Problem A. Friendly Rooks

Time limit: 1 second
Memory limit: 512 megabytes

Rook is a piece in the game of chess. It moves horizontally or vertically through any number of unoccupied squares, and can not jump over pieces. The rook can capture other piece if the piece is on the same vertical or horizontal line with the rook. There can be no more than one rook in one square of the chess board.

You are given k rooks and a chess board with of $n \times m$. You need place these rooks on the board so that they cannot capture each other.

Input

The only line of input contains three integers n, m and k — the lengths of the chess board sides and the number of rooks ($1 \le n, m, k \le 100$).

Output

If it is impossible to place k rooks on an $n \times m$ chess board, print the line `Impossible`.

If there is at least one correct placement, print `Possible`. Then output n lines of m characters each — the description of the placement of the rooks on the chess board. The j-th character of the i-th line must be "`*`" if the square (i, j) contains a rook, or "`.`" if the corresponding square in your placement is empty.

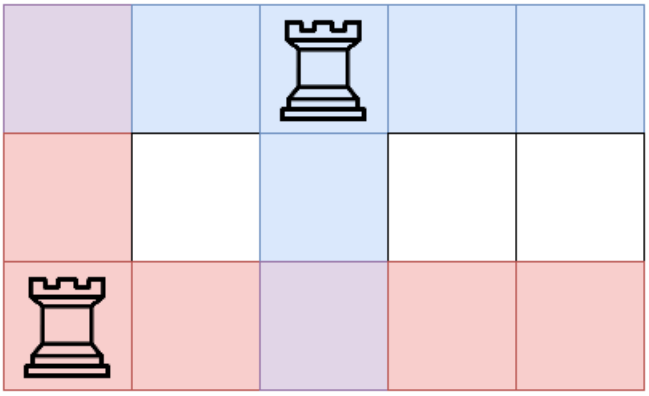
If there are several correct placements, you can output any of them.

Examples

standard input	standard output
1 2 1	Possible *.
3 3 100	Impossible
3 5 2	Possible ..*.. ..... *....

Note

Image of the rook placement for the third test case:



Red marks the squares that the lower left rook can capture, blue marks the squares that the upper right rook can capture, purple — squares that can be captured by both rooks.

Problem B. Guess the Array

Time limit:	2 seconds
Memory limit:	512 megabytes

This is an interactive problem. Your program will interact with the jury program using standard input and output.

Alice and Bob have decided to play a game called "Guess the Array". The rules of the game are very simple: Alice has an array of n integers, and Bob has to guess this array by making no more than n queries about the sums on the segments.

In one move Bob can make one of two types of queries to Alice:

1. "`? l r`" to find out the sum of numbers on the segment of the array from the l-th to the r-th element inclusive;
2. "`!`" to tell Alice that he is ready to give the answer. After this query Alice expects n integers from Bob: the initial array.

For each first type query, Alice tells Bob the sum of the numbers in the requested segment. But to make it harder to guess, after each query Alice makes one segment *blocked*. In further queries, Bob cannot ask for the sum of the numbers on the blocked segments.

Help Bob guess Alice's array by making no more than n first type queries.

Interaction Protocol

The first line of input contains an integer n — the size of Alice's array ($1 \leqslant n \leqslant 10^4$). It is guaranteed that all numbers in the array do not exceed 10^9 by their absolute values.

Next, a protocol for communicating with the jury program (interactor) is executed.

The interactor expects two types of requests from your program: "`? l r`" and "`!`", where l, r are integer boundaries of the segment that you want to know the sum at ($1 \leqslant l \leqslant r \leqslant n$). Each query must be followed by a line break. If your program does not follow the query format, your solution may get any verdict (other than `OK`).

After the first type of query, you must read three integers s, l_b and r_b from the standard input — the sum on the requested segment s and the boundaries of the new *blocked* segment $[l_b, r_b]$, that can no longer request a sum for ($|s| \leqslant 10^{18}$; $1 \leqslant l_b \leqslant r_b \leqslant n$).

The query of the second type means that your program is ready to give the answer. After the second type query, you should print n integer guessing the initial array.

