Problem A. King's Assassination

Input file: assassination.in
Output file: stdout

In Berland, a prosperous country, there are n nice cities, some of which are connected by one-way roads. Berland has a constant and long-standing enemy — Birland, with which it has been in war for many years. Birland's military leaders have understood that they can't defeat Berland by fair means, and they've chosen the way of sabotage. One of the planned acts of sabotage is to kill the King of Berland.

The King lives in his palace in the country's capital (city with index s), but it's impossible to penetrate into it by any means. However, they've got to know that soon the King is going to city with index t to inaugurate a chocolate factory. In city t unprecedented security measures have also been taken. So, it's not possible to kill the King there. The military leaders have found out that the King is going to stop in each city on his path to city t to speak to the people. The leaders came to the conclusion that during one of such speeches it's very convenient to assassinate the King. But the route of the King's trip is kept secret, thus, they don't know in which city to prepare the attempt on the King's life. Having thought for a while, they've taken the decision to choose one of the cities that the King is going to pass anyway. Your task is to find all such cities.

Input

The first line contains four integers n, m, s and t ($2 \le n \le 100\,000$, $1 \le m \le 300\,000$, $1 \le s, t \le n$, $s \ne t$). The following m lines each contain two integers x_i and y_i — indices of the cities connected by the i-th road in order of the road direction ($1 \le x_i, y_i \le n$, $x_i \ne y_i$). Each pair of different cities is connected by at most one road in each direction.

Output

In the first line, output the number of cities k which the King should pass anyway. In the second line, output k space-separated numbers — indices of these cities. Output the cities in increasing order of their indices.

Examples

assassination.in	stdout
4 3 1 4 1 2 2 3 3 4	2 2 3
4 4 1 4 1 2 2 3 3 4 1 3	1 3
4 5 1 4 1 2 2 3 3 4 1 3 2 4	0

Problem B. Command Post

Input file: post.in
Output file: stdout

Colonel Kruglyakovski has just finished his last campaign. He has established his command post in the middle of a nice lawn and now he wants to protect himself and his army from possible attacks. In order to do that several observation posts must be built. Kruglyakovski sent his best scouts to find suitable places for observation posts. They discovered n good locations, each of them on the same distance from the command post. Now the colonel has realized that he can't build n observation posts, as he has resources enough only for k. Kruglyakovski plans also to build a fence using observation posts.

Please help him to choose k observation posts in order to maximize the area protected by the fence. You may assume that the ground surface is flat, observation post size is negligible. The fence is a convex polygon with observation posts as its vertices. The command post must not lie strictly outside the area enclosed by the fence.

Input

The first line contains two integers n and k ($3 \le k \le n \le 1000$). The following n numbers represent polar angles of possible observation post locations. Angles are given on separate lines with at most four digits after the decimal point and ranged from 0 to 2π. All angles in the input are distinct. The command post is located at the origin.

Output

Print k different integer numbers ranged from 1 to n. They must represent indices of points used to build observation posts in the increasing order of the polar angle. If there are several optimal solutions, output any of them. Print 0 in the single line of the output if it's not possible to protect the command post by the fence.

Examples

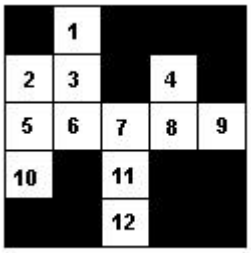
post.in	stdout
4 4 1.57 0 3.14 4.71	2 1 3 4
4 3 1.57 0 3.14 4.71	2 1 4

