Problem Set for JAG Practice Contest 2011

Japanese Alumni Group

Contest Held on 6 Nov 2011

Status of Problems

All problems were newly created by the members of Japanese Alumni Group.

This problem set was typeset by LaTeX 2_ε with a style file made by a member of JAG from scratch, so that the statement looks like those used in ACM-ICPC Regional Contest held in Japan.

Terms of Use

Problem A
Infinity Maze

Input: A.txt
Time Limit: 30 seconds

Dr. Fukuoka has placed a simple robot in a two-dimensional maze. It moves within the maze and never goes out of the maze as there is no exit.

The maze is made up of $H \times W$ grid cells as depicted below. The upper side of the maze faces north. Consequently, the right, lower and left sides face east, south and west respectively. Each cell is either empty or wall and has the coordinates of (i, j) where the north-west corner has $(1, 1)$. The row i goes up toward the south and the column j toward the east.

North

West				East
	$(1,1)$	$(1,2)$	$\cdots$	$(1,W)$
	$\vdots$	$\vdots$		$\vdots$
	$(H,1)$	$(H,2)$	$\cdots$	(H,W)

South

The robot moves on empty cells and faces north, east, south or west. It goes forward when there is an empty cell in front, and rotates 90 degrees to the right when it comes in front of a wall cell or on the edge of the maze. It cannot enter the wall cells. It stops right after moving forward by L cells.

Your mission is, given the initial position and direction of the robot and the number of steps, to write a program to calculate the final position and direction of the robot.

Input

The input is a sequence of datasets. Each dataset is formatted as follows.

H W L

$c_{1,1}c_{1,2}\ldots c_{1,W}$

$\vdots$

$c_{H,1}c_{H,2}\ldots c_{H,W}$

The first line of a dataset contains three integers H, W and $L(1 \leq H, W \leq 100, 1 \leq L \leq 10^{18})$.

Each of the following H lines contains exactly W characters. In the i-th line, the j-th character $c_{i,j}$ represents a cell at (i, j) of the maze. "`.`" denotes an empty cell. "`#`" denotes a wall cell. "`N`", "`E`", "`S`", "`W`" denote a robot on an empty cell facing north, east, south and west respectively; it indicates the initial position and direction of the robot.

You can assume that there is at least one empty cell adjacent to the initial position of the robot.

The end of input is indicated by a line with three zeros. This line is not part of any dataset.

Output

For each dataset, output in a line the final row, column and direction of the robot, separated by a single space. The direction should be one of the following: "`N`" (north), "`E`" (east), "`S`" (south) and "`W`" (west).

No extra spaces or characters are allowed.

Sample Input

```
3 3 10
E..
.#.
...
5 5 19
####.
.....
.#S#.
...#.
#.##.
5 5 6
#.#..
#....
##.#.
#..S.
#....
5 4 35
..##
....
.##.
```

```
.#S.
...#
0 0 0
```

Output for the Sample Input

```
1 3 E
4 5 S
4 4 E
1 1 N
```

Problem B
Butterfly

Input: B.txt
Time Limit: 30 seconds

Claire is a man-eater. She's a real man-eater. She's going around with dozens of guys. She's dating all the time. And one day she found some conflicts in her date schedule. D'oh!

So she needs to pick some dates and give the others up. The dates are set by hours like 13:00 to 15:00. She may have more than one date with a guy. For example, she can have dates with Adam from 10:00 to 12:00 and from 14:00 to 16:00 and with Bob from 12:00 to 13:00 and from 18:00 to 20:00. She can have these dates as long as there is no overlap of time. Time of traveling, time of make-up, trouble from love triangles, and the likes are not of her concern. Thus she can keep all the dates with Adam and Bob in the previous example. All dates are set between 6:00 and 22:00 on the same day.

She wants to get the maximum amount of *satisfaction* in total. Each guy gives her some satisfaction if he has *all* scheduled dates. Let's say, for example, Adam's satisfaction is 100 and Bob's satisfaction is 200. Then, since she can make it with both guys, she can get 300 in total.

Your task is to write a program to satisfy her demand. Then she could spend a few hours with you... if you really want.