Note that your program can make no more than n queries of the first type. If this limit is exceeded, or if you try to query the sum of the previously blocked segment, the interactor will print "`-1 -1 -1`" and will exit with the verdict `WA`. To avoid getting a `TL` or `IL` verdict, your program must terminate with zero exit code after reading "`-1 -1 -1`" from the input.

Example

standard input	standard output
3 1 2 2 3 2 3 6 1 2	 ? 1 1 ? 3 3 ? 1 3 ! 1 2 3

Problem C. Moving Cells

Time limit: 1 second
Memory limit: 512 megabytes

Little Alice has got a modern pixel picture for her birthday.

The picture is a rectangular grid of size $n \times m$. Each column of the grid has one or more consecutive cells colored black, all the other cells are colored white.

Alice considers the picture *beautiful* if there is a path between any two black cells u and v that runs only through the black cells, each time going from a cell to a side-adjacent cell — begin in the black cell u, then go to a side adjacent to u black cell w, then go to a side adjacent to w black cell, and so on, eventually reaching the black cell v.

Since the picture is modern, it can be changed. In one *action* you may select any column and move all black cells in that column one cell in the same direction — up or down. Cells can be moved only if they do not go outside the picture.

Alice wonders what is the minimum number of actions it would take to get a *beautiful* black picture.

Input

The first line of input has two integers n and m — the number of rows and the number of columns in the picture, respectively ($1 \le n, m \le 100\,000$). It is guaranteed that the total number of the picture cells does not exceed 10^6 ($1 \le n \cdot m \le 1\,000\,000$).

The next m lines contain two integers s_i and t_i each — the starting and the ending positions of black cells in the i-th column of the grid ($1 \le s_i \le t_i \le n$).

Output

Output a single integer — the minimum number of actions you need to make the given picture *beautiful*.

Example

standard input	standard output
9 3 1 2 4 5 7 9	4

Problem D. Army of Clones

Time limit: 1.5 seconds
Memory limit: 512 megabytes

The army of x clones sneaked into the spaceship "Death Star" to help Luke Skywalker battling with Darth Vader. The spaceship consists of n rooms and m bidirectional passages between them. The clones start in the room 1 and want to go to the room n, where Luke is.

However every room is guarded by droids, room i is guarded by a_i droids. When the clones appear in the room, the battle between them and the droids starts. If the number of clones is greater than the number of droids, the clones will kill all the droids and all the clones will stay alive. Otherwise the clones will kill all droids as well, but they will lose half of the army: if there are x clones at the beginning of the battle, then there will be $\lfloor \frac{x}{2} \rfloor$ clones at the end of battle, rounded down. The clones have to battle in all rooms they would visit, including rooms 1 and n.

Help the captain of the army to count the maximum number of clones that can come from the room 1 to the room n .

Input

The first line contains two integers n and m — number of rooms and passages in "Death Star" ($1 \le n, m \le 2 \cdot 10^5$).

The following m lines describe passages: the i-th passage is described by two integers u_i and v_i — the rooms that are connected by the passage ($1 \le u_i, v_i \le n$, $u_i \ne v_i$). It is guaranteed that every pair of rooms is connected by at most one passage.

The next line contains an integer x — the number of clones in the army ($1 \le x \le 10^9$).

The last line contains n integers $a_1, a_2, \ldots, a_n$ — the number of droids in the rooms ($1 \le a_i \le 10^9$).

Output

Print a single integer — the maximum number of clones that can go from room 1 to room n. If there is no path to follow, so that at least one clone survives, print 0.

Examples

standard input	standard output
4 4 1 2 1 3 2 4 3 4 7 10 2 3 1	3
4 4 1 2 1 3 2 4 3 4 7 10 3 3 1	0

Problem E. Haiku

Time limit:	1 second
Memory limit:	512 megabytes

Haiku — is a type of short form poetry originally from Japan. Traditional haiku consist of three phrases that contains 17 phonetic units. First 5 of them are on the first line, next 7 of them are on the second line, and the last 5 on the last line.

You have found a big text about haiku. However, there were no line breaks in it. You have already broken the text into words and now you want to find all potential haiku in it: segments of consecutive words that can form a haiku.