Problem C. Crossword

Input file: `crossword.in`
Output file: `stdout`

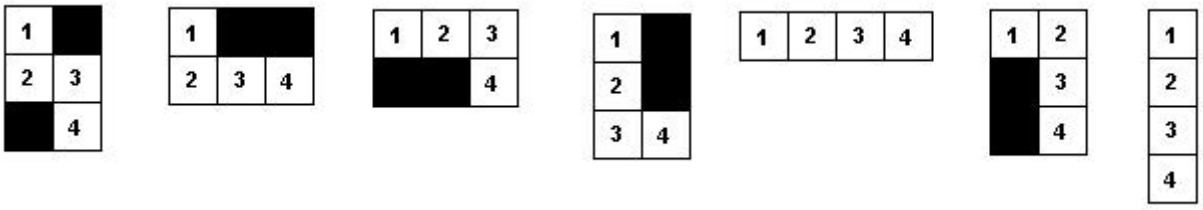
Petya loves crosswords, but he hates solving them. He has a huge collection of crosswords of the following type. Each crossword consists of a main across entry (a horizontal line) and a number of down entries (vertical lines) crossing the main entry. The number of down entries is always equal to the number of squares in the main entry. An entry can consist of one or more squares.

You will ask what does Petya do with his crosswords if he doesn't solve them. The answer is rather unexpected. He tries a new encoding on them that he invented himself. Petya takes a crossword of the described type, and he puts numbers from 1 to n into the squares (where n is the total number of white squares in the crossword). He puts numbers moving from top to bottom by the rows and moving from left to right in each row. Note that a number can be put only in a white square, black squares are ignored.



Then Petya writes a sequence of numbers obtained by moving first by columns from left to right and from top to bottom in each column. Thus he gets an encoding of the crossword. For example, for the crossword in the picture above he gets the sequence "2 5 10 1 3 6 7 11 12 4 8 9".

Now Petya is interested in decoding. Regretfully, the decoding can be not unique. For example, the sequence "1 2 3 4" corresponds to multiple crosswords (see the picture below).



However, Petya has the encoding, and he wants to get any correct crossword of the described type from it. If you find any sequence of numbers which can form the main across entry, Petya will be able to restore the whole crossword. Help Petya to decode the given sequence.

Input

The first line contains an integer n ($1 \le n \le 200\,000$) — the length of the encoding. The second line contains a permutation of integers from 1 to n. The integers of the encoding are separated by spaces.

Output

In the first line output an integer k — the length of the main across entry. The second line should contain k numbers that form the main entry. Output numbers in increasing order and separate them by spaces. It is guaranteed that at least one solution always exists.

Examples

crossword.in	stdout
12 2 5 10 1 3 6 7 11 12 4 8 9	5 5 6 7 8 9
4 1 2 3 4	1 3

Note

In the second sample, we choose the rightmost crossword in the picture as an answer. Obviously, the main entry has the length 1, but we can consider each of the four squares as the main entry. In such cases any correct answer is acceptable.

Problem D. Group Stage

Input file:	`group.in`
Output file:	`stdout`

Most of the modern football tournaments are organized in two stages. First, the group stage is played. At this stage, all the teams are split into groups, four teams in each group. Each team plays a match against every other team in its group. If a team wins a match, it gains three points. One point is given for a tie and no points for a loss. When all group matches have been played, the total score for each team is calculated. After that, all teams are ranked according to their score and two best teams from each group advance to the second stage of the tournament — the play-off round. If two or more teams have an equal score, secondary characteristics are taken into account (like goal difference, result of the match between the tied teams, etc.). If there is still a tie, the advancing teams are determined by the drawing of lots.

With such scheme of tournament, much depends on the situation inside the group and the teams which play in this group. Quite interesting things may happen — e. g., team X in group 1 can gain twice less points than team Y in group 2, and X will advance to the play-offs, but Y won't.

This can occur in the following situation.

Group 1 (teams: A, B, C, X):

- team A defeats all other teams and gains 9 points,
- team B loses to team A, but defeats team C,
- team C loses to team A, but defeats team X,
- team X loses to team A, but defeats team B.

As a result, after all matches have been played, team standings are the following:

- team A: 9 points,
- team X: 3 points,
- team B: 3 points,
- team C: 3 points.

Hitting more goals, team X advances to play-offs together with team A.

Group 2 (teams: E, F, G, Y):

- team E loses all the matches and gains 0 points,
- team F defeats E, but loses to G,
- team G defeats E, but loses to Y,
- team Y defeats E, but loses to F.

