Problems Index Sat Oct 07 14:38:19 EDT 2017

BOSPRE 2017 PROBLEMS
------ ---- --------

The problems are in approximate order of difficulty, easiest first.

For the first 3 problems ONLY, the autojudge will return the input and output of the judge's first failed test case, on an incorrect submission.

PYTHON is fast enough to do the first 6 problems if you program with moderate care.

problems/gmail
Is the name the same?

problems/friends
To be close or to be far.

problems/clockskew
Synchronize your circuits folks!

problems/spacetravel
See you soon!

problems/senplosion
Packing the Sensites in.

problems/trimare
Slithering around those tri-thingys.

problems/separation
Exactly where are the breaking points.

problems/energy
Coupons are not what they used to be.

problems/tourbets
A betting man's hidden helper.

problems/simplepoly
Beware the intersects!

GMail Addresses
----- ---------

When gmail.com receives a piece of mail, it ignores letter case. Thus

tweddledee@gmail.com
and
TweddleDee@GMail.Com
and
TWEDDLEDEE@gmail.COM

are all the same gmail address.

Similarly periods before the @ are ignored. Thus

tweddledee@gmail.com
and
tweddle.dee@gmail.com
and
t.w.e.d.d.l.e.d.e.e@gmail.com

are all the same gmail address.

Lastly any + before the @, and ALL CHARACTERS AFTER THE + and before the @ are ignored. Thus

tweddledee@gmail.com
and
tweddledee+tweedledum@gmail.com
and
tweddledee+phone-account@gmail.com

are all the same gmail address.

You have been asked to determine which gmail addresses are the same as a given gmail address.

Input

For each of several test cases, one line of the form

+ gmail-address

followed by any number of lines of the form

- gmail-address

Here every ‘gmail-address’ is a legal gmail address, as opposed to, for example, ‘bob&joe@gmail.com’, which is not legal because ‘&’ is not legal in a google account name.

Input ends with an end of file.

Output

A copy of the input in which some lines have been removed. The lines that REMAIN are:

(1) any line beginning with +

and

(2) any line beginning with - that contains a gmail address that is the SAME AS the closest preceding gmail address on a line beginning with ‘+’.

```
Sample Input
------ -----

+ tweedle.dee@gmail.com
- tweedle_dee@gmail.com
- TweedleDee@GMail.Com
- TweedleDee-attention-Bill@gmail.com
- TweedleDee+attention-Bill@gmail.com
- T.weed.le.Dee@GMAIL.COM
+ themellon+is.ripe@gmail.com
- the_mellon@gmail.com
- the.mellon@gmail.com
- thenellom@gmail.com
- the.MELLON@gmail.COM
- the.mellon.is.ripe@gmail.com


Sample Output
------ ------

+ tweedle.dee@gmail.com
- TweedleDee@GMail.Com
- TweedleDee+attention-Bill@gmail.com
- T.weed.le.Dee@GMAIL.COM
+ themellon+is.ripe@gmail.com
- the.mellon@gmail.com
- the.MELLON@gmail.COM


File:      gmail.txt
Author:    Bob Walton <walton@seas.harvard.edu>
Date:      Sat Oct  7 14:00:12 EDT 2017

The authors have placed this file in the public domain;
they make no warranty and accept no liability for this
file.
```

Friends in Space

The space colony Bad Light consists of a large cluster of asteroids what orbits the sun Dim Wit. The asteroids would drift apart, but are held together by wires. Conveniently the wires are not required to withstand much force, and are also useful for transporting oneself between asteroids. The system of Dim Wit is inhabited by the Zkpt, somewhat intelligent creatures we will not describe here.

Two pairs of friends are moving to Bad Light and need to find apartments on the asteroids. Zkpts never share an apartment and friends never live on the same asteroid. One pair of friends is Ba and Da who want to live on asteroids as close together as possible. Another pair of friends Gz and Tz want to live as far apart as possible (don't ask).

As a real estate broker you need to help both pairs of friends. You have a list of asteroids with available apartments and their coordinates relative to a reference point.

Input

For each test case, first a line that gives the test case name. Then a line of the form

```
N
```

giving the number of asteroids N. This is followed by N asteroid location lines each of the form

```
X Y Z
```

giving the coordinates (X,Y,Z) of one asteroid, relative to the center of the Bad Light cluster. All coordinates are integers.

```
-1000 <= X,Y,Z <= 1,000
3 <= N <= 100
```

For identification purposes the asteroids are numbered 1, ..., N in the order that their location lines are given.