Input

The input consists of a sequence of datasets. Each dataset has the following format:

$$
\begin{array}{l}
N \\
Guy_1 \\
\quad \vdots \\
Guy_N
\end{array}
$$

The first line of the input contains an integer N ($1 \leq N \leq 100$), the number of guys. Then there come the descriptions of guys. Each description is given in this format:

$$
\begin{array}{l}
M\ L \\
S_1\ E_1 \\
\quad \vdots \\
S_M\ E_M
\end{array}
$$

The first line contains two integers M_i ($1 \leq M_i \leq 16$) and L_i ($1 \leq L_i \leq 100{,}000{,}000$), the number of dates set for the guy and the satisfaction she would get from him respectively. Then M lines follow. The i-th line contains two integers S_i and E_i ($6 \leq S_i < E_i \leq 22$), the starting and ending time of the i-th date.

The end of input is indicated by $N = 0$.

Output

For each dataset, output in a line the maximum amount of satisfaction she can get.

Sample Input

```
2
2 100
10 12
14 16
2 200
12 13
18 20
4
1 100
6 22
1 1000
6 22
1 10000
6 22
1 100000
6 22
16
1 100000000
6 7
1 100000000
7 8
1 100000000
8 9
1 100000000
9 10
1 100000000
10 11
1 100000000
11 12
1 100000000
12 13
1 100000000
```

```
13 14
1 100000000
14 15
1 100000000
15 16
1 100000000
16 17
1 100000000
17 18
1 100000000
18 19
1 100000000
19 20
1 100000000
20 21
1 100000000
21 22
0
```

Output for the Sample Input

```
300
100000
1600000000
```

Problem C
Chinese Classics

Input: C.txt
Time Limit: 30 seconds

Taro, a junior high school student, is working on his homework. Today's homework is to read Chinese classic texts.

As you know, Japanese language shares the (mostly) same Chinese characters but the order of words is a bit different. Therefore the notation called "returning marks" was invented in order to read Chinese classic texts in the order similar to Japanese language.

There are two major types of returning marks: 'Re' mark and jump marks. Also there are a couple of jump marks such as one-two-three marks, top-middle-bottom marks. The marks are attached to letters to describe the reading order of each letter in the Chinese classic text. Figure 1 is an example of a Chinese classic text annotated with returning marks, which are the small letters at the bottom-left of the big Chinese letters.

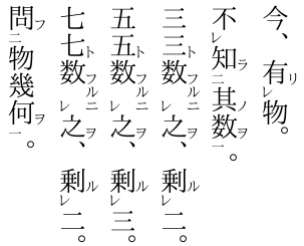

Figure 1: a Chinese classic text

Taro generalized the concept of jump marks, and summarized the rules to read Chinese classic texts with returning marks as below. Your task is to help Taro by writing a program that interprets Chinese classic texts with returning marks following his rules, and outputs the order of reading of each letter.

When two (or more) rules are applicable in each step, the latter in the list below is applied first, then the former.

1. Basically letters are read downwards from top to bottom, i.e. the first letter should be read (or skipped) first, and after the i-th letter is read or skipped, $(i + 1)$-th letter is read next.
2. Each jump mark has a *type* (represented with a string consisting of lower-case letters) and

a *number* (represented with a positive integer). A letter with a jump mark whose number is 2 or larger must be skipped.

3. When the i-th letter with a jump mark of type t, number n is read, and when there exists an unread letter L at position less than i that has a jump mark of type t, number $n+1$, then L must be read next. If there is no such letter L, the $(k+1)$-th letter is read, where k is the index of the most recently read letter with a jump mark of type t, number 1.
4. A letter with a 'Re' mark must be skipped.
5. When the i-th letter is read and $(i-1)$-th letter has a 'Re' mark, then $(i-1)$-th letter must be read next.
6. No letter may be read twice or more. Once a letter is read, the letter must be skipped in the subsequent steps.
7. If no letter can be read next, finish reading.

Let's see the first case of the sample input. We begin reading with the first letter because of the rule 1. However, since the first letter has a jump mark '`onetwo2`', we must follow the rule 2 and skip the letter. Therefore the second letter, which has no returning mark, will be read first.

Then the third letter will be read. The third letter has a jump mark '`onetwo1`', so we must follow rule 3 and read a letter with a jump mark '`onetwo2`' next, if exists. The first letter has the exact jump mark, so it will be read third. Similarly, the fifth letter is read fourth, and then the sixth letter is read.

Although we have two letters which have the same jump mark '`onetwo2`', we must not take into account the first letter, which has already been read, and must read the fourth letter. Now we have read all six letters and no letter can be read next, so we finish reading. We have read the second, third, first, fifth, sixth, and fourth letter in this order, so the output is `2 3 1 5 6 4`.