As a result, after all matches have been played, team standings are the following:

- team F — 6 points,
- team G — 6 points,
- team Y — 6 points,
- team E — 0 points.

The points of three teams are equal, but due to worse secondary characteristics, team Y does not advance.

In this problem we will consider a general situation. Let's define a group stage of some tournament with n teams in the group, m of which advance to the play-offs. Each pair of teams seeded in the same group will play a match, so totally $n \cdot (n-1)/2$ games will be played in the group. For simplicity, let's consider that in case of a tie all secondary characteristics are not taken into account and the advancing teams are chosen by the drawing of lots — that is, randomly.

You are required to find two numbers:

- the minimal score which team can obtain and still have the chance to advance to the play-off stage,
- the maximal score which still doesn't guarantee advancing to the play-offs.

Input

The first line of input contains two integer numbers n and m, separated by a space ($2 \le n \le 100$, $1 \le m \le n - 1$) — the number of teams in the group and the number of teams which advance to the play-offs.

Output

Output the two required numbers.

Examples

group.in	stdout
4 2	2 6

Problem E. Map Coloring

Input file: coloring.in
Output file: stdout

Berland Railway System consists of n stations, no two stations coincide. The stations are connected by two-way roads. It is known that one can travel from any station to any other using only Berland Railway System roads. Since the government is extremely greedy, the number of roads is equal to $n - 1$.

The world-famous Berland designer Artem Swan decided to design a new Berland Railway System map. He decided to paint each road with some color. Since any road must be clearly distinguishable from adjacent ones, any two roads sharing a common station must have different colors.

It is known that the cost of the printed map is the sum of the costs of printing each road. The cost of printing a road depends on its color: if the road is painted with the color t (colors are numbered, t is a positive integer), then the cost is equal to t.

Now Artem Swan is looking for such a way of coloring that the condition about different colors of roads around each station is satisfied, and the cost of printing such a map is minimal. He can use any colors, the colors are positive integer numbers.

Input

The first line contains n ($2 \le n \le 100$), where n is the number of stations. The following $n - 1$ lines contain descriptions of the roads. Each road is given as a pair of integers a_j and b_j which are the indices of stations connected by the j-th road ($1 \le a_j, b_j \le n$, $a_j \ne b_j$). The stations are enumerated from 1 to n.

Output

The first line should contain the required minimal total cost of printing the map. The second line should contain $n - 1$ numbers, the j-th number stands for the color of the j-th road. If there is more than one optimal coloring, output any of them. Use the order of roads from the input.

Examples

coloring.in	stdout
8 8 1 5 1 6 5 5 2 3 2 7 8 6 4	11 2 1 2 3 1 1 1
5 3 5 4 3 5 2 4 1	6 2 1 1 2

Problem F. Berland Bowling

Input file: bowling.in
Output file: stdout

Berland is a country where people like to reinvent. Bowling is not an exclusion, so Berland bowling has similar but different rules. Therefore, while solving this problem, forget about the real rules you may know and use the statement as a formal document.

Berland bowling is a sport in which players attempt to score points by rolling a bowling ball along a flat, usually wooden or synthetic surface into pins. The game consists of n base rounds. There are ten pins in each round. Two attempts are given to knock all ten pins. For each attempt, player gets a score which is equal to the number of knocked pins. There are three types of rounds:

- If a player knocks all ten pins on the first attempt, such round is called a *strike*. In a strike round, the second attempt is not used, because all the pins are already knocked. Strike gives an advantage for the player — the score of all attempts in the next round is doubled.
- If a player was able to knock all ten pins utilizing both attempts, it is called a *spare*. The advantage of the spare is less than the advantage of the strike — only the score for the first attempt of the next round is doubled.
- All other rounds are called *usual*. A usual round gives no score advantage.

If the n-th round was a strike, the *Bonus Round Rule* is triggered — player is awarded with one additional round, so totally $n + 1$ rounds are played in the game. Certainly, both attempts in the bonus round are doubled, because it was preceded by a strike. the Bonus Round Rule is not applied recursively — game ends even if the bonus round was also a strike.

