Problem A. Agnatic Seniority

Input file:	`agnatic.in`
Output file:	`agnatic.out`
Time limit:	1 second

gnatic seniority is a patrilineal principle of inheritance where the order of succession to the throne prefers the monarch's younger brother over the monarch's own sons. Monarch's children (the next generation) succeed only after the males of the elder generation have all been exhausted. Agnatic seniority essentially excludes females of the dynasty and their descendants from the succession. The *progenitor* of a dynasty is the first king in the dynasty and the common ancestor to all subsequent dynasty kings.

Agnatic seniority has been used in several historical monarchies. Acmland is one of the oldest such monarchies. Recently archaeologists have found documents with some names of the kings of the First Acmland Royal Dynasty. It is a known fact that names of sons in this dynasty were derived from the name of their father. A son had either the name of his father, or the name produced by removing all occurrences of some Roman letter from the name of his father. As a result, several sons of the same father might have the same name.

Unfortunately, the name of the progenitor of the First Acmland Royal Dynasty was lost long time ago. In order to prove the authenticity of the discovered documents scientists want to know could it be possible that the discovered names of the kings belonged to the same dynasty. Help scientists answering this question. Moreover, in case of the positive answer find a suitable dynasty progenitor name of the minimum length.

Input

The first line of the input contains integer number N ($1 \le N \le 200$) — the number of names in the discovered documents. The following N lines contain the names W_1, W_2, $\ldots$, W_N — one name per line. All names consist of the uppercase Roman letters, are nonempty, and are no longer than 10 000 letters in length each. The names are listed in an arbitrary order. Some names in the list may coincide.

Output

If the dynasty progenitor for the names W_1, W_2, $\ldots$, W_N could exist, print "`YES`" (without quotes) on the first line and the progenitor name of the minimum length on the second line of the output. Otherwise output a single word "`NO`" (without quotes). If there are several possible variants output any of them.

Example

agnatic.in	agnatic.out
3 ACM EVENT HINT	YES HIACMEVENT
4 HENRY HENRY HENRY HENRY	YES HENRY

Comment to the example 1

```
              HIACMEVENT
               /     \
              .       .
             /         \
            .           .
           /             \
       HIEVENT            .
        /   \              \
       .     .              .
      /       \              \
  EVENT     HINT              .
                               \
                               ACM
```

Name `ACM` could be derived from `HIACMEVENT` by removing all occurrences of letters `H`, `I`, `E`, `V`, `N`, `T`, so `ACM` might be a descendant of `HIACMEVENT` in the 6-th generation.

Comment to the example 2

```
     HENRY
    /  |  \
   /   |   \
  /    |    \
HENRY HENRY HENRY
```

Problem B. Blackboard

Input file:	`blackboard.in`
Output file:	`blackboard.out`
Time limit:	1 second

new electronic blackboard has been recently developed by the Anti-Chalk Membership (ACM).

It holds a sequence of bits (displayed as zeroes and ones). Since the new blackboard is a high-tech product no chalk is allowed nearby. To alter the content of the blackboard it has four buttons conveniently located on its sides. The buttons are named `L0`, `R0`, `L1`, and `R1`.

When a user presses `L0` button the leftmost bit of the sequence is erased, hence shifting the sequence one position to the left. Next, a 0-bit is appended to the right of the sequence.

Button `L1` acts similarly except for it adds 1-bit to the right after erasing the leftmost bit.

Buttons `R0` and `R1` act in a similar way but shift the sequence in the opposite direction. Namely, button `R0` erases the rightmost bit shifting the sequence to the right and then appends 0-bit to the left. Button `R1` appends 1-bit instead of 0-bit.

Changing the information displayed on the blackboard is quite cumbersome, so the user would normally want to minimize the number of button presses. Given the initial and the final configurations, your task is to transform the former to the latter with the fewest possible operations.

Input

The first line of the input contains the initial sequence currently displayed on the blackboard. The second line contains the desired final sequence. Both sequences are nonempty, have the same length not exceeding 100 000, and consist of zeros and ones (without spaces).