Input terminates with an end of file. The test case name line is at most 80 characters.

Output

For each test case, first an exact copy of the test case name line. Then a second line of the form

```
MIN a1 a2 d
```

where a1 and a2 are the identifiers of the two closest asteroids and d is the distance between them. Lastly a third line of the the form

```
MAX a1 a2 d
```

where a1 and a2 are the identifiers of the two asteroids that are farthest apart and d is the distance between them.

To make autojudging easier THE JUDGE REQUIRES THAT a1 < a2 in both the MIN and MAX lines.

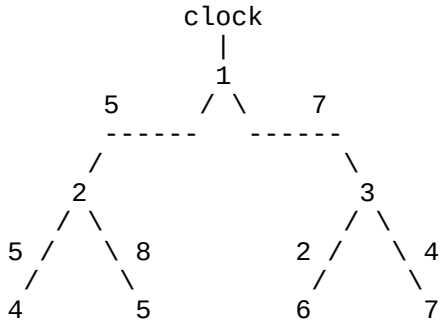
Distances must be accurate to one part in 10**5 (use default floating point format which prints 6 digits of precision).

```
Sample Input
------ -----

-- SAMPLE 1 --
4
 0 0 0
15 0 0
25 0 0
30 0 0
-- SAMPLE 2 --
6
  0  0  0
-10 10 10
 30 10 20
 20 10 30
 30 30 10
 10 30 10


Sample Output
------ ------

-- SAMPLE 1 --
MIN 3 4 5
MAX 1 4 30
-- SAMPLE 2 --
MIN 3 4 14.1421
MAX 2 5 44.7214


File:      friends.txt
Author:    Bob Walton <walton@seas.harvard.edu>
Date:      Sun Oct  1 02:20:39 EDT 2017

The authors have placed this file in the public domain;
they make no warranty and accept no liability for this
file.
```

Clock Skew
----- ----

Integrated circuits chips often use a clock that must be distributed to various places where it is used, and must be synchronous at these usage sites. Distribution is by a binary tree of gates and by wires. If the wires are of different lengths, the clock will not arrive synchronously at all its usage sites.

For example, consider the distribution network:



which uses a binary tree with 7 nodes numbered 1, 2, ..., 7 in which the wire lengths are:

length node 1 to 2 = 5 length node 1 to 3 = 7
length node 2 to 4 = 5 length node 2 to 5 = 8
length node 3 to 6 = 2 length node 3 to 7 = 4

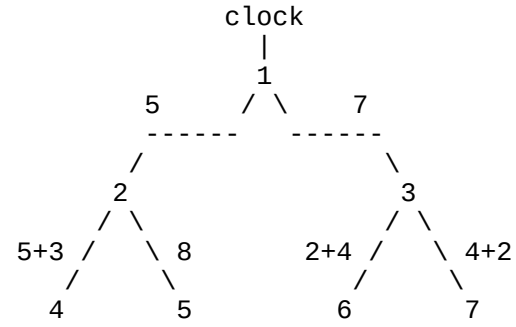
The lengths of the paths from the root node 1 to the leaves 4, 5, 6, and 7 are NOT equal:

length node 1 to 4 = 5 + 5 = 10
length node 1 to 5 = 5 + 8 = 13
length node 1 to 6 = 7 + 2 = 9
length node 1 to 7 = 7 + 4 = 11

To make the clock arrive synchronously at all leaves, we must make the lengths from the root to the leaves, equal and it is necessary to lengthen some of the wires. But this takes valuable space on the integrated circuit chip, so a goal is to keep the total length of all the wires minimal.

Consider the following two ways to make the lengths from the root in our example to each leaf equal to 13:

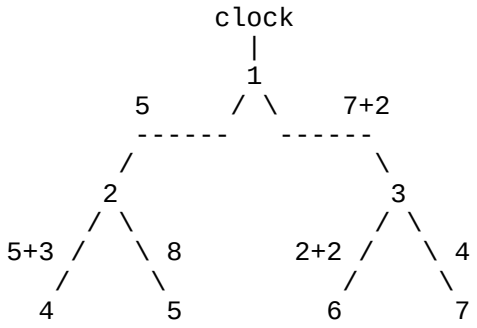
Method 1:



length node 1 to 2 = 5 length node 1 to 3 = 7
length node 2 to 4 = 5+3 length node 2 to 5 = 8
length node 3 to 6 = 2+4 length node 3 to 7 = 4+2

total length = 5 + 7 + (5+3) + 8 + (2+4) + (4+2) = 40

Method 2:



length node 1 to 2 = 5 length node 1 to 3 = 7+2
length node 2 to 4 = 5+3 length node 2 to 5 = 8
length node 3 to 6 = 2+2 length node 3 to 7 = 4

total length = 5 + (7+2) + (5+3) + 8 + (2+2) + 4 = 38

The second method is better as the total length is less.