Input

The input contains multiple datasets. Each dataset is given in the following format:

$$
\begin{aligned}
&N\\
&mark_1\\
&\quad\vdots\\
&mark_N
\end{aligned}
$$

N, a positive integer ($1 \leq N \leq 10{,}000$), means the number of letters in a Chinese classic text. $mark_i$ denotes returning marks attached to the i-th letter.

A 'Re' mark is represented by a single letter, namely, '`v`' (quotes for clarity). The description of a jump mark is the simple concatenation of its type, specified by one or more lowercase letter,

and a positive integer. Note that each letter has at most one jump mark and at most one 'Re' mark. When the same letter has both types of returning marks, the description of the jump mark comes first, followed by 'v' for the 'Re' mark. You can assume this happens only on the jump marks with the number 1.

If the i-th letter has no returning mark, $mark_i$ is '-' (quotes for clarity). The length of $mark_i$ never exceeds 20.

You may assume that input is well-formed, that is, there is exactly one reading order that follows the rules above. And in the ordering, every letter is read exactly once.

You may also assume that the N-th letter does not have 'Re' mark.

The input ends when $N = 0$. Your program must not output anything for this case.

Output

For each dataset, you should output N lines. The first line should contain the index of the letter which is to be read first, the second line for the letter which is to be read second, and so on. All the indices are 1-based.

Sample Input

```
6
onetwo2
-
onetwo1
onetwo2
-
onetwo1
7
v
topbottom2
onetwo2
-
onetwo1
topbottom1
-
6
baz2
foo2
baz1v
bar2
foo1
bar1
0
```

Output for the Sample Input

```
2
3
1
5
6
4
4
5
3
6
2
1
7
5
2
6
4
3
1
```

Problem D
Revenge of Champernowne Constant

Input: D.txt
Time Limit: 30 seconds

Champernowne constant is an irrational number. Its decimal representation starts with "0.", followed by concatenation of all positive integers in the increasing order.

You will be given a sequence S which consists of decimal digits. Your task is to write a program which computes the position of the first occurrence of S in Champernowne constant after the decimal point.

Input

The input has multiple test cases. Each line of the input has one digit sequence. The input is terminated by a line consisting only of `#`.

It is guaranteed that each sequence has at least one digit and its length is less than or equal to 100.

Output

For each sequence, output one decimal integer described above. You can assume each output value is less than 10^{16}.

Sample Input

```
45678
67891011
21
314159265358979
#
```

Output for the Sample Input

```
4
6
15
2012778692735799
```

Problem E
Full Text Search

Input: E.txt
Time Limit: 30 seconds

Mr. Don is an administrator of a famous quiz website named QMACloneClone. The users there can submit their own questions to the system as well as search for question texts with arbitrary queries. This search system employs bi-gram search method.

The bi-gram search method introduces two phases, namely preprocessing and search:

Preprocessing Precompute the set of all the substrings of one or two characters long for each question text.

Search Compute the set for the query string in the same way. Then find the question texts whose precomputed sets completely contain the set constructed from the query.

Everything looked fine for a while after the feature was released. However, one of the users found an issue: the search results occasionally contained questions that did not include the query string as-is. Those questions are not likely what the users want. So Mr. Don has started to dig into the issue and asked you for help. For each given search query, your task is to find the length of the shortest question text picked up by the bi-gram method but not containing the query text as its substring.

Input

The input consists of multiple datasets. A dataset is given as a search query on each line. The input ends with a line containing only a hash sign ("`#`"), which should not be processed.

A search query consists of no more than 1,000 and non-empty lowercase and/or uppercase letters. The question texts and queries are case-sensitive.

Output

For each search query, print the minimum possible length of a question text causing the issue. If there is no such question text, print "`No Result`" in one line (quotes only to clarify).

Sample Input

```
a
```

```
QMAClone
acmicpc
abcdefgha
abcdefgdhbi
abcbcd
#
```

Output for the Sample Input

```
No Results
9
7
9
12
6
```

Note

Let's consider the situation that one question text is "CloneQMAC". In this situation, the set computed in the preprocessing phase is {"C", "Cl", "l", "lo", "o", "on", "n", "ne", "e", "eQ", "Q", "QM", "M", "MA", "A", "AC"}.

In the testcase 2, our input text (search query) is "QMAClone". Thus the set computed by the program in the search phase is {"Q", "QM", "M", "MA", "A", "AC", "C", "Cl", "l", "lo", "o", "on", "n", "ne", "e"}.