Output

Output the minimum number of button presses.

Example

blackboard.in	blackboard.out
0111 0110	2
0110 1111	3

Problem C. Coding

Input file:	`coding.in`
Output file:	`coding.out`
Time limit:	2 seconds

he International Coding Procedure Committee (ICPC) deals with data manipulation tasks. One of its latest achievements is the algebraic convolution method (ACM) that allows for rapid composition of cyclic shifts in strings.

In an attempt to hire more young talented programmers, ICPC has decided to conduct a contest and present the Top Decoder award.

The participants are given a ciphertext Y and a sequence of cyclic shifts that had produced Y from some unknown cleartext X. Each cyclic shift is given by three parameters: i, j, and k. Given a current string Z, the shift with parameters (i,j,k) applies to the substring $Z[i..j]$ (from i-th to j-th character, inclusive) and cyclicly rotates it to the right k times. String characters are numbered starting from one.

Given the above information, your task is to guess the initial cleartext X.

Input

The first line contains the ciphertext, which is a nonempty string consisting of N lowercase Roman letters ($1 \le N \le 50\ 000$). The second line contains the number of shifts M ($1 \le M \le 50\ 000$). The following M lines describe the sequence of cyclic shifts (in the order of their application to the cleartext). Each shift is described by three parameters i, j, k ($1 \le i < j \le N, 1 \le k \le j - i$).

Output

Output the only line containing the cleartext.

Example

coding.in	coding.out
logoduck 3 1 3 1 4 5 1 1 4 1	goodluck

Problem D. Dent's Delivery

Input file:	`delivery.in`
Output file:	`delivery.out`
Time limit:	1 second

s a winner of the annual Vogon Interstellar Creative Poetry Contest Arthur Dent was awarded with a newest model of the Deep Thought netbook series. The Alpha Canis Major hypertransporting company had delivered the computer at the Arthur's gate. The Deep Thought netbook had been delivered packed into a $1m \times 1m \times 1m$ cubical box. One side of the box was marked as "fragile", corresponding to the side of the netbook containing keyboard and screen. Initially the box was positioned at the gate with the "fragile" side of the box not on the surface.

However, the box was so large and so heavy that it could be moved from one place to another only by rolling it over edges. The "fragile" side of the box must not appear on the surface, or the keyboard and screen would break. The plan of the Dent's lot was located in the local planning office. The plan was drawn on the grid paper, fortunately the grid size is $1m$, which was exactly the size of the box. Initially the box was placed at the gate, that corresponded to the cell $(1, b)$ on the plan and should be moved to the door, that corresponded to another cell (c, d) of the plan. The plot was rectangular with the size $M \times N$ meters and was surrounded by a fence, so the box could not roll out of the plot.

Marvin felt very depressed about moving the box around the plot. It could feel a bit less pessimistic if you'd written a program, which output a sequence of rolls across edges moving the box from the gate to the house.

Input

The first line of the input contains integer values M, N, b, c, d in this order ($1 \le M, N \le 10\ 000$, $1 \le b, d \le N$, $1 \le c \le M$). The second line contains one letter `L`, `R`, `T`, `F`, or `B` denoting the initial position of the "fragile" side of the box (left, right, top, front, and back respectively). The back side of the box is directed to the gate.

Output

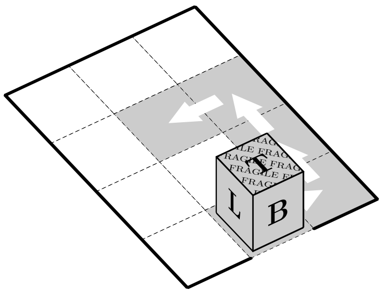
Output a sequence of rolls moving the box from the gate to the door of the house. The four possible rolls of the box are denoted as follows:

`L`	(left)	the second coordinate of the box decreases
`R`	(right)	the second coordinate of the box increases
`F`	(forward)	the first coordinate of the box increases
`B`	(back)	the first coordinate of the box decreases

The total number of rolls cannot exceed $4(M + N)$. If this is not possible, output `IMPOSSIBLE`.