You are being asked to figure out how much length to add to various wires of a binary tree clock distribution network in order to make the distances from the root to the leaves all equal while minimizing the total length of all the wires.

Notation

We number the nodes of a binary tree 1, 2, 3, ..., N from top to bottom and left to right. Then the children of a non-leaf node n are 2n and 2n+1 and the parent of a non-root node n is n/2 (where we use integer division that discards the remainder; e.g., 3/2 = 1). N must be one less than a power of 2.

We represent a binary tree with wire lengths by taking the form:

number of nodes = 7
length node 1 to 2 = 5 length node 1 to 3 = 7
length node 2 to 4 = 5 length node 2 to 5 = 8
length node 3 to 6 = 2 length node 3 to 7 = 4

and omitting everything but the numbers after the equals signs, so we get:

```
7
5 7
5 8
2 4
```

We will call this a ‘tree representative’.

Input

For each test case, first a line that gives the test case name, and then lines forming a tree representative.

N = 3, 7, 15, 31, 63, 127, 255, 511, or 1023
all wire lengths are between 1 and 10 inclusive

Input terminates with an end of file. The test case name line is at most 80 characters.

```
Output
------

For each test case, first an exact copy of the test case
name line, and then the tree representative of the
binary tree after you have lengthened the wires so that

  (1) The lengths from the root to each leaf are equal.
  (2) The total of all wire lengths is minimal.

Note that after lengthening some wire lengths may be
longer than 10.


Sample Input
------ -----

-- SAMPLE 1 --
7
5 7
5 8
2 4
-- SAMPLE 2 --
15
3 9
2 3
6 5
4 5
3 1
6 9
3 6
```

```
Sample Output
------ ------

-- SAMPLE 1 --
7
5 9
8 8
4 4
-- SAMPLE 2 --
15
17 9
2 4
6 9
5 5
3 3
9 9
6 6


File:      clockskew.txt
Author:    Bob Walton <walton@seas.harvard.edu>
Date:      Sat Oct  7 14:02:20 EDT 2017

The authors have placed this file in the public domain;
they make no warranty and accept no liability for this
file.
```

Space Travel

Travel planning in the Slow Galaxy requires a slow metabolism and a mathematical mind. A traveler takes ships going from one star system to another, but these take many years to make the journey, and it helps that the inhabitants of this galaxy have a metabolism a million times slower than ours, and a lifetime a million times longer.

Still, it pays to make trips optimally. Ships have schedules that say when they visit each star system on their itinerary. You want to get from one star system to another by taking a first ship for a while, then getting off and waiting for another ship to come and take you further, and so forth, but you want to get to your ultimate destination as soon as you can.

Input

For each test case, first a line that gives the test case name. Then a line of the form

```
N M
```

giving the number of start systems N and number of ships M. Then M lines, one per ship, each of the form

```
K S1 T1 S2 T1 ... SK TK
```

where K is the number of stops the ship makes, and Si, Ti describe the i+1'st stop which is at star system Si and time Ti. 0 <= T1 < T2 < T3 < ... < TK.

The star systems are numbered 1, 2, ..., N. You start at system 1 and your ultimate destination is system N. Times are integers in years. If you get to a star system at time T, you can transfer to any ship arriving at the star system at time T or later.

```
2 <= M <= 10,000
2 <= K <= 100
2 <= N <= 10,000
1 <= Si <= N
0 <= Ti <= 100,000,000
```

Input terminates with an end of file. The test case name line is at most 80 characters.