Since the first set contains all the elements in the second set, the question text "CloneQMAC" is picked up by the program when the search query is "QMAClone" although the text "CloneQMAC" itself does not contain the question text "QMAClone". In addition, we can prove that there's no such text of the length less than 9, thus, the expected output for this search query is 9.

Problem F
Mysterious Maze

Input: F.txt
Time Limit: 30 seconds

A robot in a two-dimensional maze again. The maze has an entrance and an exit this time, though.

Just as in the previous problem, the maze is made up of $H \times W$ grid cells, its upper side faces north, and each cell is either empty or wall. Unlike the previous, on the other hand, one of the empty cells is connected to the entrance and another to the exit.

The robot is rather complex — there is some control, but not full. It is associated with a controller that has two buttons, namely *forward* and *turn*. The forward button moves the robot forward to the next cell, if possible. The robot can not move into a wall nor outside the maze. The turn button turns the robot as *programmed*. Here the program is a finite sequence of N commands, each of which is either 'L' (indicating a left turn) or 'R' (a right turn). The first turn follows the first command; the second turn follows the second command; similar for the following turns. The turn button stops working once the commands are exhausted; the forward button still works in such a case though. The robot always turns by 90 degrees at once.

The robot is initially put on the entrance cell, directed to the north. Your mission is to determine whether it can reach the exit cell if controlled properly.

Input

The input is a sequence of datasets. Each dataset is formatted as follows.

$$
\begin{array}{l}
H\ W\ N \\
s_1 \ldots s_N \\
c_{1,1} c_{1,2} \ldots c_{1,W} \\
\quad \vdots \\
c_{H,1} c_{H,2} \ldots c_{H,W}
\end{array}
$$

The first line of a dataset contains three integers H, W and N ($1 \leq H, W \leq 1{,}000$, $1 \leq N \leq 1{,}000{,}000$).

The second line contains a program of N commands.

Each of the following H lines contains exactly W characters. Each of these characters represents a cell of the maze. "`.`" indicates empty, "`#`" indicates a wall, "`S`" indicates an entrance, and

“G” indicates an exit. There is exactly one entrance cell and one exit cell.

The end of input is indicated by a line with three zeros.

Output

For each dataset, output whether the robot can reach the exit in a line: “Yes” if it can or “No” otherwise (without quotes).

Sample Input

```
2 2 1
L
G.
#S
2 2 2
RR
G.
.S
3 3 6
LLLLLL
G#.
...
.#S
0 0 0
```

Output for the Sample Input

```
Yes
No
Yes
```

Problem G
Number Sorting

Input: G.txt
Time Limit: 30 seconds

Consider sets of natural numbers. Some sets can be sorted in the same order numerically and lexicographically. $\{2, 27, 3125, 9000\}$ is one example of such sets; $\{2, 27, 243\}$ is not since lexicographic sorting would yield $\{2, 243, 27\}$.

Your task is to write a program that, for the set of integers in a given range $[A, B]$ (i.e. between A and B inclusive), counts the number of non-empty subsets satisfying the above property. Since the resulting number is expected to be very huge, your program should output the number in modulo P given as the input.

Input

The input consists of multiple datasets. Each dataset consists of a line with three integers A, B, and P separated by a space. These numbers satisfy the following conditions: $1 \leq A \leq$ 1,000,000,000, $0 \leq B - A <$ 100,000, $1 \leq P \leq$ 1,000,000,000.

The end of input is indicated by a line with three zeros.

Output

For each dataset, output the number of the subsets in modulo P.

Sample Input

```
1 10 1000
1 100000 1000000000
999999999 1000099998 1000000000
0 0 0
```

Output for the Sample Input

```
513
899507743
941554688
```

Problem H
Sky Jump

Input: H.txt
Time Limit: 30 seconds

Dr. Kay Em, a genius scientist, developed a new missile named "Ikan-no-i." This missile has N jet engines. When the i-th engine is ignited, the missile's velocity changes to (vx_i, vy_i) immediately.

Your task is to determine whether the missile can reach the given target point (X, Y). The missile can be considered as a mass point in a two-dimensional plane with the y-axis pointing up, affected by the gravity of 9.8 downward (i.e. the negative y-direction) and initially set at the origin $(0, 0)$. The engines can be ignited at any time in any order. Still, note that at least one of them needs to be ignited to get the missile launched.