For simplicity, the following conventions are adopted in this problem. A word is a sequence of lowercase letters of the English alphabet. A phonetic unit is a sequence of consecutive vowels. Vowels are the letters "`a`", "`e`", "`i`", "`o`" and "`u`". For example, the word "contest" contains two phonetic units, and the word "beautiful" contains three of them.

The problem is to find the number of segments of consecutive words, which, if two line breaks are added to them after any two words, would form a haiku.

For example, there are two potential haiku in the text "if the real beauties of sunset in a suspended moment call for the thunder forever":

the real beauties of
sunset in a suspended
moment call for the

and

beauties of sunset
in a suspended moment
call for the thunder

Input

The first line of inpit contains integer n — the number of words in text that you have found ($1 \le n \le 10^5$). The next n lines contain words of lowercase letters. The length of each word does not exceed 20. It is guaranteed that each word contains at least one phonetic unit.

Output

Output the number of potential haiku in this text.

Example

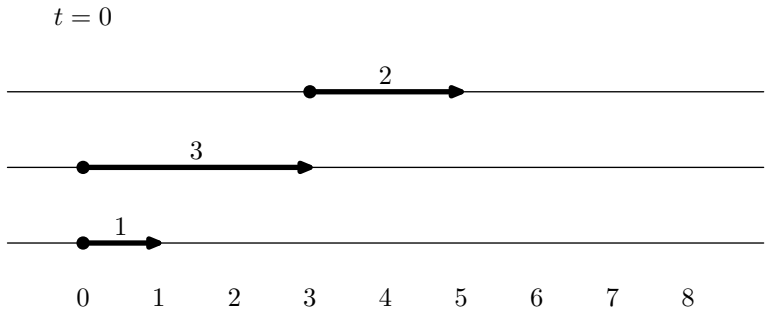
standard input	standard output
15 if the real beauties of sunset in a suspended moment call for the thunder forever	2

Problem F. Race

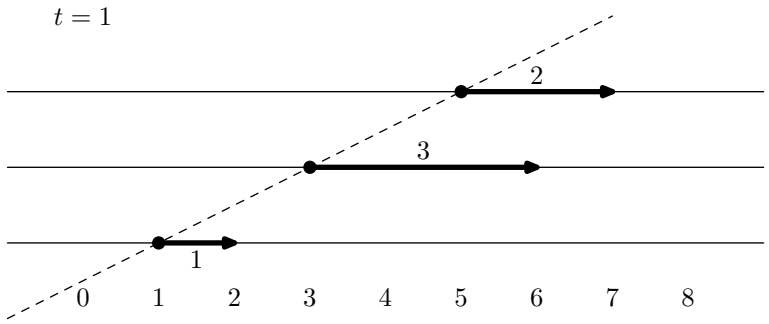
Time limit: 2 seconds
Memory limit: 512 megabytes

The race for n sportsmen is organized at a stadium. The sportsmen are running along n linear tracks of the stadium. Let us consider each track to be a horizontal line, the i-th track is a line $y = i$.

The sportsman i starts running at a point (s_i, i) and runs to the right with the speed of v_i. The race is long, so let us consider that it never ends, and the sportsmen never stop.



Young photographer Daniel is watching the race. He wonders what is the maximum number of sportsmen that would at some moment be on the same straight line. Help him find that out!



Input

Tha first line of input contains an integer n — the number of race participants ($1 \le n \le 300$).

The following n lines describe sportsmen, the i-th of them contains two integers s_i and v_i — the initial x-coordinate of the i-th sportsman and her speed ($-10^6 \le s_i \le 10^6$; $1 \le v_i \le 10^6$).

Output

Output one integer — the maximum number of sportsmen that would be on the same straight line during the race.

Example

standard input	standard output
3 0 1 0 3 3 2	3

Problem G. Maximaze XOR sum

Time limit: 1 second
Memory limit: 512 megabytes

Let us use $\oplus$ as the symbol for the operation of *bitwise "exclusive or"* for integers. In C++ and Java it is denoted by the character "^", in Pascal and Python — by the keyword "`xor`". For example, $9 \oplus 3 = 1001_2 \oplus 11_2 = 1010_2 = 10$.