Output

For each test case, first an exact copy of the test case name line. Then a line containing just the earliest possible time of your arrival at system N assuming that you start at system 1 at time 0.

```
Sample Input
------ -----

-- SAMPLE 1 --
6 4
3 1 0 2 10 3 20
4 2 0 4 10 2 20 4 30
4 3 0 5 10 3 20 5 30
6 4 0 5 10 6 20 4 30 5 40 6 50
-- SAMPLE 2 --
10 7
5 1 0 3 10 5 20 7 30 9 40
4 2 20 1 40 2 60 1 80
10 5 30 4 40 3 50 4 60 5 70 6 80 7 90 8 100 9 110 10 120
6 10 10 8 20 6 30 4 40 2 50 1 60
9 10 10 9 20 8 30 7 40 6 50 5 60 4 70 5 80 6 90
5 6 60 7 70 8 80 9 90 10 110
5 2 10 4 20 6 30 8 40 10 50


Sample Output
------ ------

-- SAMPLE 1 --
50
-- SAMPLE 2 --
110


File:      spacetravel.txt
Author:    Bob Walton <walton@seas.harvard.edu>
Date:      Sat Oct  7 14:08:33 EDT 2017

The authors have placed this file in the public domain;
they make no warranty and accept no liability for this
file.
```

Senplosion

You are housing manager of the Quorzup habitat orbiting the sun Q5874 in the Goose Down spiral of the galaxy. Quorzup consists of many hubs connected by tubes.

You have been asked to accept a ship full of Sensites who will be staying permanently. The hubs have many empty living modules so space per se is not a problem.

However, if two Sensites get too close to one another, aromas they give off mix and cause a senplosion that makes everyone nearby very, very uncomfortable. So there are rules:

(1) No two Sensites may live in the same hub.

(2) No two Sensites may live in two hubs that are directly connected by a single tube.

You need to find out quickly how many Sensites your habitat can house.

Input

For each test case, first a line that gives the test case name. Then a line of the form

```
N
```

giving the number of hubs in the habitat. This is followed by N hub description lines each of the form

```
X Y Z
```

giving the location (X,Y,Z) of the hub relative to the habitat center. Hubs are given identifiers 1, 2, 3, ..., N in the order of their description line. Lastly there are N-1 tube description lines each of the form

```
I J
```

specifying that the tube connects hub I directly to hub J.

The habitat builders designed the habitat to minimize the total length of all the tubes while also providing that any hub is reachable from any other hub by some path through tubes and other hubs.

```
2 <= N <= 100,000
-1,000,000 <= X,Y,Z <= +1,000,000
```

Input terminates with an end of file. The test case name line is at most 80 characters.

Output

For each test case, first an exact copy of the test case name line. Then a line containing just the maximum number of Sensites the habitat can accommodate while obeying the rules.

Sample Input

```
------ -----

-- SAMPLE 1 --
9
-200 0 0
-100 0 0
0 -50 -50
0 50 -50
0 -50 50
0 50 50
0 0 0
100 0 0
200 0 0
7 3
7 4
7 5
7 6
2 1
7 2
8 7
9 8
```

```
-- SAMPLE 2 --
12
-200 0 0
-200 50 50
-200 50 -50
-200 -50 -50
-200 -50 50
-100 0 0
100 0 0
200 -50 -50
200 -50 50
200 50 50
200 50 -50
200 0 0
2 1
3 1
4 1
5 1
12 8
12 9
12 10
12 11
6 1
12 7
7 6
```

```
Sample Output
------ ------

-- SAMPLE 1 --
6
-- SAMPLE 2 --
9


File:      senplosion.txt
Author:    Bob Walton <walton@seas.harvard.edu>
Date:      Sat Oct  7 14:10:44 EDT 2017

The authors have placed this file in the public domain;
they make no warranty and accept no liability for this
file.
```

Tri-Mare

You live and work in the city of Tri-Mare, which is built in a shallow section of the sea (like Venice) so that people travel around it in boats. You want to find the shortest path from your apartment to work. All the buildings in Tri-Mare are triangular prism high-rises with elevators at their corners. You go down one of the elevators from your apartment and drive your own boat to a corner of the building you work in and take the elevator up to your office.

You want to find the shortest path for the boat (ignoring your paths inside buildings). The boat can travel alongside a building, but not through a building.

Input

For each test case, first a line that gives the test case name. Then a line of the form

```
N
```

giving the number of buildings in Tri-Mare, and then N lines, one per building, of the form

```
X1 Y1 X2 Y2 X3 Y3
```

giving the 3 corners of the building, which are (X1,Y1), (X2,Y2), and (X3,Y3). All numbers are integers. The buildings are numbered 1, 2, ..., N in the order of their description lines. You live in building 1 and work in building N.

```
2 <= N <= 100
-1,000,000 <= Xi,Yi <= +1,000,000
```

Input terminates with an end of file. The test case name line is at most 80 characters.