Input

The input consists of multiple datasets. Each dataset has the following format.

N
vx_1 vy_1
vx_2 vy_2
$\vdots$
vx_N vy_N
X Y

All values are integers and meet the following constraints: $1 \leq N \leq 1{,}000$, $0 < vx_i \leq 1{,}000$, $-1{,}000 \leq vy_i \leq 1{,}000$, $0 < X \leq 1{,}000$, $-1{,}000 \leq Y \leq 1{,}000$.

The end of input is indicated by a line with a single zero.

Output

For each dataset, output whether the missile can reach the target point in a line: "`Yes`" if it can, "`No`" otherwise.

Sample Input

```
1
```

```
1 1
10 -480
2
6 7
5 5
4 1
3
10 5
4 4
8 4
2 0
0
```

Output for the Sample Input

```
Yes
Yes
No
```

Problem I
Mobile Network

Input: I.txt
Time Limit: 30 seconds

The traffic on the Internet is increasing these days due to smartphones. The wireless carriers have to enhance their network infrastructure.

The network of a wireless carrier consists of a number of base stations and lines. Each line connects two base stations bi-directionally. The bandwidth of a line increases every year and is given by a polynomial $f(x)$ of the year x.

Your task is, given the network structure, to write a program to calculate the maximal bandwidth between the 1-st and N-th base stations as a polynomial of x.

Input

The input consists of multiple datasets. Each dataset has the following format:

$$
\begin{aligned}
&N\ M\\
&u_1\ v_1\ p_1\\
&\ \ \vdots\\
&u_M\ v_M\ p_M
\end{aligned}
$$

The first line of each dataset contains two integers N ($2 \leq N \leq 50$) and M ($0 \leq M \leq 500$), which indicates the number of base stations and lines respectively. The following M lines describe the network structure. The i-th of them corresponds to the i-th network line and contains two integers u_i and v_i and a polynomial p_i. u_i and v_i indicate the indices of base stations ($1 \leq u_i, v_i \leq N$); p_i indicates the network bandwidth.

Each polynomial has the form of:

$$a_L x^L + a_{L-1} x^{L-1} + ... + a_2 x^2 + a_1 x + a_0$$

where L ($0 \leq L \leq 50$) is the degree and a_i's ($0 \leq i \leq L$, $0 \leq a_i \leq 100$) are the coefficients. In the input,

- each term $a_i x^i$ (for $i \geq 2$) is represented as $\langle a_i \rangle$`x^`$\langle i \rangle$;

- the linear term ($a_1 x$) is represented as $\langle a_1 \rangle$`x`;
- the constant (a_0) is represented just by digits;
- these terms are given in the strictly decreasing order of the degrees and connected by a plus sign (“`+`”);
- just like the standard notations, the $\langle a_i \rangle$ is omitted if $a_i = 1$ for non-constant terms;
- similarly, the entire term is omitted if $a_i = 0$ for any terms; and
- the polynomial representations contain no space or characters other than digits, “`x`”, “`^`”, and “`+`”.

For example, $2x^2 + 3x + 5$ is represented as `2x^2+3x+5`; $2x^3 + x$ is represented as `2x^3+x`, not `2x^3+0x^2+1x+0` or the like. No polynomial is a constant zero, i.e. the one with all the coefficients being zero.

The end of input is indicated by a line with two zeros. This line is not part of any dataset.

Output

For each dataset, print the maximal bandwidth as a polynomial of x. The polynomial should be represented in the same way as the input format except that a constant zero is possible and should be represented by “`0`” (without quotes).

Sample Input

```
3 3
1 2 x+2
2 3 2x+1
3 1 x+1
2 0
3 2
1 2 x
2 3 2
4 3
1 2 x^3+2x^2+3x+4
2 3 x^2+2x+3
3 4 x+2
0 0
```

Output for the Sample Input

```
2x+3
0
```

```
2
x+2
```

Problem J
Blue Forest

Input: J.txt
Time Limit: 30 seconds

John is playing a famous console game named 'Tales of Algorithmers.' Now he is facing the last dungeon called 'Blue Forest.' To find out the fastest path to run through the very complicated dungeon, he tried to draw up the dungeon map.

The dungeon consists of several floors. Each floor can be described as a connected simple plane graph. Vertices of the graph are identified by X-Y coordinate, and the length of an edge is calculated by Euclidean distance. A vertex may be equipped with a one-way warp gate. If John chooses to use the gate, it brings John to another vertex in a possibly different floor. The distance between a warp gate and its destination is considered as 0.

