Problems

Problem 1 – Merging IP ranges

Some specialized programs fight advertising and spyware by blocking incoming and outgoing connections using IP blocklist files (also known as blacklists.) A blocklist file contains a simple sequence of IP ranges that usually gets very long. Blocklists with more than 250,000 lines are not unheard of. Here is the initial segment of a blocklist file:

```
3.0.0.0-3.255.255.255
4.0.25.146-4.0.25.148
4.0.26.14-4.0.29.24
4.2.144.64-4.2.144.95
4.2.144.224-4.2.144.231
4.2.144.248-4.2.144.255
4.2.145.224-4.2.145.239
...
```

Some IP blocking programs offer the ability of merging multiple blocklists into optimized blocklists, which are as compact as possible. However this functionality tends to be buggy in the first releases of many of those programs because it requires some effort to come up with a merging algorithm that is both accurate and efficient. This will be one of your tasks for today.

Task

Please, write a program that takes a sequence of IP ranges (unsorted and coming from several sources) and produces an equivalent result sequence, sorted and with a minimal number of lines. Note that given the input sequence, the corresponding result sequence is unique.

For testing purposes, your program should also support the option of producing the length of the result sequence, a mere integer, instead of the entire result sequence.

Input

The program gets its input from the standard input stream. The first line of the input contains a single upper-case char, indicating the intended output:

- S requests the entire result sequence;
- L requests only the length of the result sequence.

A sequence of IP ranges with up to 300.000 lines follows, with one IP range per line.

Output

The program writes its output to the standard output stream. The output may be a sequence of IP ranges in lexicographical order or a single non-negative integer, depending on what has been requested in the input.

Sample Input

```
S
3.201.0.0-3.255.255.255
4.0.25.146-4.0.25.148
4.0.26.14-4.0.29.24
4.2.144.64-4.2.144.95
3.0.0.0-3.200.255.255
4.2.144.224-4.2.144.231
4.2.144.248-4.2.144.255
4.2.145.224-4.2.145.239
3.100.255.0-3.110.111.111
```

Sample Output

```
3.0.0.0-3.255.255.255
4.0.25.146-4.0.25.148
4.0.26.14-4.0.29.24
4.2.144.64-4.2.144.95
4.2.144.224-4.2.144.231
4.2.144.248-4.2.144.255
4.2.145.224-4.2.145.239
```

Problem 2 – Dialogue among Civilizations

“Dialogue among Civilizations” is a theory in international relations. In its current form, it was first introduced by Mohammad Khatami, former President of Iran (http://en.wikipedia.org/wiki/Dialogueue_Among_Civilizations).

Mr. Khatami wants to start a universal conversation, but he does not know in which sequence he should do it, in order to attain the best possible results. Since dialogue with a nation can affect the relations with others, he asks you and a civilization analyst to assist him in solving this problem in the most effective way.

After long research, the analyst found a model for civilizations relationship. The model is a complete weighted graph, where the nodes represent the civilizations and the weights on the edges represents the *level* of relationship between the civilizations at the vertices. Negative weights indicate a conflict and positive weights show a friendly, peaceful relationship. Figure 1 displays such a graph, with four civilizations.

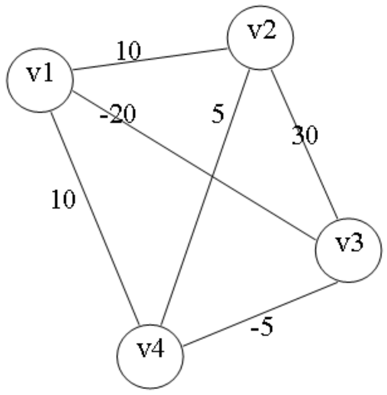


Figure 1

In this case, civilization v1 is in good terms with v2 and v4 but has a bad conflict with v3. Civilization v2 is in good terms with all the others, and v3 and v4 have a minor conflict.

At first, before any dialogue among civilizations is carried out, the weights on the edges are between -100 and 100. Note that the graph is complete. Therefore, a graph with *n* vertices has *n*(*n*-1)/2 edges. In Figure 1, the number of vertices is 4 and the number of edges is 6.

When two civilizations *i* and *j* start a dialogue, the weights in the graph change according to the following formulas:

n : number of vertices in graph

$1 \le i, j \le n$

$$\forall k : 1 \le k \le n \wedge k \ne i \wedge k \ne j \Rightarrow e'_{ik} = e'_{jk} = \min\{e_{ik} + e_{jk}, 1000\}$$

for other edges : $e'_{ij} = e_{ij}$

For instance, if nodes 1 and 4 start the dialogue, the graph in figure 1 will change to left graph in figure 2. As you see this condition is worse than the first graph. The negative relations have greater absolute values, and this means more conflict. On the other hand, if node 1 and 2 start a dialogue, the graph in figure 2 will change to right graph in figure 2. As you see this condition is better, since there are more friendly relations than in the first graph.

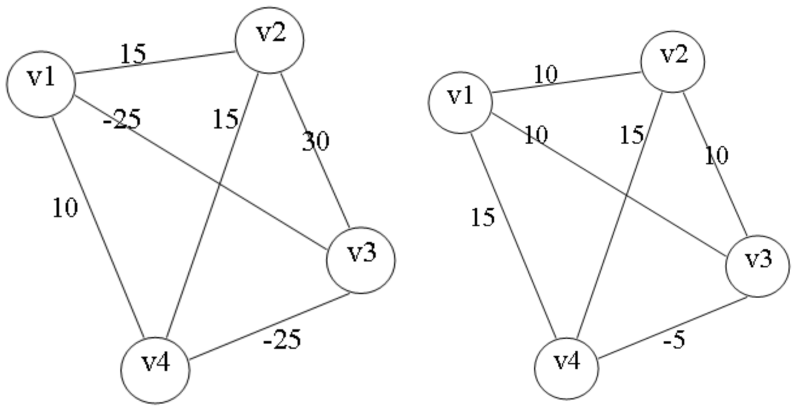


Figure 2

Now, let’s assume that Mr Khatami is located at node 1. Can you guide him in choosing the nodes with which he should start a dialog, and in which order, so that all the relationships among civilizations become friendly, and the world reaches universal peace? Recall that friendly relationships are described by positive weights.

For instance, there are two ways to solve the graph in figure 1:

1. Mr Khatami (node 1) first speaks to node 2 and then speaks to node 3. This is illustrated in figure 3.

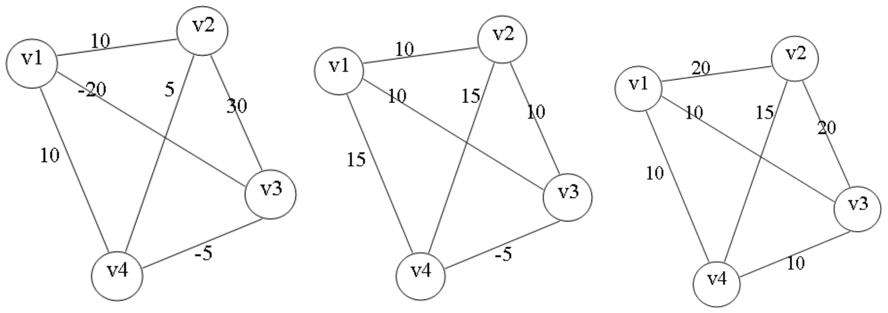


Figure 3

In figure 3, the left graph shows the initial graph, which is the same as figure 1. The graph in the middle shows the situation after dialogue between node 1 and node 2. The right graph shows the situation after the second dialogue, between node 1 and node 3. This graph displays universal peace, since all weights are positive.

