Problem A. Army on the March

Input file:	army.in
Output file:	army.out
Time limit:	4 seconds
Memory limit:	256 Mebibytes

Several soldiers are placed on a cellular field of size $N \times M$. They want to change their disposition to a new one. The soldiers can move simultaneously but at any moment there can be at most one soldier in each cell. Every second, each soldier either moves to one of the neighboring cells which has a common side with the soldier's cell or stays in his cell. There are obstacles in some cells so the soldiers can't go through them.

It is known that speed is one of the most valuable factors in the art of war. That is why the soldiers must move to their new disposition in minimal time. Secondly, the soldiers must get tired as little as possible to keep their energy for a battle. So, among the solutions which take minimal time, you must choose such a solution that the total length of the soldiers' movement paths is as small as possible. The length of the path is the number of movements between adjacent cells.

You must find out the minimal time enough for the soldiers to perform their task and the minimal total length of their movement paths.

Input

The first line of the input contains two integers N and M ($1 \le N, M \le 15$) — the size of the field. Each of the following N lines contains M symbols which describe the cellular field and define the initial and final positions of the soldiers. Each symbol in these lines represents one cell and can be one of the following:

- "`.`" — an empty cell;
- "`#`" — a cell with an obstacle;
- "`S`" — an initial position of a soldier;
- "`E`" — an empty cell, a final position of a soldier.

It's guaranteed that the soldiers can reach the cells marked with "`E`" from the cells marked with "`S`" (that is, connected regions of the field contain the same number of cells marked with "`S`" and "`E`").

Output

Output a single line with two integers T and P separated by space, where T is the minimal number of seconds required to change the disposition, and P is the minimal total length of the soldiers' movement paths.

Examples

army.in	army.out
3 3 S#. S.. #EE	3 6
3 5 SSSSS ##.## EEEEE	6 22

Problem B. Budget Cutting

Input file:	budget.in
Output file:	budget.out
Time limit:	3 seconds
Memory limit:	256 Mebibytes

Several square building plots were allocated in Howmaney to build a new enterprise. A fence should be built on the perimeter of the building plot. But the budget for construction was too big and it was decided to cut down the budget by reallocating the building plots. The total area should be the same and the length of the fence should be minimal.

Write a program that figures out how to modify the plan of the allocated building plots to minimize the length of the fence. If there are several such modifications, choose the one with the minimum number of changes made to the plan of the allocated plots. If there are still several possibilities, you may choose any of them.

Input

The first line of the input contains two integers N and M ($1 \le N, M \le 20$). Each of the following N lines contains M symbols, each of which is either "`#`" or "`.`". Symbol "`#`" defines a plot allocated for construction and symbol "`.`" defines a free plot.

Output

The first line of the output should contain two integers R and C ($1 \le R, C \le 40$). R lines should follow. Each of them should contain C symbols, each of which is "`.`", "`+`", "`-`" or "`#`". Symbol "`#`" defines a plot allocated for construction both in the original plan and in the new plan. Symbol "`+`" defines a plot that was free in the original plan, but became allocated for construction in the new plan. Symbol "`-`" defines a plot that was allocated for construction in the original plan, but became free in the new plan. Symbol "`.`" defines a free plot.

Example

budget.in	budget.out
2 3 #.# ###	4 3 ... -+# ### ...

Problem C. Contact

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

Contact is a word-guessing game for three or more players. The rules of this game are the following. One player (called a "wordmaster") secretly chooses a target word and the goal of other players is to guess that word by revealing its first letter, then its second letter, and so on until all the letters are revealed.

When the game begins, the wordmaster says the first letter of the target word for free. Each round, some player except the wordmaster secretly chooses a word ("clue word") that begins with the revealed prefix of the target word, and then tells everyone a description of that word. Note that the clue word should be strictly longer than the revealed prefix. A player tries to choose and describe a clue word in such a way that it is hard enough to stump the wordmaster but easy enough for the other players to "make contact" with the clue. If the wordmaster guesses the clue word, the guessers will fail the round, and they will have to think of another word. And if the wordmaster can't crack the clue word but some players guess it, the guessers win the round, and the wordmaster will have to say the next letter of the target word.