One vertex has at most one warp gate, though some vertices might be the destination of multiple warp gates.

He believed he made one map for each floor, however after drawing maps of all the floors, he noticed that he might have made a few mistakes. He might have drawn the same floor several times, and might have forgotten to mark some warp gates in the maps. However, he was sure he marked all warp gates at least once. So if the maps of same floor are unified to one map, all the warp gates must be described there. Luckily there are no floors which have the same shape as the other floors, so if two (or more) maps can be unified, they must be the maps of the same floor. Also, there is no floor which is circular symmetric (e.g. a regular triangle and a square).

Map A and map B can be unified if map B can be transformed to map A using only rotation and parallel translation. Since some of the warp gates on maps might be missing, you should not consider the existence of warp gates when checking unification. If it is possible to unify map A and map B, a vertex on map A and the corresponding vertex on map B are considered as 'identical' vertices. In other words, you should treat warp gates on map B as those on map A where the warp gates are located on the corresponding vertices on map A. Destinations of warp gates should be treated similarly. After that you can forget map B. It is guaranteed that if both map A and map B have warp gates which are on the identical vertices, the destinations of them are also identical.

Remember, your task is to find the shortest path from the entrance to the exit of the dungeon, using the unified maps.

Input

The input consists of multiple datasets. Each dataset is in the following format.

n
$component_1$
$component_2$
$\vdots$
$component_n$
$sl\ sn$
$dl\ dn$

n is a positive integer indicating the number of maps. $component_i$ describes the i-th map in the following format.

A
$x_1\ y_1$
$x_2\ y_2$
$\vdots$
$x_A\ y_A$
B
$s_1\ d_1$
$s_2\ d_2$
$\vdots$
$s_B\ d_B$
C
$sn_1\ dl_1\ dn_1$
$sn_2\ dl_2\ dn_2$
$\vdots$
$sn_C\ dl_C\ dn_C$

A denotes the number of vertices in the map. Each of the following A lines contains two integers x_i and y_i representing the coordinates of the i-th vertex in the 2-dimensional plane. B denotes the number of the edges connecting the vertices in the map. Each of the following B lines contains two integers representing the start and the end vertex of each edge. Vertices on the same map are numbered from 1.

C denotes the number of warp gates. Each of the following C lines has three integers describing a warp gate. The first integer is the vertex where the warp gate is located. The second and the third integer are the indices of the map and the vertex representing the destination of the warp gate, respectively. Similarly to vertices, maps are also numbered from 1.

After the description of all maps, two lines follow. The first line contains two integers sl and dl, meaning that the entrance of the dungeon is located in the sl-th map, at the vertex dl. The last line has two integers sn and dn, similarly describing the location of the exit.

The values in each dataset satisfy the following conditions:

- $1 \leq n \leq 50$,
- $3 \leq A \leq 20$,
- $A - 1 \leq B \leq A(A-1)/2$,
- $0 \leq C \leq A$, and
- $-10{,}000 \leq x_i, y_i \leq 10{,}000$.

Output

For each dataset, print the distance of the shortest path from the entrance to the exit. The output should not contain an absolute error greater than 10^{-1}. If there is no route, print -1.

Sample Input

```
2
5
0 0
10 0
20 0
30 0
30 10
4
1 2
2 3
3 4
4 5
2
1 2 4
3 2 2
5
-10 0
0 0
0 -10
0 -20
0 -30
4
1 2
2 3
3 4
4 5
```

```
1
4 1 3
1 1
2 1
4
3
4 3
0 0
5 0
2
1 2
2 3
0
3
0 0
3 4
0 5
2
1 2
1 3
1
2 3 4
4
0 13
0 0
13 0
13 13
4
1 2
1 4
2 3
2 4
0
4
5 12
0 0
-7 17
-12 5
4
1 2
2 3
2 4
4 3
0
1 1
```

```
4 1
4
3
0 0
2 0
0 4
2
1 2
1 3
0
3
0 0
-2 0
0 4
2
1 2
1 3
1
1 4 1
3
0 0
1 0
0 2
2
1 2
1 3
1
1 4 1
3
0 0
2 0
0 4
2
1 2
2 3
0
1 1
4 1
0
```

Output for the Sample Input

```
10.0000000000
41.3847763109
-1.000000000
```