So, in this case, peace is possible and the sequence (2, 3) is one solution.

2. Mr Khatami (node 1) first speaks to node 2 and then speaks to node 4. This is illustrated in figure 4.

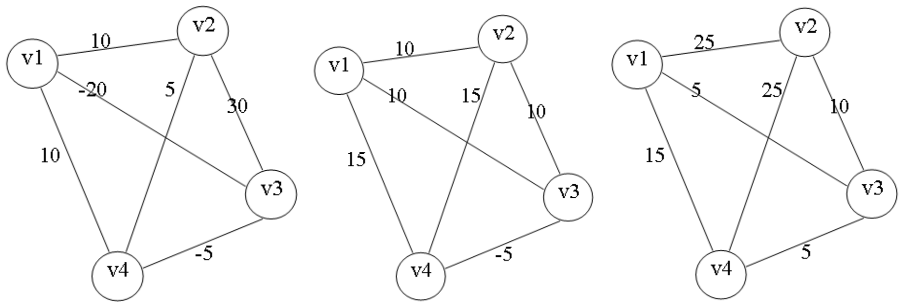


Figure 4

This means that the sequence (2, 4) is also a solution.

As you can see, in this case we can reach universal peace in more than one way.

Task

Please, write a program to solve the dialogue among civilizations. The program inputs a description of the graph, in the format described below, and outputs the message “yes” followed by a solution, if universal peach is possible, or the message “no” only, if it is impossible to reach universal peace.

Your program should solve a graph with number of nodes equal to countries in United Nations. (How many are there?!)

Input

Your program is to get the file name as an argument in the command line. The file contains the description of the graph, as a square matrix. The first line of the file contains a positive integer, *n*, representing the number of nodes. Each of the remaining *n* lines contains a list *n* numbers, representing a row of the matrix.

Output

If the graph is solvable, in the sense that there is a way to reach universal peace, the program output two line, on the console. The first line is the message “yes”, without the quotes, and the second line is the sequence of nodes with which to carry the dialogue, separated by single spaces. If universal peace is not possible, the program write “no”, without the quotes.

Sample Input 1

The graph in figure1 is represented by the following file:

```
4
0      10     -20    10
10     0      30     5
-20    30     0      -5
10     5      -5     0
```

Sample Output 1

The output corresponding to the above input example can be the following:

```
yes
2 3
```

It can also be:

```
yes
2 4
```

Sample Input 2

Here are two more examples:

```
3
0 -10 -10
-10 0 -10
-10 -10 0
```

Sample Output 2

```
no
```

Sample Input 3

```
3
0 10 -20
10 0 30
-20 30 0
```

Sample Output 3

```
yes
2
```

Limits

Your program should be able to handle a graph with all the countries in the world.

Problem 3 – S language interpreter

The word "algorithm" has no general definition separated from a particular language (That is why famous Church's thesis cannot be proved as a mathematical theorem).

What is then the simplest programming language powerful enough to express any algorithm? Such a language was introduced in "Computability, Complexity and Languages", Martin Davis, Elain Weyuker, Academic Press, 1983, page 15, and is called language "S" (S for "Simple", maybe). It is proven that "S" language is equivalent to Turing machine, i.e. every Turing's machine algorithm can be expressed as an "S" program, and vice versa.

Task

Your task is to write an interpreter for a variant of S language.

Syntax

The syntax and semantics of the language will be stated using a non-formal description.

“S” program consists of sequence of statements separated by “;” (semicolon).

Statements have the form:

[statement_label] statement_body

statement_label is a unique sequence of one or more alphanumeric characters of which the first is a letter. The *statement_label* is optional; when present must be put in brackets, as shown above.

statement_body is one of the following four types of statements:

var=const
var++
var--
var?statement_label

where:

- *var* is a variable name - any alphanumeric sequence beginning with a letter. Variables may have non-negative integer values of any size. By “any size” we mean that integer values are not restricted to 32-bit or even 64-bit integers.
- *const* is an non-negative integer of any size,
- ++ is arithmetic operation of adding 1 to an integer,
- –– is arithmetic operation of subtracting 1 from a non-negative integer. Note: subtracting 1 from 0 gives 0.
- *var?statement_label* is a statement in which variable *var* is tested against 0. If value of *var* is not zero program continues execution with statement labeled *statement_label*, otherwise the next statement is executed.

Space and whitespace characters are ignored.

At the start of program execution, all the variables are initialized to 0.

Program starts the execution with the first statement. Program ends the execution when it reaches the end of the list of statements or when it is directed to a non-existing statement label. If, during the execution, the interpreter encounters a syntax error it will print the message “error in line n” and stop the execution.

Note: Remember that execution of a statement of the form *var?statement_label*, where *statement_label* does not exist in the current program will stop program execution. This is treated as “normal” execution, just like one where last statement was encountered.

Upon the “normal” execution of the program, the interpreter should print the final values of all variable mentioned, one variable per line, in lexicographical order of the names of the variables.

The interpreter should receive the input file via the standard input, and write results on standard output.

Note: You may be now wandering on how it is possible to have a universal programming language of such simplicity. The only ability you need to express and to execute any algorithm is to know how to count up (*var*++) and down (var––) and to test if something is null. Amazing and wonderful, isn’t it?

Sample Input

This program to “copy” the value in variable X to variable Y.

```
      X= 12345;
      Z1++;
      Z2++;
      Z3++;
[A]   X?B;
Z1?C;
[B]   X--;
      Y++;
      Z++;
      Z2?A;
[C]   Z?D;
      Z3?E;
[D]   Z--;
      X++;
      Z4?C
```

Sample Output

```
X 12345
Y 12345
Z  0
Z1 1
Z2 1
Z3 1
Z4 1
```

Additional questions

1. Write and submit an S program to sum integers X1 and X2 whose values are given as constants. Test with 9999999999999999999999 + 999.

2. Write and submit an S program to subtract integers X1 and X2 whose values are given as constants. Test with 1111111111111111111111 – 9999.
3. What is the output of the following "S" program:

```
      X = const;
[A]    X?B;
      Y++;
      Y?E;
[B]    X++;
       X?A;
       Y++;
```

   Discuss for different values of const (and submit your discussion).
4. Extend "S" language with an input statement of the form:

   input var,var,…,var

   The input statement is used for accepting values (comma separated) for the variables from the standard input (console). Submit your new interpreter.