You are given two integer arrays A and B of length n. Let's denote $X(A)$ as the result of calculating bitwise "exclusive or" for all elements of the array: $X(A) = A_1 \oplus A_2 \oplus \ldots \oplus A_n$. Simiarly, let's denote $X(B) = B_1 \oplus B_2 \oplus \ldots \oplus B_n$.

For each i from 1 to n, it is allowed to swap elements A_i and B_i. You must find out which elements should be swapped in order for the sum $X(A) + X(B)$ to be maximum possible.

Input

The first line contains an integer n — the size of the arrays ($1 \le n \le 10^5$). The next line contains n integers A_i — elements of the array A ($0 \le A_i \le 10^{18}$). The next line contains the array B in the same format.

Output

The first line of output must contain the maximum possible sum $X(A) + X(B)$ and an integer k — the number of required swaps. The next line must contain k different integers from 1 to n — indices of the elements to be swapped.

Example

standard input	standard output
2 1 1 2 2	6 1 1

Note

In the example after the swap the arrays are $A = [2, 1]$ and $B = [1, 2]$.

$X(A) = 2 \oplus 1 = 10_2 \oplus 1_2 = 11_2 = 3$, $X(B) = 3$, $X(A) + X(B) = 6$.

Problem H. Octopus Game

Time limit: 1 second
Memory limit: 512 megabytes

The tournament of "Octopus Game" is held in some country.

This round the participants will deal with math puzzle. Each player has two cards, initially there are integers a_0 and b_0 at the cards, respectively.

Players make actions with their cards. Let the integers on player's cards be a and b. The player first chooses an integer k, and then performs one of the following operations:

1. replace the integer on the first card with $a + kb$;
2. replace the integer on the second card with $b + ka$.

While playing, the absolute value of an integer written on a card must not exceed 10^{18}, otherwise something bad might happen. Those players are winning the round, who get 0 written on one of the cards, after performing at most 50 actions.

You are going to play the game, and of course you would like to win!

Input

The only line of input contains two integers a_0 and b_0 — the initial integers written on the cards ($-10^{18} \le a_0, b_0 \le 10^{18}$).

Output

The first line must contain n — the number of actions that the player is willing to perform to get 0 on one of the cards ($0 \le n \le 50$). Note that you need not minimize the number of actions, but it must not exceed 50.

The following n lines must contain two space separated integers each: t_i and k_i — the type of the respective action and the chosen integer k.

If there are multiple valid solutions, it is allowed to output any of them, but note that during the game the integers on the cards must not exceed 10^{18} by their absolute values.

Examples

standard input	standard output
-3 9	1 2 3
-27 57	2 2 2 1 9
56 15	6 1 -2 1 -1 2 -2 1 1 2 2 1 -4

Note

The first test requires just one action: add three times integer on the first card to the integer on the second card.

The second test: after the first action there are integers -27 and 3 on the cards, respectively, after the second action the integers are 0 and 3.

The third test: the integers on the cards are in turn: 56 and 15, 26 and 15, 11 and 15, 11 and -7, 4 and -7, 4 and 1, 0 and 1.

Problem I. Third Group Exam

Time limit: 1 second
Memory limit: 512 megabytes

One teacher came up with a new format for an exam.

- The exam consists of n blocks, each corresponding to one of the topics; a student receives a grade c_i for the i-th block for all i from 1 to n, all grades are independent;
- A grade for each block is an integer value from 0 to 100 both inclusive. A student chooses one way to get the grade for a block: to answer a *theoretical question* or to solve a *practical problem*;
- An exam is successfully passed if at least a blocks were graded by answering a theoretical question and at least b blocks were graded by solving a practical problem;
- If the previous condition is satisfied, the final grade for the exam C is calculated as the sum of grades for all blocks, that is $C = \sum_{i=1}^{n} c_i$.

Ilya is about to take the exam. He has a pretty good idea of his knowledge for each topic, and he is sure that passing the i-th block by theory will get him a grade of x_i, and passing it by practice — a grade of y_i. Help him determine which blocks (at least a of them) he should pass by theory and which blocks (at least b) he should pass by practice, to get the maximum possible total score for the exam.

Input

The first line of input contains three integers n, a and b — the total number of topics, the minimum number of topics to pass by theory, and the minimum number of topics to pass by practice, respectively ($1 \leqslant n \leqslant 2 \cdot 10^5$; $0 \leqslant a, b \leqslant n$). It is guaranteed that $a + b \leqslant n$.