Players can use each word as a clue only once. If the clue word appears to be the target word, the game is over. If all letters of the target word have been revealed by the wordmaster, the game is over too. The goal of the game is to guess the target word (and thus end the game) as fast as possible.

Alexander, Oleg and Iliya have played this game so many times that they know all the words that can be chosen. Moreover, they know which of these words can be guessed by the current wordmaster: if such a word is picked as a clue word, the wordmaster will always guess it. For all other words, the wordmaster will always fail to crack them, so by picking them, players can always "make contact" and win a round.

You don't know the target word. The only thing you know is that it is chosen from the given list of words. Your task is to write a program which finds the minimal number of rounds before the game is over in the worst case.

Input

The first line of the input file contains number of words N ($1 \le N \le 10^4$). The following N lines contain information about these words. Each line of that block contains a word followed by one space and a digit 0 or 1. Here, 1 means that the word can be guessed by the current wordmaster and 0 means that the wordmaster could not guess that word. All words consist of lowercase Latin letters only and start from the same letter. The length of each word is from 2 to 100 letters, inclusive.

Output

On the first line of output, write the minimal number of rounds before the game is over in the worst case.

Example

contact.in	contact.out
7 cheat 1 cheburashka 0 cent 1 centaur 0 center 1 celebration 1 cell 1	4

Problem D. Dream Totem

Input file:	totem.in
Output file:	totem.out
Time limit:	1 second
Memory limit:	256 Mebibytes

Cobb: This one was hers. She'd spin it in a dream and it would never topple. Just spin and spin...

...

Cobb knows that we can lose our sense of reality in a dream very easily. In order to avoid this, Cobb's wife, Mal, has invented a totem. A totem is a small object and only its owner knows its exact physical characteristics. After looking at your totem, you can always find out whether you belong to someone's dream or not. To achieve this, a totem must be fairly unique. Cobb thinks that a good totem satisfies the following conditions:

- The totem must be a concave (non-convex) polyhedron.
- Each face of the totem must be a convex polygon.
- Due to the uniqueness requirement, the totem must have exactly n faces where n is a given number which Cobb can assign to each team member uniquely.
- The totem must have n true faces, i. e. there are no three vertices of a face that lie on the same line and there are no parallel adjacent faces.
- All totem vertices must have integer coordinates in some Cartesian coordinate system.

Given n, you have to generate any totem which satisfies Cobb's conditions.

Input

The first line of the input file contains one integer number n ($6 \le n \le 100$).

Output

Write exactly n lines to the output, one line per totem face. Each line must contain an integer k_i, followed by k_i triples of integers (x_j, y_j, z_j) — the coordinates of the i-th face in either clockwise or counter-clockwise order. Coordinates of vertices must fit in 32-bit signed integer type. Additionally, the sum of all k_i must not exceed 10^4.

Example

totem.in	totem.out
6	3 0 1 0 1 -1 0 0 0 1 3 0 1 0 1 -1 0 0 0 3 3 -1 -1 0 0 1 0 0 0 1 3 -1 -1 0 0 1 0 0 0 3 3 1 -1 0 -1 -1 0 0 0 1 3 1 -1 0 -1 -1 0 0 0 3

Problem E. Enclose the Trees

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

Flatlanders believe that four walls of the building symbolize the four basic elements — fire, water, air and earth. If the pairs of the elements — fire and water, earth and air — will equipoise each other, the building will stand firmly and it will be cool in summer and warm in winter. So, Flatlandish houses have a rectangular shape. But the harmony between the elements of water, earth and air is important for plants and the element of fire is dangerous for them. So Flatlandish fields, gardens and parks have the shape of an equilateral triangle.