Example

delivery.in	delivery.out
4 3 2 3 2 T	RFFL
2 1 1 2 1 F	IMPOSSIBLE



Problem E. Equality of routes

Input file:	`equality.in`
Output file:	`equality.out`
Time limit:	1 second

fter a series of grave incidents with the air traffic the global World Air Company was created. Now there are only 40 largest airports left on Earth and there is a single flight from each of them to any other. Also the company decided to fix the price of each flight and remove all airport fees.

A route is a sequence of flights connecting some cities. Calculating the price of a route is simple — you just have to sum up the prices of its flights.

You are given several pairs of routes $(A_1, B_1), \ldots, (A_n, B_n)$, where routes A_i and B_i are known to have the same price and the same departure and destination cities.

Given another pair of routes (U, V) with the same departure and destination cities, you have to check if U and V definitely have the same price.

Input

The first line of the input contains a single integer N ($1 \le N \le 100$). The next N lines describe pairs of routes (A_i, B_i) that are known to be of the same price. Each route in given by a sequence of distinct airport codes. Each code is a sequence of three capital Roman letters. Routes are separated by the equality sign "=" (without quotes).

The final line of the input contains a pair of routes (U, V) for which you have to determine if their prices must be equal. Routes U and V are given in the same way as A_i and B_i but are separated by the question mark "?" (without quotes).

Airport codes and symbols "=" and "?" are separated by spaces (there is exactly one space between words, there are no spaces after the last word). All the routes are nonempty and the total number of flights is not greater than 10 000.

Output

The output should consist of a single word: "YES" (without quotes) if the given routes U and V must be of the same price, and "DUNNO" (without quotes) if the prices of U and V might differ.

Example

equality.in	equality.out
1 DME CDG = DME FRA CDG DME CDG MAD ? DME FRA CDG MAD	YES
2 DME CDG = DME FRA CDG SWO DME = SWO MON NCE DME DME CDG NCE MON ? DME FRA CDG MON	DUNNO

Problem F. Farm

Input file:	farm.in
Output file:	farm.out
Time limit:	1 second

ingo — John had just won the first prize in a lottery! First of all he was going to build a new farm of his own project.

Next morning after the winning day John started drawing the plan of his future farm. He took a sheet of grid paper and outlined several alternative projects. Each project had a fence consisting of the segments laid either on cell sides or cell diagonals of the grid.

John noticed that different layouts of the same number of fence segments might bound regions of different areas. Help John to determine the maximum area of a region that can be bounded with no more than N fence segments.

Input

The input contains one integer number N ($3 \le N \le 10\,000$).

Output

Output the maximum area of a region that can be bounded with no more than N segments. Answer must be accurate up to 0.1.

Example

farm.in	farm.out
3	0.5

Problem G. Garden Adventure

Input file:	garden.in
Output file:	garden.out
Time limit:	1 second

our are looking at a garden from above and can see its trees as non-overlapping circle disks in the plane.

Next, you can see a pair of circle lakes A and B.

Finally, you notice Alice and Bob swimming in lakes A and B, respectively. They are getting tired, feel lonely, and want to see each other but the trees may block their sights.

Your task is to help the couple and check if it is possible for them to choose their positions in lakes A and B so that the segment connecting these points does not intersect any of the trees.

Input

The first line of the input contains the number of trees N ($0 \le N \le 256$). Each of the following N lines contains three integers — the coordinates of the center of the corresponding tree and its radius, respectively. The last two lines contain the coordinates of the center and radius of the two lakes. All coordinates do not exceed 10 000 by absolute value.

Trees and lakes cannot overlap but may touch each other.

It is guaranteed that if Alice and Bob can view at each other then it is possible to choose their positions inside the corresponding lakes so that the segment connecting these positions is no closer than 10^{-6} m to any tree.

Output

Output "YES" (without quotes) if Alice and Bob can see each other by choosing appropriate positions in their lakes (as described above). Otherwise, output "NO" (without quotes).