The second line consists of n space-separated integers x_i — the grades Ilya will get if he passes the blocks by answering the theory questions ($0 \leqslant x_i \leqslant 100$).

The third line consists n of integers y_i in the same format — the grades he will get by solving practice problems ($0 \leqslant y_i \leqslant 100$).

Output

The first line of output must contain a single integer C — the maximum total grade that Ilya can get for the exam.

The second line must contain n space-separated characters, the i-th of which is '`T`' if Ilya should answer theory in the i-th block, and '`P`' if he should solve practice. At least a of the characters must be equal to '`T`', and at least b of them must be equal to '`P`'.

Examples

standard input	standard output
4 1 1 10 30 50 70 80 60 40 20	260 P P T T
4 1 1 30 40 60 90 10 25 50 85	215 T T T P
4 2 1 0 17 70 13 2 21 55 99	190 T P T P

Problem J. Computational ethnography

Time limit: 1 second
Memory limit: 512 megabytes

Native inhabitants of the L Island write numbers the other way round: most significant digits of a number are written in the end. For instance, number 144 is written as 441.

Novice ethnographer-mathematician Petya is studying square numbers and the culture of Island L's natives. He noticed that some numbers are perfect squares when considered both as regular numbers and as written by Island L's native inhabitants. For instance, number 144 mentioned above is such a number: when considered as written in usual way $144 = 12^2$ and when considered as number 441 written by natives, then $441 = 21^2$. Petya calls such numbers *interesting*.

Petya is interested how many interesting numbers there are from A to B inclusive.

Input

The first line of input contains integer A, the second line of input contains integer B ($1 \le A \le B \le 10^{11}$).

Output

Output the number of interesting numbers from A to B.

Example

standard input	standard output
1 1000	10

Note

In the first sample test interesting numbers are 1, 4, 9, 121, 144, 169, 441, 484, 676 and 961. Island L's native inhabitants don't use leading zeros when writing numbers, so 100 is not an interesting number.

Problem K. Work or Sleep!

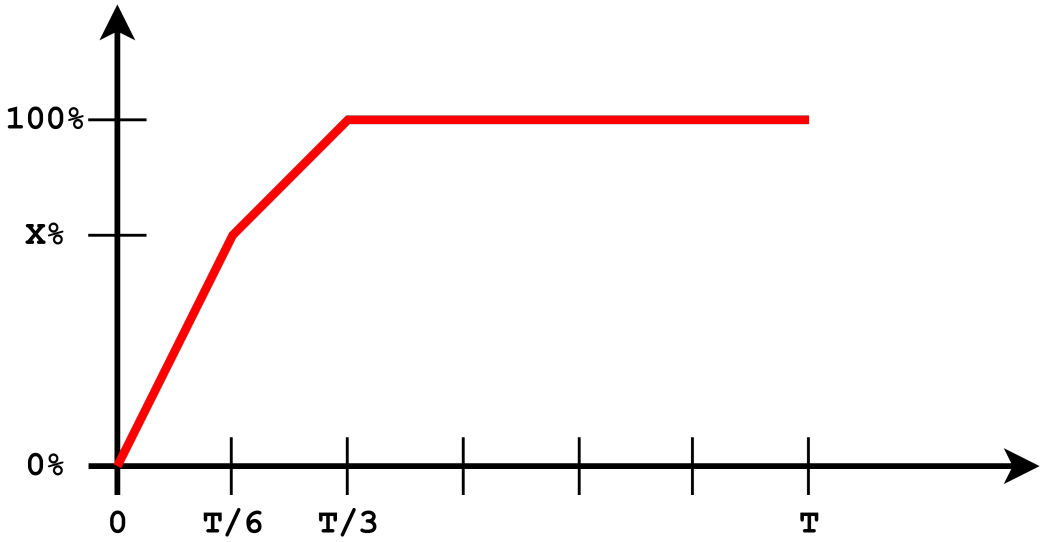
Time limit: 1 second
Memory limit: 512 megabytes

Oleg is a programmer from a planet where there are exactly T hours in a day. Oleg adheres to the principle of "Work or Sleep". According to this principle, all the time that you are not sleeping, you are working.