Problem 4 – Road Construction

You are in charge of constructing a new road in StrangeWorld. The road will be a straight line. To pay you for your exceptional efforts, the local government has decided do offer you some terrain adjacent to the road, in the form of a determined number of rectangular parcels. You have the ability to choose the specific parcels you want and of course you want to maximize the area of land you will receive (after all, you can then build some gas stations, hotels and even a nice farm for you and your family).

StrangeWorld is a grid. The road is a vertical line occupying all squares with *X* coordinate zero. The available terrain extends from -*W* to *W* in terms of *X* coordinate (where we call *W* the width), and extends from 1 to *H* in terms of *Y* coordinate (where we call *H* the height). There are some houses on the terrain and when you choose the parcels you must take in consideration the following:

- The rectangular parcels must always include the road (i.e., you cannot choose a rectangle "outside" the road);
- The rectangular parcels must not include grid cells with houses;
- The rectangular parcels should be consecutive and complete (i.e., they must cover all rows, from 1 to *H*).
- You must always use all parcels (i.e., if you are given 4 parcels, you can not use only 2 or 3).

Figure 1 gives an example terrain (with width 8 and height 18), and shows in color the optimal allocation if you are given exactly 3 parcels.

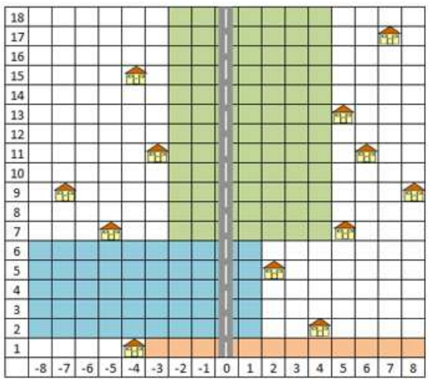


Fig. 1 - Example terrain and optimal allocation for 3 rectangular parcels

Task

Given a terrain (including the locations of the houses) and the number of rectangular parcels you are offered by the government, you must discover the way to allocate the parcels that maximizes the area of terrain you receive. The allocation of the parcels has the constraints defined above.

Input

In the first line of input come two integers **H** and **W** representing respectively the height and width of the terrain to consider ($5 \leq H \leq 1000$ *and* $5 \leq W \leq 5000$).

The second line of input contains a single integer **R** indicating the number of rectangular parcels you must allocate ($1 \leq R \leq 100$).

The third line of input contains a single integer **N** indicating the number of houses in the terrain ($0 \leq N \leq 5000$). Then follow exactly **N** lines, each one with integers indicating the Y and X coordinates of a single house in the terrain. The houses can come in any order and there may be more than one house in each row (or column).

Output

The output should consist of a line containing a single integer indicating the largest area you can achieve by allocating N rectangle parcels that obey the restrictions given above. The area is measured in number of grid cells, and it includes the road itself.

The sample input and output correspond to the example in figure 1.

Sample Input

```
18 8
3
12
1 -4
2 4
7 5
9 -7
11 -3
11 6
5 2
13 5
15 -4
17 7
7 -5
9 8
```

Sample Output

```
146
```

Problem 5 – Escaping route

Miners are trained for the possibility of an accident. Part of their standard equipment is a PDA with a digital map of the mine. This map is given as a matrix of size $n \times n$, where each cell contains a number:

- 0, meaning that the cell is empty and the miner can pass without effort.
- $x>0$, meaning that there is a wall in that cell. In order to destroy this wall and pass through the cell, x sticks of dynamite are needed.

Miners also carry with them *b* sticks of dynamite, all the time.

In case of an accident, each miner has to escape as fast as possible from his present position to the exit of the mine. They have been instructed to look for the shortest route to the exit. For that, they may destroy some of the walls, using the sticks of dynamite.

The route may have only horizontal and vertical movements between the cells.

Task

Please, write the program to guide a lone miner from his current position to the mine exit. More precisely, your program must compute the shortest path that leads from the current position by using the empty cells and by breaking wall, using at most the b

Input

The program should read the input file from the standard input. In the first line of the file there are 6 integers: the map size, *n*, the number of sticks of dynamite *b*, the coordinates of the current position, *xi*, *yi*, and the coordinates of the exit position, *xf*, *yf*. The input file has *n* more lines, each one with *n* numbers. Each number represents the value of the corresponding cell in the map (zero if it is empty, *x*>0 if *x* sticks are needed to break the wall). In this map, the first row is row 0, and the first column is column zero. This means that the upper left corner has coordinates (0, 0) and the lower-right corner has coordinates (*n*–1 1, *n*–1).

Output

The program should write the result of the computation on the standard output. The first line will contain the length of the route in number of moves, *m*. The following *m*+1 lines contain two numbers each, separated by a space, representing the coordinates of the cells through which the route passes, one cell per line. The starting and the final points are part of the route, and should both be displayed, if different. If there is no route, the length will be displayed as –1.

Sample Input

```
5 2 0 1 4 4
1 0 1 0 1
0 1 0 2 0
1 0 1 0 2
0 2 0 1 0
0 0 0 0 0
```

Sample Output

```
7
0 1
1 1
2 1
2 2
3 2
4 2
4 3
4 4
```

Problem 6 – Focus List

The *focus list* is a list of financial instruments (shares and indexes) the trader must pay special attention to: the trader has to mainly monitor the volatility and dividend table changes.

Years ago, we developed under Windows a small C++ command prompt application that lists the focus instruments. At that time, the number of focus elements was less than 30. Currently the list is more than 2,000 elements and the pressure from the trader to improve the application is everyday more insistent.

After talking with the traders, we extracted the following function to implement: the trader would like to dynamically filter the list on the instrument name, the Reuters and/or Bloomberg code with characters and some wildcards: '*' for multiple characters and '?' for single character). Furthermore, the trader wants to use three other special characters, as follows:

- '~' to display the previous page;

- '#' to display the next page;
- '^' to reset the filter string.

Task

Your task is to adapt the current software with the above demand. This software is provided to you, "as is".

Input

A list of 2,000 focus instruments is also provided.

Further Requirements

Your program must compile under UNIX and under Windows using the ANSI Standard.

Please note that you are not to write a new application from scratch. On the contrary you should reuse the available code whenever possible and reasonable.

As the windows workstations of the traders have from one dual core up to 2 quad cores processors, we expect an ad hoc use of open mp in the solution.

We would also that you supply the following documents together with your solution:

- A small analysis of the implementation.
- A few examples illustrating to the trader the functionality of your solution.
- A small report showing the advantage of a multi threaded solution *versus* a single threaded one.

Problem 7 – Snapshot

Usual search engines receive a set of keywords and look for all the documents that contain these keywords. The documents are listed in the order of document significance. In this problem we consider the significance of a document for a set of keywords is given by the minimum number of words of the continuous piece of text that contains all the searched keywords.

For instance: consider the keywords "2008" and "IEEEXtreme", and the following two texts: "The registration for the 2008 edition of IEEEXtreme is now open" and "IEEEXtreme 2008 edition is going to take place on March 8th 2008". The significance of the first text is 4, and of the second one is 2. If any of the given words is not present in the text, the significance is zero.