Example

garden.in	garden.out

1
0 0 2
0 4 1
0 -4 1
	NO

1
0 0 1
0 4 2
0 -4 2
	YES

Problem H. Harmless

Input file:	harmless.in
Output file:	harmless.out
Time limit:	1 second

t the far end of the Galaxy lies a small, unregarded yellow star. Orbiting this star at a distance of roughly eight light minutes is an utterly insignificant planet. Those few encyclopaedias paying any attention to such insignificant matters reward the planet with an entry consisting of a single word — “harmless”. However, the recent study indicates that this description should be updated to “mostly harmless”. On the planet there exists an utterly insignificant local university known as Harmvard University.

The Harmvard University has M faculties each accepting at most K applicants. The number of applicants for the class of 2013 is N, and each applicant has filed applications to all M faculties. The university admission office has collected lists of personal preferences of each of K applicants. A list of personal preferences is a list of all M faculties arranged in the order of preference for the corresponding applicant. The first faculty in the list has the highest preference, the second is the second highest, etc.

Admission requirements are different for each faculty, thus each faculty has its own rating list of N applicants rated in the decreasing order of the applicant’s rating for this faculty. The first K applicants in the rating list of a faculty form the *shortlist* for that faculty. Only the applicants from the shortlist of the faculty are admitted to that faculty. But as initially one applicant may be in several shortlists, the following procedure is used by the admission office to shake down the shortlist of the faculties.

1. Consider all applicants that appear in the shortlist for their first preference. We will now remove them from the rating lists of all other faculties but their first preference.
2. Now, the shortlists are updated, and it may appear that some more applicants appear in the shortlist for their first preference. If that is the case, we go to step 2 again and repeat steps 2 and 3 until there’s no new applicant that is in the shortlist for his/her first preference.
3. Now, consider all applicants that appear in the shortlist for their second preference. We will now remove them from the rating lists of all faculties but their first and second preferences.
4. The shortlists are updated again, and it may appear that some more applicants appear in the shortlist for their first preference. If yes, then go back to step 2. If not, but there are new applicants in the shortlist for their second preference, we go back to step 4.
5. And so on.

Let’s explain the algorithm more formally. We define $PROCESS(q)$ as “take all candidates who are in the shortlist for their q-th preference and remove them from the rating lists of their $q+1$-th, $q+2$-th, ... preferences”. The algorithm then repeatedly chooses the smallest q for which $PROCESS(q)$ performs at least one removal and executes $PROCESS(q)$.

We repeat the above steps until the rating lists stop changing. At this point, we admit people from each shortlist to the corresponding faculty.

The above procedure is so nice that the resulting assignment possesses the following properties. Each applicant may be accepted to at most one faculty. If the faculty F accepting an applicant A is not the first in A’s list of personal preferences, no faculty G with higher preference in A’s list of personal preferences will accept an applicant B with rating less than the rating of A in the rating list of the faculty G. If an applicant B is not accepted by any faculty, no faculty F will accept an applicant C with rating less than the rating of B in the rating list of the faculty F.

You need to implement the procedure.

Input

The first line of the input contains values N, M, and K ($0 < M \leq 100$, $0 < NM \leq 100\,000$, $0 < KM \leq N$). Each of the next N lines contains M distinct numbers in range from 1 to M that constitute the list of personal preferences of the corresponding applicant — the numbers of faculties in the order the applicant prefers them (from the most preferred to the least preferred). Each of the next M lines contains N distinct numbers in the range from 1 to N that constitute the rating list of the corresponding faculty (from the highest rated applicant to the lowest rated).

Output

The output must contain M lines. The i-th line must contain the numbers of applicants accepted by the i-th faculty. Each line of output must contain exactly K distinct numbers. The applicants’ numbers in these lists must appear in the order of their ratings for the corresponding faculty. Please note that the lists are uniquely determined since the above procedure has no randomness involved.

