so the minimum is 2 and the maximum is 4.

Problem H. Hotkeys

Time limit: 2 seconds
Memory limit: 1024 megabytes

As an experienced programming competition organizer, Heinz knows a lot about bugs in related systems. That is why he decided to make a new tool for creating programming problems, *Hyperbola v0.01*!

One of the frequent menus in *Hyperbola v0.01* contains several items. It is very important to make sure users can navigate through that menu using single key presses. Therefore, for each item of the menu, Heinz wants to pick a distinct character contained in the name (its *hotkey*), so that pressing that key would invoke the corresponding item.



Heinz only values lowercase English letters and digits, so for the purpose of the algorithm he preemptively converted all uppercase English letters to lower case and removed all other possible characters from names.

Suppose that we pick the p_i-th character of the i-th item as a hotkey. Heinz wants to minimize the sum of p_i. Find such an arrangement of hotkeys, or report that this is impossible.

Input

The first line contains a single integer t ($1 \le t \le 10^4$): the number of test cases. Then, t test cases follow.

The first line of a test case contains a single integer n ($1 \le n \le 30$).

The next n lines contain strings $s_1, s_2, \ldots, s_n$ ($1 \le |s_i| \le 10^5$) of lowercase English letters and digits, denoting the menu items. It is guaranteed that the sum of $|s_i|$ over all test cases does not exceed $5 \cdot 10^5$.

Output

For each test case, print `NO` if it is impossible to assign pairwise distinct hotkeys. Otherwise, print `YES`, and in the next line, print n integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le |s_i|$): the positions of the chosen hotkeys in $s_1, s_2, \ldots, s_n$ that achieve the minimum sum.

If there are several valid sets of positions, print any. You may print the words in any case.

Example

standard input	standard output
3 4 invokesolutions checktestsvalidity printpreview close 2 a4a aa4 2 aa aa	YES 1 2 1 1 YES 2 1 NO

Problem I. Intersecting Spirals

Time limit: 1 second
Memory limit: 1024 megabytes

You are given several pairwise distinct Archimedean spirals on a two-dimensional plane. All the spirals start at the origin and leave it along the vector $(1, 0)$.

The i-th spiral is characterized by a nonzero integer d_i: it increases its distance to the origin by $|d_i|$ while making a full revolution, and it winds counterclockwise if $d_i > 0$ and clockwise if $d_i < 0$.

Formally, a spiral with parameter d consists of the points $\left(r \cos \frac{2\pi r}{d},\ r \sin \frac{2\pi r}{d}\right)$ for all $r \geq 0$.

Find the number of pairwise intersections of these spirals at a distance of at most R from the origin. If an intersection point belongs to more than two spirals, it is counted multiple times: once per each unordered pair of these spirals. Note that the origin is an intersection point of all given spirals.

Input

The first line contains two integers n and R ($1 \leq n \leq 2 \cdot 10^5$, $1 \leq R \leq 10^9$): the number of spirals and the maximum distance from the origin.

The second line contains n integers $d_1, d_2, \ldots, d_n$ ($-10^9 \leq d_i \leq 10^9$, $d_i \neq 0$): the parameters of the spirals.

It is guaranteed that $d_1, d_2, \ldots, d_n$ are pairwise distinct.

Output

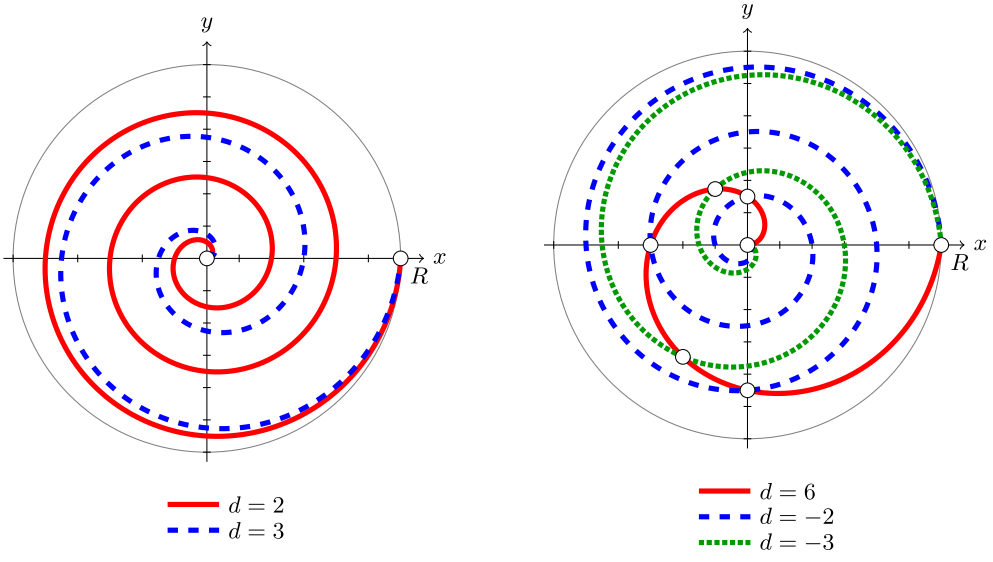
Print the number of pairwise spiral intersections located within distance R from the origin.