Task

Please write a program that reads from the standard input a text in which the words are separated only by spaces, and finds the significance of text against the keywords given as the parameters to your program.

Example

For the input text:

```
The registration for the 2008 edition of IEEEXtreme is now open
```

your program executed as:

```
> snapshot 2008 IEEEXtreme
```

should write 4 on the standard output.

Note: if not all the words are found, the program should return 0.

Problem 8 – GPS Track Area

GPS is a satellite-based system for determining your position an earth. Apart from that, it is also used for tracking. This means some software logs the GPS data and the user can afterwards analyse the data. One application is to compute the area, which is covered by the track.

Task

Please, write a program that computes the area of a given GPS track. The data is in the NMEA 0183 format and is fed to the program using standard input.

You should assume the following:

- the start point and the end point of the track are connected;
- the altitude as irrelevant;
- the track as one single polygon (loops and interruptions do not need to be handled);
- the sentence GPGGA is in the tracking data;
- the earth is a sphere with radius 6,378,137 meters;

The result is the area in square meters, which has to be printed to the standard output. If an error happens during the computation, your program should print "error".

Examples

For example, if goodlog is valid logfile, describing a track whose area is 324 square meters, your program should run as follows:

```
gps2area < goodlog
324
```

If badlog is an invalid logfile, we should get “error”:

```
gps2area < badlog
error
```

Problem 9 – Secret Agents

Recently Qubitland made big progress in its war against Byteland. Using spy satellites it managed to create a map of all Byteland's bases. Now it wants to destroy all of them as fast as possible. Each base has to be destroyed by aQubitland agent. Since, in general, there are more bases than agents, some agents will have to destroy more than one base. Qubitland sends an aircraft with the agents to the enemy territory and at any moment one or more agents can jump out of the plane. After landing with parachutes on the ground, the agent starts his mission. He goes to the first base that has been assigned to him and destroys it. After destroying it he can go to another base and destroy it too and so on.

Task

Please, write a program to schedule the mission for the plane and agents. Each agent has its own moving speed and time he needs to destroy the base. You can assume that the airplane cannot be destroyed and it is flying with constant speed. It can take off and land only once but after starting it has enough fuel to fly as long as it is needed. You can also assume that the terrain is flat and has no obstacles (like rivers, fences or precipices) so agents can go everywhere. Qubitland wants to complete the mission as fast as possible so the shorter the schedule you create, the higher the score you will receive.

Input

In the first line of the input file there are three numbers **x, y, v** – coordinates of the airport where the plane is at the beginning of the mission and speed of the plane. In the second line there are two numbers **m, n** – the number of bases and number of agents respectively. In the third line there are **m** pairs of numbers – coordinates of bases. After this line there are **n** lines. Each line contains one string consisting of up to 10 alphanumeric characters – the name of the agent and two numbers – the speed of the agent and time that he needs to destroy the base. All coordinates are double numbers greater or equal 0.0 and less or equal 10000.0. All speeds and times are double numbers greater or equal 1.0 and less or equal 1000.0. To travel from point x1, y1 to point x2, y2 with the speed v the agent spends the time computed by the following formula:

$$t_{12} = \frac{\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}}{v}$$

Time is measured in hours. Distance is measured in some units. You can assume that time between jumping out from the plane and landing is 0.0.

Output

The output should consist of two parts, separated by an empty line. The first part describes the mission of the plane and consists of two kinds of lines:
FLY x y – **x** and **y** are coordinates of the next target of the plane. They should be double numbers and have no more than 6 digits after decimal point. Note that the target does not have to be directly above one of the bases.
OUT name – name is the name of the agent who should jump out at the coordinates given by the last **FLY** line before this line. Note that more than one agent can jump out at the same coordinates.

The second part describes the missions of the agents. There should be one line for each agent who jumped out from the plane. Some agents can stay in the plane to the end of the mission – then there should be no lines for them. Lines can be in arbitrary order. For each agent you should output his name and the list of bases that this agent should destroy. The agent will first go to the first base on the list, destroy it, then go to the second, destroy it and so on. The number of the base is defined by the position of its coordinates in the input and starts with 1. At the end of the list there should be -1.

Sample Input

```
10.0 0.0 20.0
3 2
10.0 20.0 20.0 20.0 50.0 50.0
A005 2.0 1.0
A007 1.0 5.0
```

Sample Output

```
FLY 10.0 20.0
OUT A005
FLY 50.0 50.0
OUT A007

A005 1 2 -1
A007 3 -1
```

According to this output the plane should first fly to (10, 20). It takes 1.0 hour and then agent A005 should jump out and plane should fly to (50, 50). The flight takes 2.5 hours so agent A007 will leave the plane in 3.5 hours after start. A005 starts his mission after 1.0 hour from start, spends 1.0 hour on destroying the base 1, then 5.0 hours walking to the base 2 and again 1.0 hour on destroying it so he ends his mission after 8.0 hours. A007 starts his mission 3.5 hours from start and spends 5.0 hours on destroying base 3 so he ends it after 8.5 hours. The time when the last base is destroyed is 8.5 so this is the score for the solution. For this particular case the better solution does not exist.

Judging

With this task we provide 10 test cases. You should write a program that for each test generates the mission schedule which is as short as possible. The length of the mission is the time from the taking off of the airplane after which the last base will be destroyed. You do not have to care about what happens with agents after completing their missions. This problem is very hard so you do not have to find the best solution but you should find as good solution as you can. After the end of the contest for each test for which you generated a correct output the number of points will be calculated in the following way:

$$p = \left(1 - 0.8 * \frac{S - S_{min}}{S_{max} - S_{min}}\right) * M$$

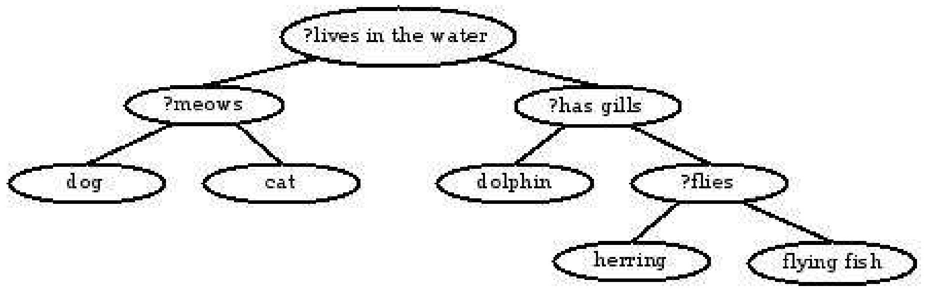
Where p is number of points, S is your score, S_{min} is score of the best participant, S_{max} is score of the worst participant and M is maximum number of points for the test. The score is equal to the length of the schedule in hours.

For each input file you should generate the output file with the same name but its extension should be ‘out’ instead of ‘in’. You can use provided program (binaries for Windows, MacOS and Linux) to check if your output file is correct (but not to score it).