Recently Oleg has read an article in a scientific journal about the dependence of a programmer's performance on sleep time. The results of the study are the following:

- If you sleep $T/3$ hours a day, then the performance will be 100%.
- If you sleep $T/6$ hours a day, then the performance will be X%.
- If you sleep 0 hours a day, then the performance will be 0%.
- Also, if you sleep from $T/6$ to $T/3$ hours a day, the performance will increase linearly from X% to 100%.
- Also, if you sleep from $T/6$ to 0 hours a day, the performance will decrease linearly from X% to 0%.

Oleg decided to structure this information, so he drew a graph of the dependence of performance on sleep time and got the following result:



Oleg believes that the amount of work he will do per day is equal to the product of working time and performance. So, the problem is that if you sleep more, you work less, and if you sleep less, you have less performance.

Oleg wants to get back to work as soon as possible. So, help him determine the maximum possible daily amount of work that he can do with the optimal choice of sleep time.

Input

Input contains two integers X and T — performance as a percentage if the sleep time is $T/6$, and the number of hours per day on Oleg's planet ($0 \le X \le 100$, $1 \le T \le 10^5$).

Output

Output a single real number — the maximum possible daily amount of the work that can be done by Oleg. Your answer will be considered correct if its absolute or relative error doesn't exceed 10^{-6}.

Examples

standard input	standard output
75 24	1600.00000000
100 24	2000.00000000
77 123	8214.26086957

Note

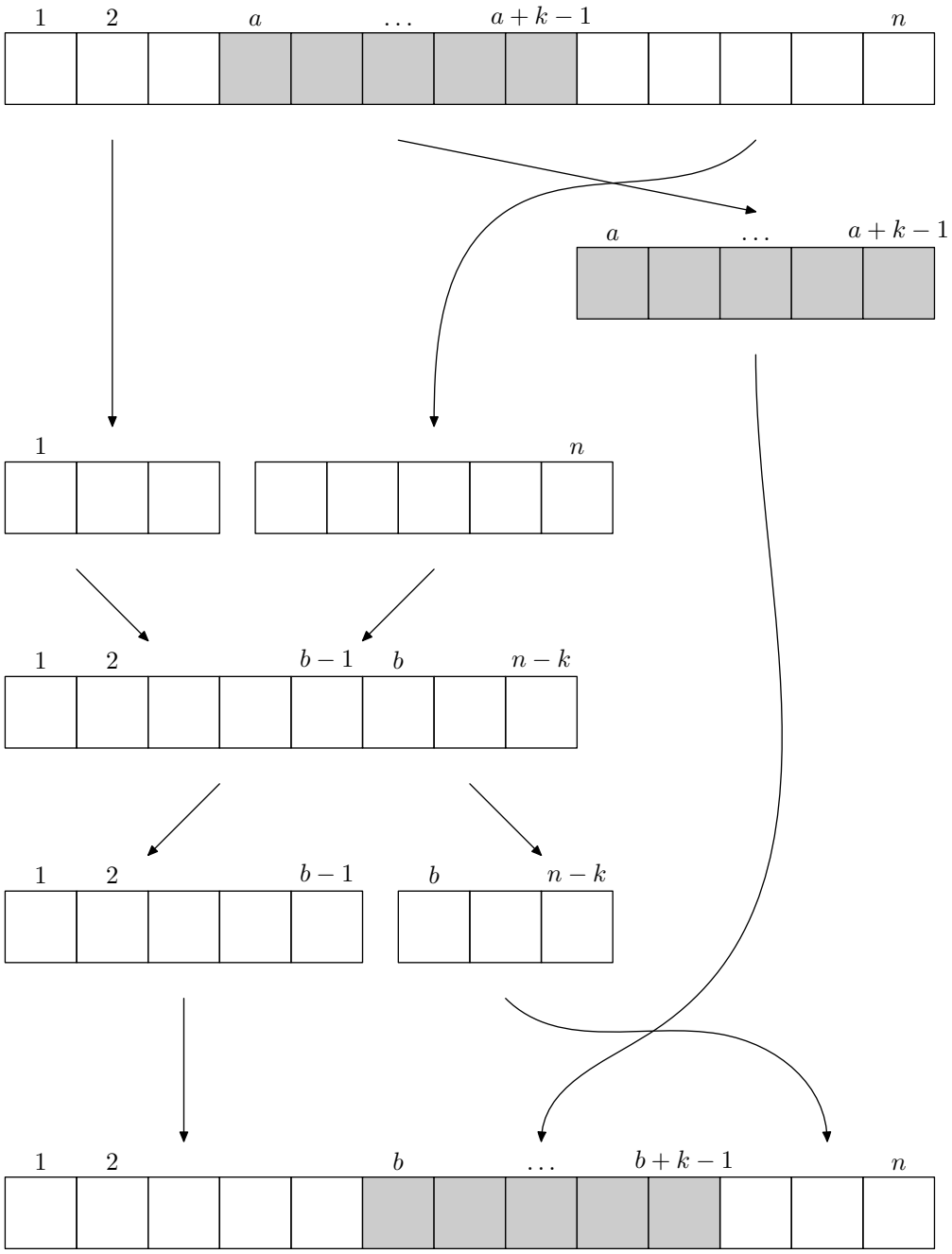
In the first test case, the maximum amount of the work that could be done if Oleg sleeps for eight hours. In the second test case, Oleg needs to sleep for four hours to maximize his amount of work.