It is necessary to protect few valuable trees by a fence in a shape of an equilateral triangle. The diameter of the trees is insignificant, the fence may touch the trees.

Input

The first line of the input contains one integer N ($2 \le N \le 2011$) — the number of trees. It is followed by N lines, each line contains two integers from 0 to 10^5, inclusive — the coordinates of the trees. All pairs of coordinates are distinct.

Output

Output a single real number — the minimum size of the edge of the triangular fence. Your answer should differ from the optimal one by no more than 10^{-6}.

Example

fence.in	fence.out
3 0 8 6 0 5 5	10.000000

Problem F. Fast Refactoring

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

It is known that many modern IDEs, such as Eclipse, have a wide support of refactorings — actions which can make a batch of automatic modifications by just hitting one button. Consider one kind of refactoring: a renaming. The renaming is a process of keeping the code consistent after changing a name of an identifier. After the rename refactoring is launched, IDE looks for references to the renamed identifier, and applies the same changes to them. In this problem, we will modify the source code only with rename refactorings.

We can do the following actions:

- Replace all occurences of the word under cursor with a new word.
- Remove one word under cursor (with the space symbol after that word). Other occurences of the same word are untouched.
- Print a code fragment that starts from a specified position and has a specified length.

A word is said to be "under cursor" if the cursor points at a symbol of the word or at the space immediately after the word.

The source code is just a single line which consists of Latin letters, digits and single spaces.

Input

The first line of the input file contains the source string S. Its length does not exceed 10^5. S consists of Latin letters, digits and spaces. S does not start or end with a space.

The second line of the input file contains the number of queries Q ($1 \le Q \le 10^5$).

The following Q lines contain descriptions of queries, one line per query. Each line satisfies one of the following formats:

- `REPLACE` p_i s_i
  Denotes a replace query of the word under the p_i position to the new word s_i. The string s_i contains only Latin letters and digits. The length of s_i does not exceed 10^5.
- `REMOVE` p_i
  Denotes a remove query of the word under position p_i.
- `PRINT` b_i l_i
  Denotes a print query of the fragment of the length l_i ($l_i > 0$) which starts on position b_i.

The symbol positions in all queries are numbered from zero. All positions and intervals do not violate current boundaries of the text, and the total length of the printed fragments does not exceed 10^5. Also, the total size of the input file does not exceed 1 MiB.

Output

For each print query, you must output a line with the required fragment.

Example

refactor.in	refactor.out
if a1 equals b then 5 REPLACE 4 A2 PRINT 3 5 REPLACE 12 less REMOVE 13 PRINT 0 12	A2 eq if A2 less b

Problem G. Garden

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

Several cherry trees are growing in Uncle Vanya's garden. The garden has a form of a convex polygon and is enclosed by a fence which is attached to pillars. The pillars are located at the vertices and on the edges of the polygon. Uncle Vanya wants to replace the old fence with a higher and stronger one so that:

- all trees would be strictly inside the area enclosed by the fence;
- the fence would be a polygon with vertices in the existing pillars;
- the length of the fence would be minimal.

Input

The first line of the input contains one integer N ($3 \le N \le 250$) — the number of pillars in the old fence. Each of the next N lines contains two integers — the coordinates of the pillars in clockwise order. The next line contains an integer K ($1 \le K \le 10\,000$) — the number of trees in Uncle Vanya's garden. Each of the next K lines contains two integers — the coordinates of the trees. It is guaranteed that all trees grow strictly inside the area enclosed by the old fence. Absolute values of all coordinates do not exceed 10^6. All pairs of coordinates are distinct.