Examples

standard input	standard output
2 6 2 3	2
3 6 6 -2 -3	11

Note

Illustrations for both examples (note that some points may be counted several times):



Problem J. Jiggly Sequence

Time limit: 2 seconds
Memory limit: 1024 megabytes

You are given an integer $k \geq 1$. A sequence of n numbers $s_1, s_2, \ldots, s_n$ is called *jiggly* if, for any indices i, j such that $1 \leq |i - j| \leq k$, the condition $s_i \neq s_j$ is satisfied.

You are also given an array A of n integers $a_1, a_2, \ldots, a_n$. Find any way to permute the elements of A so that it becomes jiggly. Also, for every position i, find how many different values appear in the i-th position across all possible jiggly permutations of A.

Input

The first line contains two integers n and k ($1 \leq k \leq n \leq 3 \cdot 10^5$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq n$): the elements of the initial array A.

Output

If it is impossible to permute the array for it to become jiggly, print a single integer -1.

Otherwise, print two lines. The first line should contain n integers $b_1, b_2, \ldots, b_n$ ($1 \leq b_i \leq n$): the elements of any valid jiggly permutation of the initial array.

The second line should contain n integers $c_1, c_2, \ldots, c_n$ ($1 \leq c_i \leq n$), meaning that if we consider all possible jiggly permutations of the array A, there are exactly c_i different values in position i in them.

Examples

standard input	standard output
7 2 3 2 1 1 1 2 3	1 2 3 1 2 3 1 1 2 2 1 2 2 1
4 1 4 4 4 2	-1

Note

In the first example, the only valid permutations are $(1, 2, 3, 1, 2, 3, 1)$ and $(1, 3, 2, 1, 3, 2, 1)$.

In the second example, it is impossible to permute the array in such a way that no two 4-s stand next to each other.

Problem K. Keep the Sum

Time limit: 1 second
Memory limit: 1024 megabytes

You are given a sequence of n non-negative integers $a_1, a_2, \ldots, a_n$. However, some of its elements may be missing.

Find a way to fill in all the missing elements so that, for any k consecutive elements, their sum equals s, or report that this is impossible.

Input

The first line contains a single integer t ($1 \le t \le 10^4$): the number of test cases. Then, t test cases follow.

The first line of a test case contains three integers n, k, s ($1 \le k \le n \le 2 \cdot 10^5$, $0 \le s \le 10^9$).

The second line of a test case contains n integers $a_1, a_2, \ldots, a_n$ ($-1 \le a_i \le 10^9$). If $a_i = -1$, the i-th element is missing; otherwise, a_i is the value of that element.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, print -1 if it is impossible to fill in all the missing elements properly; otherwise, print n integers $b_1, b_2, \ldots, b_n$ ($0 \le b_i \le 10^9$): the filled sequence. If $a_i \ne -1$ for some i, then $b_i = a_i$ must hold.

If there are several valid answers, print any. It can be shown that if a solution exists, then there also exists a solution in which all the numbers do not exceed 10^9.

Example

standard input	standard output
4 8 3 11 7 -1 -1 -1 -1 -1 -1 1 5 2 1 -1 -1 -1 -1 -1 6 3 30 10 10 10 10 10 10 2 1 1 1 2	7 1 3 7 1 3 7 1 1 0 1 0 1 10 10 10 10 10 10 -1

Note

In the first test case of the example, we can see that any $k = 3$ consecutive elements b_i, b_{i+1}, b_{i+2} of the filled sequence add up to $s = 11$.

In the second test case, the filled sequence $(0, 1, 0, 1, 0)$ can also be a valid answer.