Joe likes Berland bowling a lot. He spends much time for training. But still his skills are not so good as he wants them to be. He decided to enter the special Berland Bowling School to learn bowling strategy and tactics. To pass preliminary examination, Joe has to solve a difficult task. Joe was given a description of a bowling game with n base rounds. He was allowed to reorder rounds in the game, keeping the number of pins knocked in each attempt. Certainly, after reordering, the game should remain correct, i. e. the new game should contain bonus round if and only if the initial game contains it. Among all such games, Joe is asked to find the one which has the maximum total score. Not being good enough in maths and programming, he is asking for your help.

Input

The first line of the input file contains a single integer n ($1 \le n \le 50$) — the number of base rounds in the game.

Next n lines describe the rounds of the game. The i-th line contains two integers — the number of pins knocked in the first and second attempt of round i. Strike is described as `10 0`.

If the n-th round was a strike, the input file contains one more line — the description of the bonus round in the same format.

Output

Output should contain exactly one integer — the maximal score Joe can obtain after reordering the rounds.

Examples

bowling.in	stdout
2 5 2 10 0 3 7	44

Problem G. Orange

Input file:	orange.in
Output file:	stdout

The New Year celebration has passed. The New Year holidays are coming to an end. The same is true for food in Peter's fridge.

One day he opened his fridge to find something for his friends and found an orange. All of his friends wanted a part of the orange, so Peter decided to divide it into several parts. To do this, he took a knife and made several cuts passing through the center of the orange (we assume that the orange is an ideally spherical figure). Each cut divides the orange into two equal hemispheres. So you may think about a cut as a plane passing through the center of the orange.

After the cutting was completed, Peter became interested in the volumes of the parts. Help him to determine the volume of each part to distribute them fairly among friends.

Input

The first line contains two numbers r and n ($1 \le r \le 100$, $1 \le n \le 300$), the radius of the orange and the number of cuttings. Each of the next n lines contains three numbers x_i, y_i, z_i — coordinates of a vector orthogonal to a plane along which the cut passed. The center of the orange coincides with the origin. All coordinates do not exceed 200 by absolute value, $|x| + |y| + |z| > 0$. All numbers in the input are integer. No two cuts coincide.

Output

The first line of output should contain m — the number of parts. The following m lines should contain volumes of the resulting parts in non-descending order. The printed volumes must have an absolute or relative error less than 10^{-6}.

Examples

orange.in	stdout
1 1 3 4 5	2 2.0943951024 2.0943951024
10 2 0 0 1 1 0 0	4 1047.1975511966 1047.1975511966 1047.1975511966 1047.1975511966

Problem H. Puzzle Quest

Input file: `puzzle.in`
Output file: `stdout`

Puzzle Quest is a game played on $n \times m$ grid. As you are competing in a programming contest, you have to play this game on your own without any opponents, so please read the rules below.

Each cell of the grid is represented by one of the following characters:

- "`.`" — empty space
- "`R`" — red gem tile
- "`G`" — green gem tile
- "`B`" — blue gem tile
- "`Y`" — yellow gem tile
- "`D`" — diamond tile
- "`M`" — coin tile
- "`*`" — skull tile
- "`#`" — red skull tile

Each move goes as follows:

- You move by swapping two horizontally or vertically adjacent tiles, or moving a tile to a neighboring empty cell. A move can be performed only if an explosion happens right after the move (see the next bullet for the definition).
- After the move is done, the grid is evaluated for explosions. If there are 3 or more identical tiles located next to each other in a row or in a column, all of them are marked for an explosion. Then, after then grid scanning is finished, an explosion happens destroying all tiles at once which were marked for removal.
- After the explosion, some tiles fall straight down, occupying the next available space within their respective columns.
- The previous "Explode" and "Fall" steps are repeated until the grid becomes stable, i. e. no more explosions are detected.