Output

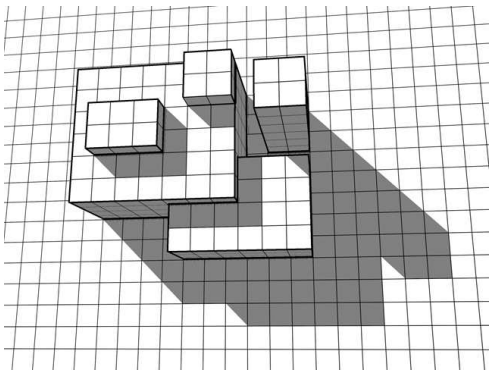
Print a single number — the minimum possible length of the new fence. Your answer should differ from the optimal one by no more than 10^{-4}.

Examples

garden.in	garden.out
6 -20 -10 -10 20 10 20 20 10 10 -20 -10 -20 2 0 10 0 -10	120.0000
4 0 4 4 4 4 0 0 0 3 1 3 2 3 1 2	13.6569

Problem H. Hot Day

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



It is too hot in Sharm el-Sheikh and the participants try to hide from the burning sun in shadow of hotel buildings while waiting for the contest outside the hall. But is there sufficient space in the shadow?

In this problem, the hotel buildings have a form of rectangular boxes. The sides of the boxes are parallel to the coordinate axes. The base of every box stands on the ground. The boxes can intersect to form complicated structures.

The sun is the only light source in the scene. You don't know the exact location of the sun, but you can assume that its coordinates are $(-X, -X, X)$, where X is a big positive number. The sun is so far that you must assume that light rays are parallel.

Input

The first line of the input contains one integer N ($0 \le N \le 1000$). Following N lines contain descriptions of N boxes. Every box is described by a line of five space-separated integers x_1, y_1, x_2, y_2, h — the coordinates of two opposite corners of the base of the box and the height of the box, respectively. The constraints are: $-1000 \le x_1, y_1, x_2, y_2 \le 1000$, $0 < h \le 1000$.

Output

Output the area of the shadowed part of the ground. You have to write the exact value.

Example

shadow.in	shadow.out
1 0 0 1 1 1	2
5 0 0 6 7 4 4 4 2 1 5 0 5 2 7 5 1 8 3 10 6 4 4 8 10 3	64.5

Problem I. Interchange

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

N mathematicians are sitting around a round table to calculate their balance after a cooperative journey. During the journey, each group member paid for himself and for the other group members. Everyone has some amount of money left after the trip and everyone has calculated how much money would have come out if he payed only for himself.

Instead of transferring money directly to each other, the mathematicians have calculated S, the average amount of money remained after the trip. They want to try to interchange the money as needed using only the following operations to transfer money.

Let the current amount of money of i-th person be equal to a_i. If $a_i > S$ then i-th person will have to give the amount of $a_i - S$ to a neighbor to the left and the same amount to a neighbor to the right. If $a_i < S$ then i-th person will take the amount of $S - a_i$ from a neighbor to the left and the same amount from a neighbor to the right.

Write a program that figures out whether the mathematicians can obtain the correct final result using the specified operations, and how to achieve it. It is allowed for the transferred amounts of money to be negative or fractional since the mathemathicians are used to deal with negative, fractional and even imaginary numbers.

Input

The first line of the input contains one integer N ($3 \le N \le 1000$) — the number of mathematicians. The second line contains N integers a_i from 0 to 1000 — the initial amounts of money left after the journey. The amounts of money possessed by mathematicians are listed in clockwise order of their arrangement at the table. The third line contains N integers b_i from 0 to 1000 — the desired final amounts of money in the same order. It is guaranteed that $\sum a_i = \sum b_i$.

Output

If the interchange of money is possible, output the message "`YES`" on the first line, one integer M ($0 \le M \le 10^6$) on the second line — the number of transactions in the transfering plan, and M integers on the third line — the numbers of the mathematicians who are involved in the transfering plan. Mathematicians are numbered starting from 1. You can output any plan of operations, not necessarily the shortest one.

If the interchange of money is impossible, then output only the message "`NO`".