Problem L. Permutation Transformation

Time limit: 1 second
Memory limit: 512 megabytes

Fedor works in the department of permutation transforming. Today Fedor should solve the following problem: he needs to transform the permutation $[p_1, p_2, \ldots, p_n]$ of integers $1, 2, \ldots, n$ to the permutation $[q_1, q_2, \ldots, q_n]$ using at most n^3 k-transfer operations.

Consider an array of length n. The k-transfer operation with the parameters (a, b) is defined as follows: a segment of k consecutive elements starting with an element at index a is cut away from the array and inserted back starting with the index b.



More formally: consider an array $[t_1, t_2, \ldots, t_n]$ and two integers a and b ($1 \le a, b \le n-k+1$). Let's create the temporary array $[r_1, r_2, \ldots, r_{n-k}]$, consisting of the numbers $[t_1, t_2, \ldots, t_{a-1}, t_{a+k}, t_{a+k+1}, \ldots, t_n]$. Then the result of the k-transfer with parameters (a, b) for an array t is an array, consisting of the numbers $[r_1, r_2, \ldots, r_{b-1}, t_a, t_{a+1}, \ldots, t_{a+k-1}, r_b, r_{b+1}, \ldots, r_{n-k}]$.

Fedor doesn't know how to solve the task, so he asks you to help him!

You are to solve the problem for t test cases.

Input

The first line contains a single integer t ($1 \le t \le 100$) — the number of test cases.

Each test case consists of three lines. The first line contains two integers n and k ($1 \le k \le n \le 100$).

The second line contains n different integers $p_1, p_2, \ldots, p_n$ ($1 \le p_i \le n$) — the permutation p.

The third line contains n different integers $q_1, q_2, \ldots, q_n$ ($1 \le q_i \le n$) — the permutation q.

It's guaranteed that the sum of n over all test cases doesn't exceed 100.

Output

Print the answer for each test case. Output your answer for a single test case in the following format.

If it's impossible to obtain a permutation $q_1, q_2, \ldots, q_n$ from a permutation $p_1, p_2, \ldots, p_n$ using k-transfers, print a single line consisting of the word "NO".

Otherwise, print "YES" at the first line.

The second line must contain a single integer m — the number of k-transfers performed to obtain the permutation q from the permutation p ($0 \le m \le n^3$). Note that you don't need to minimize m. It's guaranteed that if the permutation q can be obtained from the permutation p using k-transfers, then there is a solution that requires at most n^3 operations.

Each of the following m lines should contain two integers — parameters a and b for the corresponding k-transfer.

Example

standard input	standard output
3 2 1 2 1 2 1 4 2 1 2 3 4 1 2 4 3 3 2 2 1 3 1 3 2	YES 0 NO YES 2 1 2 1 2

Note

In the third test case there is another way to obtain a permutation q from a permutation p — a single k-transfer with the parameters $a = 2$, $b = 1$.