There is one type of tile in the game which is considered special. It's a very powerful red skull tile. If a red skull tile located at cell (r,c) participates in an explosion, it also explodes up to eight additional neighboring tiles with coordinates $(r-1,c-1)$, $(r,c-1)$, $(r+1,c-1)$, $(r-1,c)$, $(r+1,c)$, $(r-1,c+1)$, $(r,c+1)$, $(r+1,c+1)$. Also, a red skull tile is considered identical to the regular skull tiles when doing matching on the grid, meaning if there are 3 or more skulls located next to each other, they will explode irrespective of which of them are red and which of them are regular. Note — an explosion of a red skull might potentially trigger an instant explosion of other red skulls not participating in the initial explosion (e. g. a red skull explodes its neighbor by diagonal, which is also a red skull), causing them to explode according to the same rules. You may assume that the chain of all possible explosions happens instantly, and the tiles start to fall down only after all explosions happen.

Given an initial configuration of the grid, you have to find a sequence of moves for the puzzle that will either completely clear the given grid from tiles, or leave only some number of red skulls on the grid.

Input

The first line of input contains two integers n and m — the number of rows and columns respectively. Each of the next n lines contains m characters describing the cells. Each character will be from the set "`.RGBYDM*#`".

Here comes a surprise! All test cases are known to you in advance. It's guaranteed that your submission will be judged only against the given set of test cases. All these test cases can be generated by the program which is given to you. And even more, you have a choice of either solving these test cases manually or writing a program to solve them. In all test cases both n and m are equal to 8. There are 54 tests in the problem.

First of all, you are given a test generator in a file "gentest.jar". You may download it by the link "`http://acm.sgu.ru/download/puzzle/gentest.jar`". You can run it with no parameters to get help on the usage, but the typical execution line looks like:

```
java -jar gentest.jar --test <test-id> --output <output-file>
```

For example, the following command will write 11-th input test case to the file 11.in in the current directory:

```
java -jar gentest.jar --test 11 --output 11.in
```

Also, you're given a player/visualizer in "visualizer.jar" so you can view the results of your work. You may download it by the link "`http://acm.sgu.ru/download/puzzle/visualizer.jar`".

The following command will read the input from <input-file>, read the answer from <output-file>, and will playback/validate your solution showing each step in GUI:

```
java -jar visualizer.jar --mode playback --input <input-file> --output <output-file>
```

The following command will read the input from <input-file> and run GUI for solving the input manually. You can use mouse for making moves. If <output-file> is specified and the input is successfully solved by hand, the answer will be written to the specified <output-file>:

```
java -jar visualizer.jar --mode manual --input <input-file> [--output <output-file>]
```

It's guaranteed that all test cases have a solution. If you are not sure about certain aspect of the rules, you are encouraged to use the provided programs to better understand the game mechanics. Visualizer is not guaranteed to work correctly with values of n and m other than 8.

Output

Print the sequence of moves which solves the puzzle to the only line of output.

Represent each move as a triple d,r,c where r is a zero-based row index, c is a zero-based column index of the tile being moved, and d is a direction (one from "`LRUD`"). The statement is already long by itself, so please see the notes to sample input/output if you still have questions about the output format.

If there is more than one solution, you may print any of them.

Examples

puzzle.in	stdout
8 8 ....#. ......YG .G....Y* GB....G* YB....*Y BY....R* YB.G.GR#	R60L73R56D66

Note

Here is an explanation of the sample input/output.

Move R60 swaps yellow and blue gems, generating two explosions (3 yellow gems explode in column 0, and 4 blue gems explode in column 1). After that two green gems fall down to the bottom and the field becomes:

```
........
......#.
......YG
......Y*
......G*
......*Y
......R*
GG.G.GR#
```

Then move L73 moves green gem to an empty space, exploding three green gems in a row (note that R72 is not a valid representation of this move):

```
........
......#.
......YG
......Y*
......G*
......*Y
......R*
.....GR#
```

Then, the move R56 swaps regular skull and yellow gem, generating an explosion out of 5 skulls. One of these skulls is a red skull, so it also explodes two neighboring red gems. After the tiles fall down in the last two columns, the field becomes:

```
........
........
........
......#.
......Y.
......Y.
......G.
.....GYG
```

Finally, move D66 swaps green and yellow gems, exploding the last three green and the last three yellow gems on the field. The last red skull falls to the bottom and the puzzle is solved.