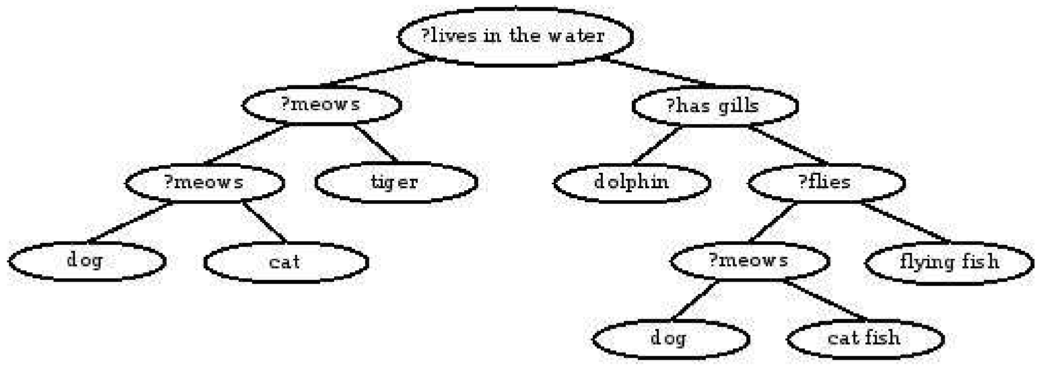
Problem 10 – Consistency of binary decision trees

Decision trees are an important technique used in knowledge representation, classification systems and AI computer games. In a decision tree, each branch node is called a decision node and represents a choice between some alternatives. As for the terminal nodes, each one represents a particular classification or a particular entity.
A decision tree is said to be binary if each decision node signifies a choice between exactly two alternatives, i.e., a YES/NO question. In this problem, it is assumed that each decision node causes an actual split in the domain of entities, meaning that every decision node must have exactly two children.

To expound, the following binary decision tree represents some knowledge about animals. At each decision node, the left sub-tree represents the NO answer, and the right sub-tree represents the YES answer.



A decision tree is said to be inconsistent if some contradiction could be derived from it. Here is an example of an inconsistent decision tree:



There are two contradictions in this tree: (1) the cat meows and does not meow; (2) the dog lives in the water and does not live in the water.

Decision trees are usually built in a way that ensures consistency. However, checking the consistency of a binary decision tree is an interesting problem by itself. The definition of inconsistency can be directly applied; however the careful analysis of the circumstances that make a tree inconsistent may provide a simpler solution to program.

Task

Please, write a program to check the consistency of a binary decision tree.

Input

The program gets it input from the standard input stream. The input represents a binary decision, defined using the following recursive rules:

- The root of the tree is the first line;
- The left sub-tree is in the following lines;
- The right sub-tree is in the following lines.

Output

The program writes its output to the standard output stream. The output is a line with a single word. The possible results are: CONSISTENT, INCONSISTENT.

Sample Input

```
?lives in the water
?meows
dog
cat
?has gills
dolphin
?flies
herring
flying fish
```

Sample Output

```
CONSISTENT
```

Further requirements

Your program in this problem is to be written in Algol60, using the RHA Algol60 V5.5 compiler for MSDOS, available at http://www.angelfire.com/biz/rhaminisys/algol60.html. This compiler runs on Windows.

Algol60 is a fundamental historical language we all have heard of, but in which most of us have never programmed. Well, it’s time to fill that gap in our resumés: from now on, we may add Algol60 in the list of the languages that we have mastered. This will certainly create a deep impression on our prospective employers.

Problems

Problem 11 – NASA ducks

Cathy is writing code for NASA project. Since she doesn't want the spaceship to Mars to hit any flying ducks on its way, she added a lot of asserts to her code.

Dan is now code-reviewing part of the code. Help Dan debug Cathy's code by finding an input for which her program will trigger one of the assert statements.

Cathy's code is provided.

Task

Your task is to find and submit an input for which the provided program will trigger one of the assert statements.

Problem 12 – Cynical Times

It is the year 2073, and the Eurasian Coalition is planning (again…) to free the world. However, since around 2007 (Bali Conference) there has been extreme care in doing whatever possible not to harm the environment during warfare.

A new, ecologically efficient, bomber is being developed, the B-73-ECO. It is lightweight, it has a hybrid engine, solar panels on the wings, etc. Of course, it is a bomber, and so the bombs themselves produce less CO2 in the explosion. There is a lot to do in order to make each bombing more efficient, and this brings us to you.

After signing a highly constrained confidentiality document, you are asked to develop the main bomb-routing algorithm. You should be happy; after all you'll be helping to save the planet!

Each bomber has extremely accurate Differential GPS receivers as well as all of the geographical maps available at the moment. It also has powerful computers, but each decision has to be made as quickly as possible. With the speed of the plane, a millisecond can make a big difference.

Task

Please, write a program to find the sequence of points on the map that the bomber is expected to travel such that it hits all *vital links*. A *link* from A to B is *vital* when its absence isolates completely A from B. In other words, the only way to go from A to B (or vice versa) is via that link. In Figure 1, we show an example of three vital links in a road map.

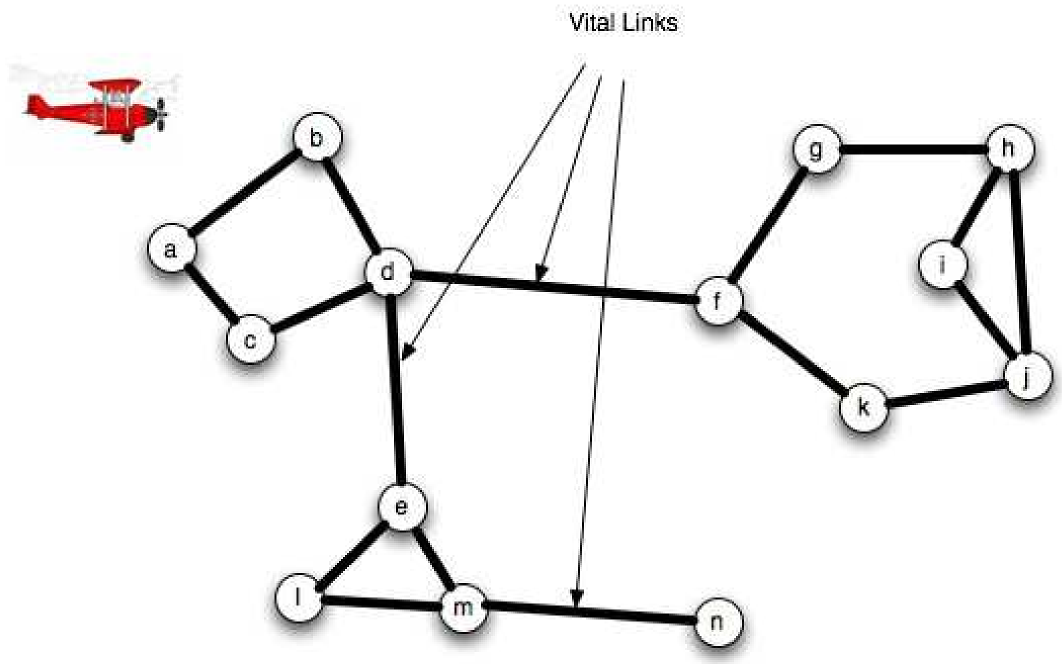


Figure 1 - The Vital Links

Notice that if we destroy for example link (d,e), it becomes impossible to go from d to e,m,l or n in any way.