Example

harmless.in	harmless.out

10 3 3
2 1 3
1 2 3
1 2 3
3 1 2
3 2 1
2 3 1
3 1 2
2 1 3
2 1 3
1 2 3
3 4 1 6 9 2 7 8 5 10
1 3 7 4 9 8 2 5 10 6
5 8 3 1 9 6 2 4 7 10
	3 4 2
	1 7 9
	5 8 6

Problem I. Infinite Improbability Drive

Input file: `infinite.in`
Output file: `infinite.out`
Time limit: 1 second

ne of those exceptionally improbable events, which initiated a sequence of even more exceptionally improbable events, which eventually ended up in delivery of the Deep Thought netbook at the gate of the Arthur Dent’s house, took place in the same laboratory, where, in fact, the Infinity Improbability Drive was invented some million years later.

The standard Improbability Computing Portable Connector was a device generating a sequence of random bits. The bits were printed out on a tape of soft, comfortable paper, which was good for the purpose of thinking and relaxing. In one exceptionally boring day one exceptionally average scientist noticed one exceptionally improbable fact: the fragment of the tape in his hands contained all possible sequences of length N of binary digits as substrings. But... oops, this fragment of tape had spilled onto floor.

Excited by that extraordinary event the scientist ran to the lab eager to reproduce his findings, but found himself at a complete state of disarray, as the main computer was down due to maintenance work. This story might left unknown to public, but by an exceptionally lucky chance the death letter of that scientist had been found in the box of the Arthur’s netbook some million years later.

Write a program regenerating that sequence and outputting the shortest string of 0 and 1 containing all possible sequences of length N as substrings.

Input

The input contains a single number N ($1 \le N \le 22$).

Output

Output any of the shortest strings of 0's and 1's satisfying the above property.

Example

infinite.in	infinite.out
2	01100

Problem J. Jeltz' Torture

Input file:	`japanese.in`
Output file:	`japanese.out`
Time limit:	1 second

rostetnic Vogon Jeltz, the captain of the Vogon Construction Fleet, is notorious for torturing his captives with renditions of poetry of his own composition. Only slightly less notorious he is for his obsession with a kind of puzzles called Japanese crosswords. No historians ever know what the word "Japanese" stands for, or where those puzzles have come from. Arthur Dent has narrowly escaped death from Vogon poetry and now faces the next challenge. Given a picture $M \times N$ he has to produce a form for filling up a Japanese crossword.

Japanese crosswords, also known as Nonograms are picture logic puzzles in which cells in a grid have to be colored or left blank according to numbers given at the side of the grid to reveal a hidden picture. In this puzzle type, the numbers measure how many unbroken lines of filled-in squares there are in any given row or column. For example, a clue of "4 8 3" would mean there are sets of four, eight, and three filled squares, in that order, with at least one blank square between successive groups.

To the right is an example of a completely open Japanese crossword.

Given a non-blank picture $M \times N$ (i.e. containing M rows and N columns) print out the clues and empty grid for this picture. Follow the format of the the example as shown below.

```
**____2____
**467747764
22.XX...XX.
44XXXX.XXXX
_9XXXXXXXXX
_9XXXXXXXXX
44XXXX.XXXX
_7.XXXXXXX.
_5..XXXXX..
_3...XXX...
_1....X....
__.........
```

Input

The first line of the input contains integer numbers M and N ($0 < M, N < 200$). The next M lines contain N characters each either "." (dot) or "X" (capital Roman letter "X"). The lines end with a newline character. The length of unbroken lines of filled-in squares in any given row or column does not exceed 9. The picture has at least one colored cell.

Output

Output the form for filling up the crossword as shown below. The nonsignificant top left corner is filled with '*' characters. If a position in a clue line or column is empty, an underscore '_' is written.