Example

inter.in	inter.out
3 1 2 3 3 1 2	YES 2 3 2

Problem J. Jumper

Input file:	`jumper.in`
Output file:	`jumper.out`
Time limit:	1 second
Memory limit:	256 Mebibytes

In the far far future, a way to move faster than light by using black holes have been discovered. A jumper ship moves at superlight speed in accordance with Zeno's postulates.

1. About the discreteness
   If everything when it occupies an equal space is at rest, and if that which is in locomotion is always occupying such a space at any moment, the flying arrow is therefore motionless.
2. About the dichotomy
   That which is in locomotion must arrive at the half-way stage before it arrives at the goal.
3. About the casuality
   In a race, the quickest runner can never overtake the slowest, since the pursuer must first reach the point whence the pursued started, so that the slower must always hold a lead.

It follows from the third postulate that the principle of relativistic causality at superlight locomotion is not violated, and the jumper can not arrive earlier than the message of its departure sent on the radio before the start.

The first postulate implies that the superlight locomotion occurs discretely by jumps. One of the black holes is selected for moving, and as follows from the second postulate, after the jumping, the ship occurs exactly in the middle of the segment connecting the jumper's old position and the black hole.

The routes of ships are limited by a tetrahedron. Its vertices are found to be four black holes. Every jump requires a certain amount of energy and the ship has very small amount of energy.

In the end point, in order to moor jumpers to docks, tugs are used that operate on chemical fuel. The further the ship will be from the dock after a series of jumps, the more fuel is needed to towing the ship to the dock.

Write a program that selects a sequence of no more than K jumps which will lead the ship as close as possible to the end point.

Input

The first line of the input contains one integer K ($1 \le K \le 20$) — the maximal number of jumps the jumper can make. It is followed by four lines, each line contains three integers X_i, Y_i, Z_i from 0 to 10^3 — the coordinates of the black holes that are used for travelling. The black holes do not lie in one plane, and the maximal distance between pairs of black holes is not more than 1.5 times the minimal one. They are followed by 2 lines, each line contains three integers from 0 to 10^3 — the coordinates of the start and end points of the travel. The coordinates of these points lie strictly inside the tetrahedron whose vertices are the black holes.

Output

On the first line of the output print a single integer N ($0 \le N \le K$). On the second line print N integers from 1 to 4 — the numbers of black holes selected for jumping. If there are several variants of the shortest route minimizing the distance to the endpoint, you can print any of them.

Example

jumper.in	jumper.out
3 0 0 0 100 0 0 0 100 0 0 0 100 20 20 20 5 31 5	2 3 1

Problem K. K-th Element

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

There are two sequences sorted in ascending order. The first one contains N elements, the second one contains M elements. Assume that all elements in these sequences are different.

It is needed to create a program for a special device — a comparator, that must find K-th element in the sequence formed by merging two arbitrary increasing sequences of lengths N and M. The program should search for K-th element in minimal time, or, strictly speaking, minimize the number of comparisons of the elements in the worst case. In addition, the number of commands in the program must not exceed 10^5, otherwise the program will not fit in the memory of the comparator.

Write a program that generates a program for the comparator for given N, M and K.

Input

The first line of the input file contains three integers N, M, and K ($1 \le N, M \le 10^4$, $1 \le K \le N + M$).

Output

The first line of the output file contains a single integer P ($1 \le P \le 10^5$) — the number of commands in the program. Next P lines contain the program.

The program for the comparator can contain two types of commands:

- `C` i j l g
  Compare the i-th element of the first sequence ($1 \le i \le N$) with the j-th element of the second sequence ($1 \le j \le M$). If the i-th element is less than the j-th, then go to the command with number l ($1 \le l \le P$). Otherwise, go to the command with number g ($1 \le g \le P$).
- `R` p i
  Indicate that the K-th element in the merged sequence is the i-th element of the p-th sequence. This command terminates the program.

Program execution begins from the first command.

If there are several programs which find the K-th element in the least possible number of steps, output any of them.

Example

kth.in	kth.out
20 11 31	3 C 20 11 2 3 R 2 11 R 1 20