A vital link can be hit at any point that lies in its segment (e.g. a hit close to d is as valid as a hit close to e). Of course, only one hit is enough to neutralize a vital link. Moreover, each bomb affects an exact circle of radius R, i.e., every segment that intersects that circle is considered hit (figure 2).

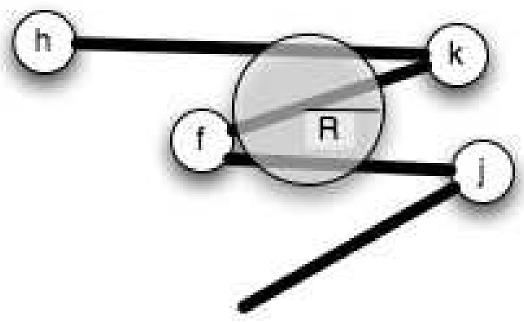


Figure 2 - Range R of a bomb – (h,k), (f,k) and (f,j) are hit

Due to enemy counter-attack, the plane may have to retreat at any moment, so the plane should follow, at each moment, to the closest vital link possible, even if the total distance grows larger in the end.

Given all coordinates (the initial position of the plane and the nodes in the map) and the range R, you have to determine the sequence of positions in which the plane has to drop bombs.

This sequence should start (takeoff) and finish (landing) at the initial position. Except for the start and finish, all the other positions have to fall exactly in a segment of the map (i.e. it should correspond to a point in a non-hit vital link segment). In figure 3, we show an example of a possible trajectory that eliminates the vital links shown in figure 1.

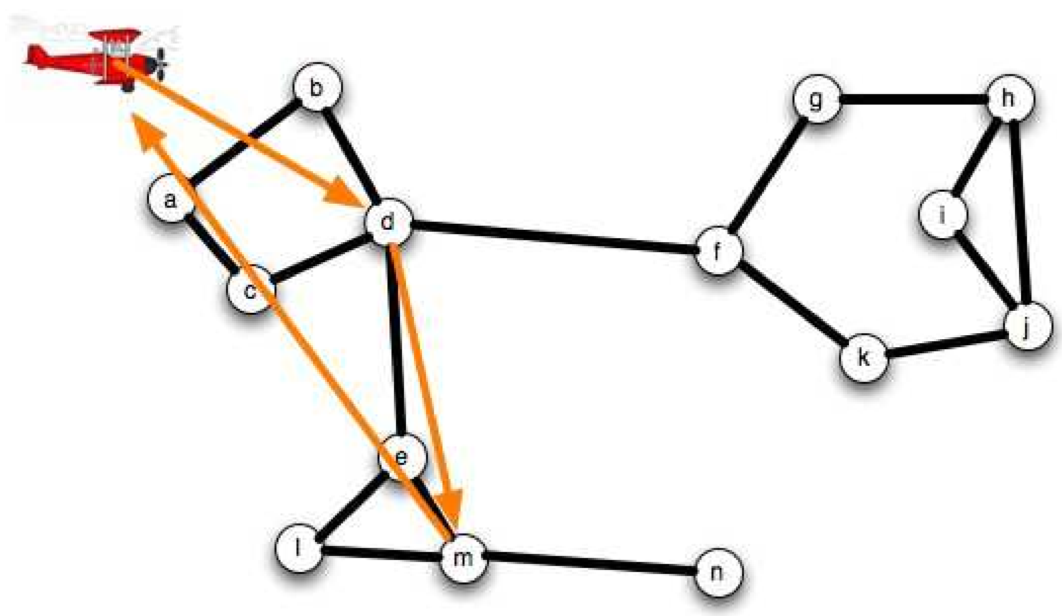


Figure 3 - Sequence of bomb drops: the plane drops a bomb in d (hitting (d,f) and (d,e)), then in m (hitting (m,n)) and returns to base.

The coordinate system used will be UTM (Universal Transverse Mercator) northing and easting, which basically corresponds to a Euclidian perspective of the world (X=Easting; Y=Northing).

Input

Each input file will start with three floating point numbers indicating the X0 and Y0 coordinates of the airport and the range R.
The second line contains an integer, N, indicating the number of nodes in the road network graph. Then, the next N (<10000) lines will each contain a pair of floating point numbers indicating the Xi and Yi coordinates (1<i<=N). Notice that the index i becomes the identifier of each node.
Finally, the last block starts with an integer M, indicating the number of links. Then the next M (<10000) lines will each have two integers, Ak and Bk (1<Ak,Bk<=N; 0<k<M) that correspond to the identifiers of the points that are linked together. No two links will ever cross each other.

Output

The program will print the sequence of coordinates (pairs of floating point numbers with exactly one decimal place), each one at a line, in the order that the plane should visit (starting and ending in the airport).

Sample Input 1

```
102.3 553.9 0.2
14
342.2 832.5
596.2 638.5
479.7 991.3
720.4 874.8
744.3 1284.1
1294.6 924.2
1467.5 659.6
1802.6 659.6
1686.2 860.7
1548.6 1111.2
1834.4 1054.8
564.4 1442.8
850.1 1460.5
1294.6 1485.1
17
1 2
1 3
2 4
3 4
4 5
4 6
6 7
7 8
8 9
8 10
9 10
10 11
6 11
5 12
5 13
12 13
13 14
```

Sample Output 1

```
102.3 553.9
720.4 874.8
850.1 1460.5
102.3 553.9
```

Sample Input 2

```
102.3 553.9 5
14
342.2 832.5
596.2 638.5
479.7 991.3
```

```
720.4 874.8
744.3 1284.1
1294.6 924.2
1467.5 659.6
1802.6 659.6
1686.2 860.7
1548.6 1111.2
1834.4 1054.8
564.4 1442.8
850.1 1460.5
1294.6 1490.1
17
1 2
1 3
2 4
3 4
4 5
4 6
6 7
7 8
8 9
8 10
9 10
10 11
6 11
5 12
5 13
12 13
12 14
```

Sample Output 2

```
102.3 553.9
720.4 874.8
683.1 1450.5
102.3 553.9
```

Problem 13 – The hard life of a mole

Moles live underground and burrow holes. When travelling underground from point A to point B, any clever mole takes advantage of existing tunnels and minimizes the amount of digging through the soil. Of course, any rock must be circumvented because it is impossible to dig through a rock. Furthermore, any spot tainted with castor oil repellent should also be avoided because moles hate castor oil so badly that they are willing to dig longer tunnels to circumvent it. Unfortunately, sometimes there is no clear path available and the mole has no other option than, at some point, dig through the castor oil - Yuck! Disgusting!

A map of the underground is available in the form of a grid of cells. The map can be traversed in the horizontal direction, vertical direction and both diagonal directions. The points A and B are marked in the map. It is assumed that the start point A is a cell of an existing tunnel and the end point B is a cell of uncontaminated soil.