Example

japanese.in	japanese.out
`10 9`	`**____2____`
`.XX...XX.`	`**467747764`
`XXXX.XXXX`	`22.........`
`XXXXXXXXX`	`44.........`
`XXXXXXXXX`	`_9.........`
`XXXX.XXXX`	`_9.........`
`.XXXXXXX.`	`44.........`
`..XXXXX..`	`_7.........`
`...XXX...`	`_5.........`
`....X....`	`_3.........`
`.........`	`_1.........`
	`__.........`

Problem K. Kaleidoscope

Input file:	`kaleidoscope.in`
Output file:	`kaleidoscope.out`
Time limit:	2 seconds

aleidoscopic publications, one of the great publishing houses of Ursa Minor Beta publishes the Beginner's Guide to Kaleidoscopes, which is, in fact, the second most popular book in the Galaxy. It outsells The Kaleidoscope Encyclopaedia due to two simple facts: 1) it is written in simple plain language that even the Hedroses of the Reddish swamps can understand, and 2) it contains numerous pictures of earthlings swimming in round pools.

This is what The Beginner's Guide talks about Betelgeuse Kaleidoscopes. A Betelgeuse kaleidoscope is much like a Barnard one, but larger and completely different. It contains N beads, each having a fixed-point positive number written on it. No two beads have the same number. Each bead may drop into one of K ($K > 2$) cells. Each cell must accommodate at least one bead. A kaleidoscope is called *blasting* if the difference of the numbers written on any two beads from the same cell is not equal to 10^m for any integer m, including negative. A blasting kaleidoscope gives its viewers such an experience that Betelgeuse kaleidoscopes are strictly banned in almost all populated worlds in the Galaxy.

Write a program, which finds a distribution of N beads into K cells producing a blasting kaleidoscope if this is possible.

Input

The first line of the input contains integer values N and K ($3 \le N \le 3 \cdot 10^5, 3 \le K \le N$) respectively. The next N lines contain one fixed-point number each. No number contains more than 15 digits. If a fixed-point number has zero fractional part (i.e. it is an integer value), the decimal dot is not written.

Output

If a blasting kaleidoscope does not exist for the given input data, output one number -1. Else output K lines, each line containing the numbers going into the corresponding cell. The numbers on the line are separated by a space character and may appear in arbitrary order.

Example

kaleidoscope.in	kaleidoscope.out

5 3
1
2
3
0.1
0.2
	3
	1 0.1
	2 0.2

Problem L. Laboratory of ACM

Input file:	`laboratory.in`
Output file:	`laboratory.out`
Time limit:	1 second

he laboratory of Advanced Combinatorial Mechanisms (ACM) is ordered to construct a $2 \times 2 \times 2$ Rubik's cube codenamed R2D2.

Each of six facets of R2D2 has to be painted with some color. All these colors must be different. Moreover, R2D2 can be disassembled into eight $1 \times 1 \times 1$ corner cubes. Thus, each corner cube has three facets with a common vertex painted different colors, and the other three facets not painted at all.

You have got a shipment of N corner cubes in the depot, each having three facets painted with three different colors. A coloring is given by a sequence of three color names in clockwise order around the common vertex of the facets.

You task is to determine whether it is possible to choose eight of the given corner cubes and combine them into a proper R2D2 instance.

Input

The first line of the input contains the number of cubes N ($1 \le N \le 1\,000$). Each of the following N lines describes the corresponding corner cube and contains three different color names.

Color names are nonempty sequences of up to 16 Roman letters. Color names are case-sensitive. There is exactly one space between color names.

Output

Output "`YES`" (without quotes) if it is possible to choose eight cubes out of N given and build R2D2 out of them. Otherwise, output "`NO`" (without quotes).

Example

laboratory.in	laboratory.out
10 Red Cucumber Blue Red Blue Magenta Red Yellow Magenta Red Magenta Yellow Red Magenta Yellow Red Yellow Cucumber Cyan Blue Cucumber Cyan Magenta Blue Cyan Yellow Magenta Cyan Cucumber Yellow	YES
8 Red Cucumber Blue Red Blue Magenta Red Yellow Magenta Red Yellow Cucumber Cyan Blue Cucumber Cyan Magenta Blue Cyan Yellow Magenta Cyan Cucumber Yellow	NO