Problem I. Lies, Damned Lies and Statistics

Input file: `statistics.in`
Output file: `stdout`

As you know, there are three kinds of lies: lies, damned lies and statistics. Journalist Bob needs to urgently find some material for the front page of "The Berland Times" newspaper. Any discovery, a celebrity scandal or anything of such kind will do. Unfortunately, nothing sensational happens in Berland, apart from a series of mysterious burglaries of ice-cream kiosks.

Journalist Bob decided to strike the public with an unexpected discovery — statistically all the burglaries lie on one line! He marked the burglaries as points on the map. Now he needs to draw such a line, that maximum amount of the points is close to it. Bob decided to regard as close to the line only those points that lie at most at distance d from the line. The amount of such points will be a crushing argument in support of his theory!

Help Bob to find out the maximum amount of points on the map that lie at most at distance d from some line.

Input

The first line contains a pair of integers n and d ($1 \le n \le 1000, 0 \le d \le 1000$). The following n lines contain the coordinates of the points. Each of these lines contains a pair of integers x_i, y_i ($-1000 \le x_i, y_i \le 1000$) — coordinates of the i-th burglary. No two burglaries happened in one point.

Output

Output the required maximum amount of points.

Examples

statistics.in	stdout
4 3 0 0 5 5 5 0 0 5	4
4 2 0 0 5 5 5 0 0 5	3

Problem J. Taxi

Input file: `taxi.in`
Output file: `stdout`

At one of the famous night clubs, a party has just finished. A group of n boys and m girls was there. They've enjoyed themselves, but now it's time to go home. Everybody is tired, and it's rather late, that's why they want to go home in a taxi. It occurred that all the young people live on the same street, and it's the street where the night club is! The reason is that there is one central street in that city. All the most stylish city establishments are situated in one part of this street, the other part is where the most stylish city residents live. So, all the young people need to go in the same direction from the night club. Thus, they could save a lot of money (going to stylish night clubs is a costly affair).

One taxi can take from one to four people. A taxi drives to the house of the passenger who lives the farthest from the night club, and on its way, it drops off the other passengers at their houses. Taxi fees are one berllar per one kilometer. Local taxi drivers are hot southern guys, that's why the young people agreed that at least one boy should get into each taxi (but it is *not* required for him to be in the taxi during the whole trip).

Find a way for the young people to divide and get into the taxis so that the total sum of money paid to the taxi drivers is minimal.

Input

The first line contains integer n ($1 \le n \le 2011$) — the number of boys. Then follow n lines, each containing a boy's name and the distance from the night club to his house in kilometers. Next follow the number of girls m on a separate line, and their descriptions in the same format ($0 \le m \le 2011$, $m \le 3n$). The boys' and the girls' names consist of Latin letters, the first letter of a name is capital, the rest are lower-case. Each name has length from 1 to 15 characters. No two names coincide. Distances are non-negative integers not greater than 10^4.

Output

On the first line, output the minimum total amount of money paid to the drivers by the group members. On the second line output k — the amount of taxis they should take. In the following k lines output who goes in which taxi. Study the output format in the sample tests. You can output the taxis and the people in each taxi in any order. If the answer isn't unique, output any.

Examples

taxi.in	stdout
2 Anton 5 Maxim 10 5 Anna 1 Maria 12 Tanya 10 Elena 8 Marina 6	18 2 Taxi 1: Maxim, Maria, Tanya and Elena. Taxi 2: Anton, Marina and Anna.
1 Romeo 100 1 Juliet 200	200 1 Taxi 1: Romeo and Juliet.
1 Jack 17 0	17 1 Taxi 1: Jack.