Task

Please, write a program that, given a map of the underground, computes the minimum amount of digging necessary to travel from A to B, circumventing the contaminated cells as most as possible.

Input

The program gets its input from the standard input stream. The first input line contains two integers, *n* and *m*, separated by a single space:

- *n* (2 <= *n* <= 300) is the number of lines in the map;
- *m* (2 <= *m* <= 300) is the number of characters per line in the map.

The following *n* lines, each with *m* characters, contain the following valid character values:

- #   uncontaminated soil cell, a section of clean compact underground soil;
- .   open cell, a section of an existing tunnel;
- @   rock cell, a rock;
- ~   contaminated soil cell, a section of soil tainted with castor oil repellent;
- A   the point from where the Mole starts digging;
- B   the point the Mole intends to reach.

Output

The program writes its output to the standard output stream. The output is a line with a single positive integer, representing the final result.

Sample Input

```
30 70
######################################################################
#.A...################################################################
######################################################################
######################################################################
######################################################################
##................................................................####
########.#############################################################
#########.############################################################
##########.###########################################################
###########.####################@@@@@@@@@@@@@@@@@@@@@@@@@#############
############.###################@.@@@.....@@.....@@.....@#############
#############.##################@.@@@.@@@@@@.@@@@@@.@@@@@#############
##############.#################@.@@@.@@@@@@.@@@@@@.@@@@@#############
###############.################@.@@@.@@@@@@.@@@@@@.@@@@@#############
################.###############@.@@@....@@@....@@@....@@#############
#################.##############@.@@@.@@@@@@.@@@@@@.@@@@@#############
#@@@@@@@@@@@@@@@@@.#############@.@@@.@@@@@@.@@@@@@.@@@@@#############
#@@@@@@@@@@@@@@@@@@.############@.@@@.@@@@@@@.@@@@@.@@@@@#############
#@@@@@@@@@@@@@@@@@#@.###########@.@@@.....@@.....@@.....@#############
#@@@@@@@@@@@@@@@@@##@.#########@@@@@@@@@@@@@@@@@@@@@@@@@@#############
#@@@@@@@@@@@@@@@@@###@.###############################################
#@@@@@@@@@@@@@@@@@####@.##############################################
```

```
#######################@.############################################
########################@.###########################################
#######################~~~.~~~~######################################
#######################~~~~.~~~######################################
#####.........................................................~~~~###
###########################################################...~~~~~##
###########################################################...@@B~~##
###########################################################@@@@@@@@@@
```

Sample Output

10

Problem 14 – Codebreaking

The president of Damogran asks you for help with a serious situation. He has evidence that his neighboring country Qvarne is supporting smugglers operating in Damogran. The Damogranian intelligence service tracked down a woman named Alice Beeblebrox and it suspects that she belongs to the smuggler's organization. However, all attempts to monitor Alice's communications have failed so far, because Alice is using some sort of encryption to hide her messages from eavesdroppers.

Some weeks ago a crack team of spies swung into action to search for information about how the smugglers are communicating with each other and how they encrypt their messages for transmission.

They were able to take possession of an algorithm Alice seems to use to encrypt the messages she sends to other people. The encryption algorithm is encrypting data in blocks of 32 bits using a key length of 30 bits. An interesting aspect the spies noted is that Alice seems to use the same algorithm for encryption as well as for decryption. Since the smugglers are probably operating all over the world they are most likely communicating using English text messages which are represented by standard ASCII characters for compatibility.

For your convenience, we translated the algorithm the spies found into the official competition languages. Use one of the following code snippets as cipher function to encrypt or decrypt a single data block with a given key. The key needs to be located in the lower 30 bits of the key argument, the upper bits are ignored.

C/C++

```
unsigned int xCrypt(unsigned int block, unsigned int key) {
      unsigned int skey1, skey2;
      key &= 0x3FFFFFFF;
      skey1 = (key >> 16) | (key << 16);
      skey2 = (key << 2) | (key >> 28);
      return ((block + skey1) ^ skey2) - skey1;
}
```

C#

```
static public uint xCrypt(uint block, uint key) {
      uint skey1, skey2;
      key &= 0x3FFFFFFF;
      skey1 = (key >> 16) | (key << 16);
      skey2 = (key << 2) | (key >> 28);
      return ((block + skey1) ^ skey2) - skey1;
}
```

Java

```
static public long xCrypt(long block, int key) {
      long lkey = key < 0 ? ((long) (key)) + 0x80000000 : key;
      lkey &= 0x3FFFFFFF;
      long skey1 = (lkey >> 16) | (lkey << 16);
      long skey2 = (lkey << 2) | (lkey >> 28);
      long rblock = (((block + skey1) ^ skey2) & 0xFFFFFFFF) - skey1;
      return rblock < 0 ? rblock + 0x80000000 : rblock;
}
```

These functions are provided.

One of the Damogranian listening posts was recently able to intercept a message send from Alice to a person only known as the “Almighty Bob”. The encrypted message consists of 32 bit words that have been recorded in a text file containing the hexadecimal representation of an encrypted word in each line. The hexadecimal numbers are in big-endian digit order, so you can use standard conversion functions to convert them without reordering the digits.

The intercepted message you have to analyze is provided.

Task

Please, write a program to find the key used to encrypt the message as well as the decrypted text. It should be able to decrypt this message and other similar messages that will be intercepted in the future, of course. However, in those cases where it is unable to break the code and to decrypt the given message, it should output the words UNABLE TO BREAK THE CODE.

The input to your program is the name of a single text file containing the encrypted message as described above. The output should be formatted as in the examples below, where “program” represents the name of your program.

Examples

```
> codebreak encoded.txt
Key found: 0x1234ABCD

This is the decrypted message!
```

(Of course, this is not the correct key and message... ;-))

```
> codebreak shortmsg.txt
UNABLE TO BREAK THE CODE
```

Problems

Problem 15 – Coins

Alice and Bob were sitting in the sun; drinking orange juice; and watching some migrating ducks fly to Africa.

"Look", noted Alice, "one of the ducks left a trail of golden coins on the floor".

"Great!" exclaimed Bob, "let's play a game with this line of coins. We will take turns, where each one of us will flip one coin from 'head' into 'tail' state".

"Ok", agreed Alice and added, "but when we flip a coin, we can also opt to flip the coin immediately after it, even if that coin is a 'tail', in which case it becomes a 'head'".

"And whoever can not play - loses" cried both of them simultaneously.

Cunning Bob knew that he could count on witty IEEEXtreme contestants to help him win. Can you help him do that?

Task

Your task is to write a program that given a string of H/T letters, computes a winning move for the flip-coin game, if there is one, or reports that there in no winning move, if this is the case. A winning move is a legal move such that either the player wins immediately (because there are no more coins to flip), or else, after any subsequent move by the opponent there is a winning move for the player.