In the third test case, all the elements are already filled in, which is a completely acceptable situation. Each $k = 3$ consecutive elements add up to $s = 30$, making the original sequence a valid answer.

In the fourth test case, the single element a_2 does not equal $s = 1$, so there are no valid answers.

Problem L. Leisure Geometry

Time limit: 1 second
Memory limit: 1024 megabytes

Have you ever been to a boring lesson with absolutely nothing to do? Everyone deals with it differently: some students prefer to train their art skills, some try to compensate for the lack of sleep, and Lucas is the only one in this class who does robotics instead.

Lucas has just programmed a robot. It draws a polyline on the 2D plane using a simple algorithm. Initially, the pencil controlled by the robot is located at the point $(0, 0)$ and faces right along the vector $(1, 0)$. Then, the following pair of actions is repeatedly performed:

- the robot draws a segment of length 1 and turns the direction clockwise by α degrees;
- the robot draws a segment of length 1 and turns the direction counterclockwise by β degrees.

If the pencil returns to $(0, 0)$ for the first time, and the resulting polyline does not intersect itself, the robot stops, and Lucas calls it a successful operation. In that case, Lucas wonders what the area of the figure bounded by the drawn polyline is.

We say that the polyline *intersects itself* if two of its segments have a common point other than a shared vertex of adjacent segments. Segments are adjacent if they are consecutive while drawing, and the first and the last segments are also adjacent if the polyline has returned to $(0, 0)$ for the first time.

Input

The first line contains a single integer t ($1 \le t \le 10^4$): the number of test cases. Then, t test cases follow.

The only line of a test case contains two integers α and β ($1 \le \alpha, \beta \le 179$).

Output

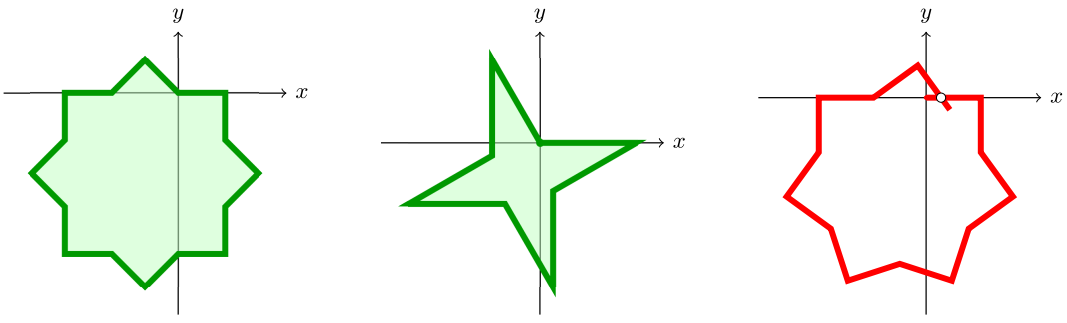
For each test case, print the area of the figure on a separate line if the robot succeeds, or -1 otherwise. If the printed value is not -1, the absolute or relative error must not exceed 10^{-6}.

Example

standard input	standard output
3 90 45 150 60 90 36	13.6568542495 1.2679491924 -1

Note

The figures below correspond to the example test cases (with possibly different scales of figures).



In the third test case of the example, the polyline intersects itself, and the figure is drawn only up to the first intersection.

Problem M. Multiply and Add

Time limit: 1 second
Memory limit: 1024 megabytes

You are given two non-negative integers m and n.

Find any triple of non-negative integers (a, b, c) such that:

- $0 \le a, b, c \le m$;
- $a \cdot b + c = n$.

Input

The first line contains a single integer t ($1 \le t \le 10^4$): the number of test cases. Then, t test cases follow.

The only line of a test case contains two integers m and n ($0 \le m, n \le 10^{18}$).

Output

For each test case, print `YES` in the first line if such a triple exists; otherwise, print `NO`. You may print the words in any case.

If a triple exists, print three required integers a, b, c in the second line for that test case. If there are several valid answers, print any.

Example

standard input	standard output
6 5 7 10 67 300 10000 1511 2026 20 9000 0 3	YES 2 2 3 YES 7 9 4 YES 200 50 0 YES 1511 1 515 NO NO

Note

In the first test case of the example, the triples $(1, 2, 5)$ and $(4, 1, 3)$ are also valid answers since $1 \cdot 2 + 5 = 4 \cdot 1 + 3 = 7 = n$. However, $(6, 1, 1)$ is not a valid answer because $a = 6 > 5 = m$.