For example, if the input is TTTT then Bob lost the game (there is no "head" so he can not play and thus he lost). For the input TTTTHTTTT, Bob wins by flipping the fifth coin; for the input TTHHT, Bob wins by flipping both "Heads" (third and fourth coins); for the input THHTHTHT, Bob wins if he flips coins 2 and 3.

Input

The input file to be read from the console contains one line in which there is a string entirely composed of the letters H and T, representing the state of the coins, as explained.

Output

The output file, to be written at the console, contains one line, with one number. A positive number *N* means that flipping the *N*th coin is a winning move. A negative number, written –*N*, means that flipping the *N*th and the *N*+1th coins is a winning move. Zero, written 0, means that there is no winning move. Note that, in general, there can be several winning moves, for a given list of coins. Your program can output any of them.

Sample Input 1

```
TTTT
```

Sample Output 1

```
0
```

Sample Input 2

```
TTTTHTTTT
```

Sample Output 2

```
5
```

Sample Input 3

```
TTHHT
```

Sample Output 3

```
-3
```

Sample Input 4

```
THHHTHTTHT
```

Sample Output 4

```
6
```

Sample Input 5

```
THHHTTTTHT
```

Sample Output 5

```
0
```

Problem 16 – Soccer AI

Soccer is considered to be the "King of Sports" in most of the world. The game is quite simple: each team wants to send the ball into the opponent team's goal, the more times, the better. Still, there are some complicated rules, namely the offside rules and rules defining what goalkeepers can do, cannot do and what can be done to them. So, in this problem, we will simplify...

First, suppose that there are no goalkeepers. That's a huge simplification, but understandable.

The simplified rules for offside are the following: a player is considered to be offside when all the following four conditions are true:

1. The high pass was made by a co-player of the player examined for offside
2. The ball is heading towards the goalpost of the defending team

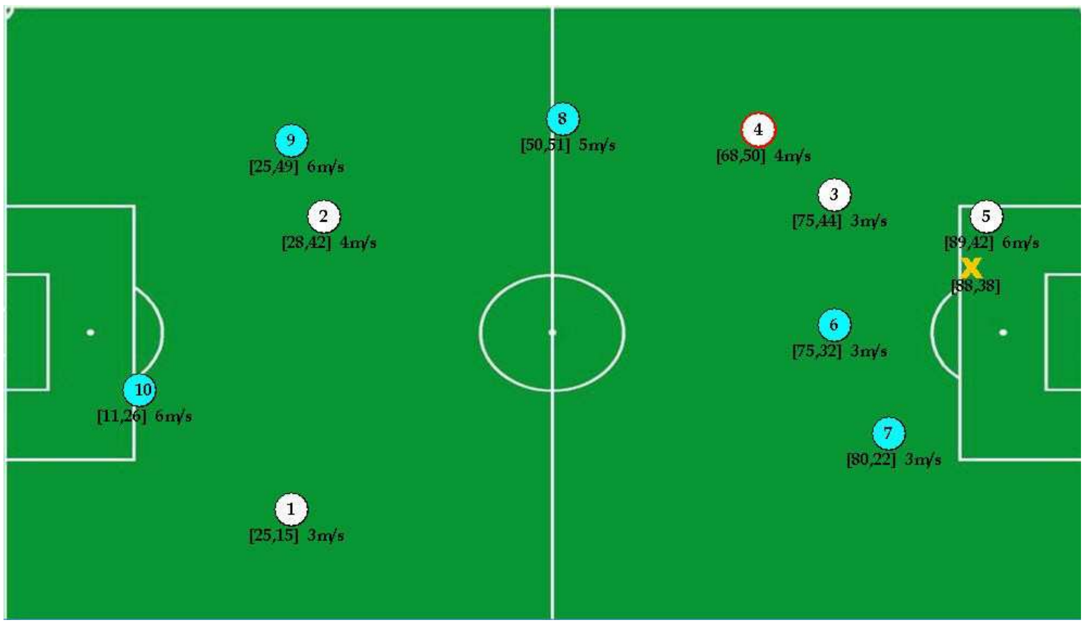
3. The player, examined for offside, lied behind all the players of the defending team at the moment that the high pass was made.
4. The player, examined for offside, is not the one who initially had the ball.

If a player is offside, he should not move towards the end position of the ball after an attacking pass is made, because that would result in cancelling the attack effort of his team.

Under these simplified rules, wouldn't it be interesting to estimate which player or players – not being offside – are in best position to get the ball, after a high pass is made?

Consider that for each high pass we know the exact location (in *x*-*y* coordinates) where the ball will finally land after the pass is made. Moreover, we suppose that all players move at a constant speed and that the first team attacks from left-to-right whereas the second team attacks from right-to-left. The attacking team is the one who initially has the ball; the other team is the defending team.

As an example, consider the following figure which corresponds to the first input in the samples below. In this figure, the players of the first team (attacking form left-to-right) are represented by white circles and the players of the second team (attacking form right-to-left) by cyan circles. Below every player, the coordinates (in [*x*, *y*] format) and the speed (in meters/sec) of the player are shown. The player initially having the ball has a thick red line around its circle and the final position of the ball is marked by an orange X.



In this figure, we observe that the player initially having the ball belongs to the first team and makes a high pass towards the goalpost of the defending team. The player with coordinates [89, 41] is closer to the end position of the ball, but he should not move towards that point because he is offside. At the moment when the high pass is made, the non-offside players closest to the end position of the ball are the ones with coordinates [75, 44] and [75, 32]. Since they move at the same speed, they reach the landing position of the ball at exactly the same time. (But note that this particular case: in general, not all speeds are the same, as the example also shows.)

Task

Please, write a program that, given a description of the playing field, with the positions and speed of all the players, the player who has the ball and the end point of the ball after the player who has the ball makes a high pass, finds the non-offside players that are closest, in terms of time, to the end position of the ball at the exact moment the high pass is made.

Input

The input file is given as an argument to the program in the command line. It is formatted as follows:

- The first line has two comma-separated positive integer numbers. The first one, A, represents the number of players of the first team, and the second one, B, represents the number of players of the second team.
- In each of the following A lines, there are three comma-separated positive integer numbers, representing the X and Y coordinates and the speed in meters/second of each of the players of the first team.
- After that, in each of the following B lines we have the data for the players of the second team.
- There is one more line, with three comma-separated positive integer numbers. The first number ranges from 1 to (A+B) and represents the player (in order of appearance in the input file) that currently has the ball and makes the high pass. The other two numbers represent the X and Y coordinates that map the location of the end position of the ball.

Output

Your program writes at the console a list of comma-separated pairs representing the coordinates of the players who can reach the end position of the ball faster, after the pass is made. There will be one pair per line, write in the order they appear in the input file.

Sample Input 1

```
5,5
25,15,3
28,42,4
75,44,3
68,50,4
89,42,6
75,32,3
80,22,3
50,51,5
25,49,6
11,26,6
4,88,38
```

Sample Output 1

```
75, 44
```

```
75, 32
```

Sample Input 2

```
3,2
94,22,5
95,44,8
89,45,5
88,50,5
85,40,6
2,94,21
```

Sample Output 2

```
94, 22
```