Output

For each test case, first an exact copy of the test case name line. Then a line containing just the length of the shortest path from any corner of building 1 to any corner of building N. The length must be accurate to one part in 10**5 (use default floating point format which prints 6 digits of precision).

Sample Input
------ -----

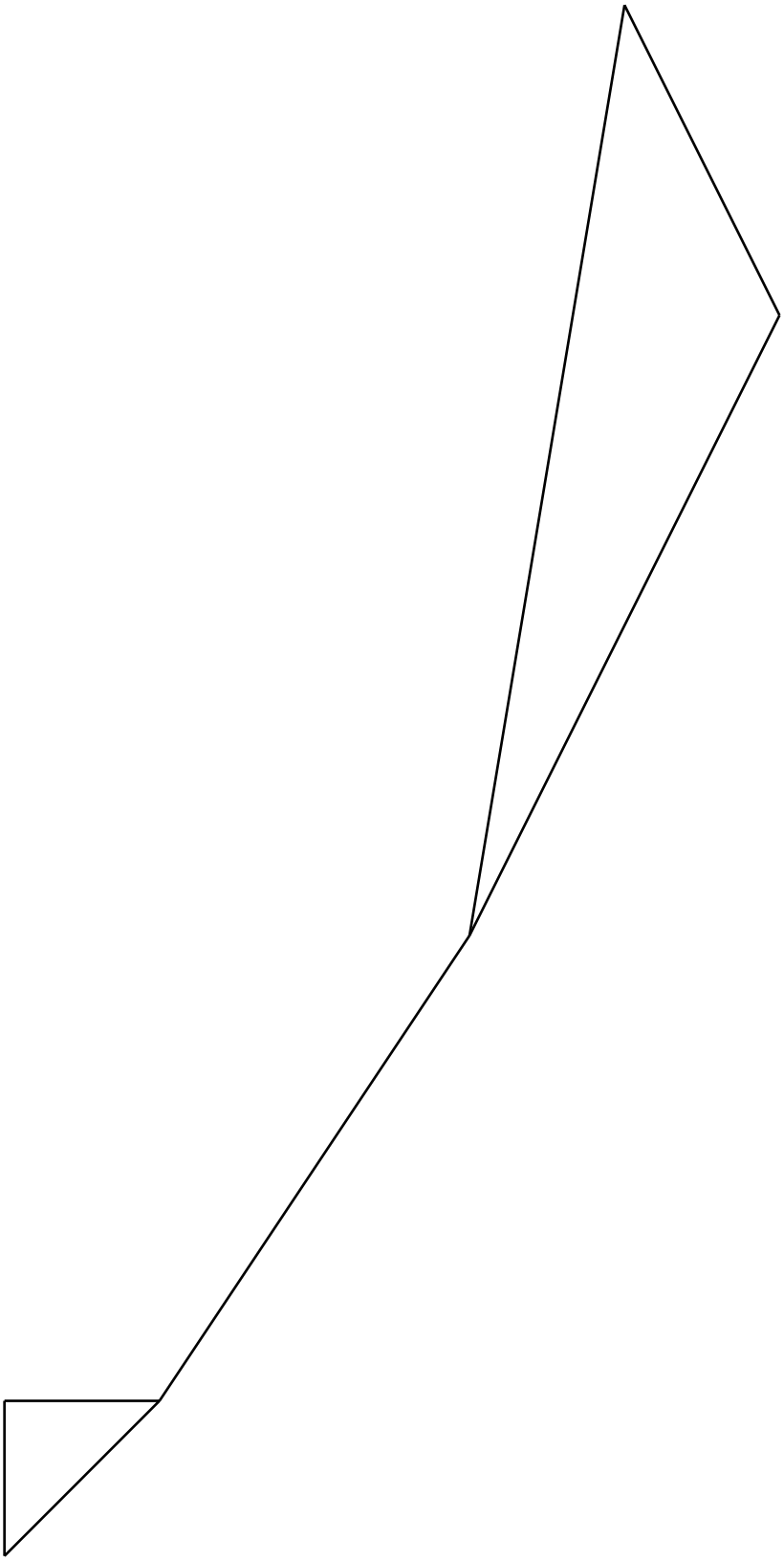
```
-- SAMPLE 1 --
2
  0   0  10  10   0  10
 30  40  50  80  40 100
-- SAMPLE 2 --
3
  0   0  10  10   0  10
 15  25  35  15  25  55
 30  40  50  80  40 100
```

Sample Output
------ ------

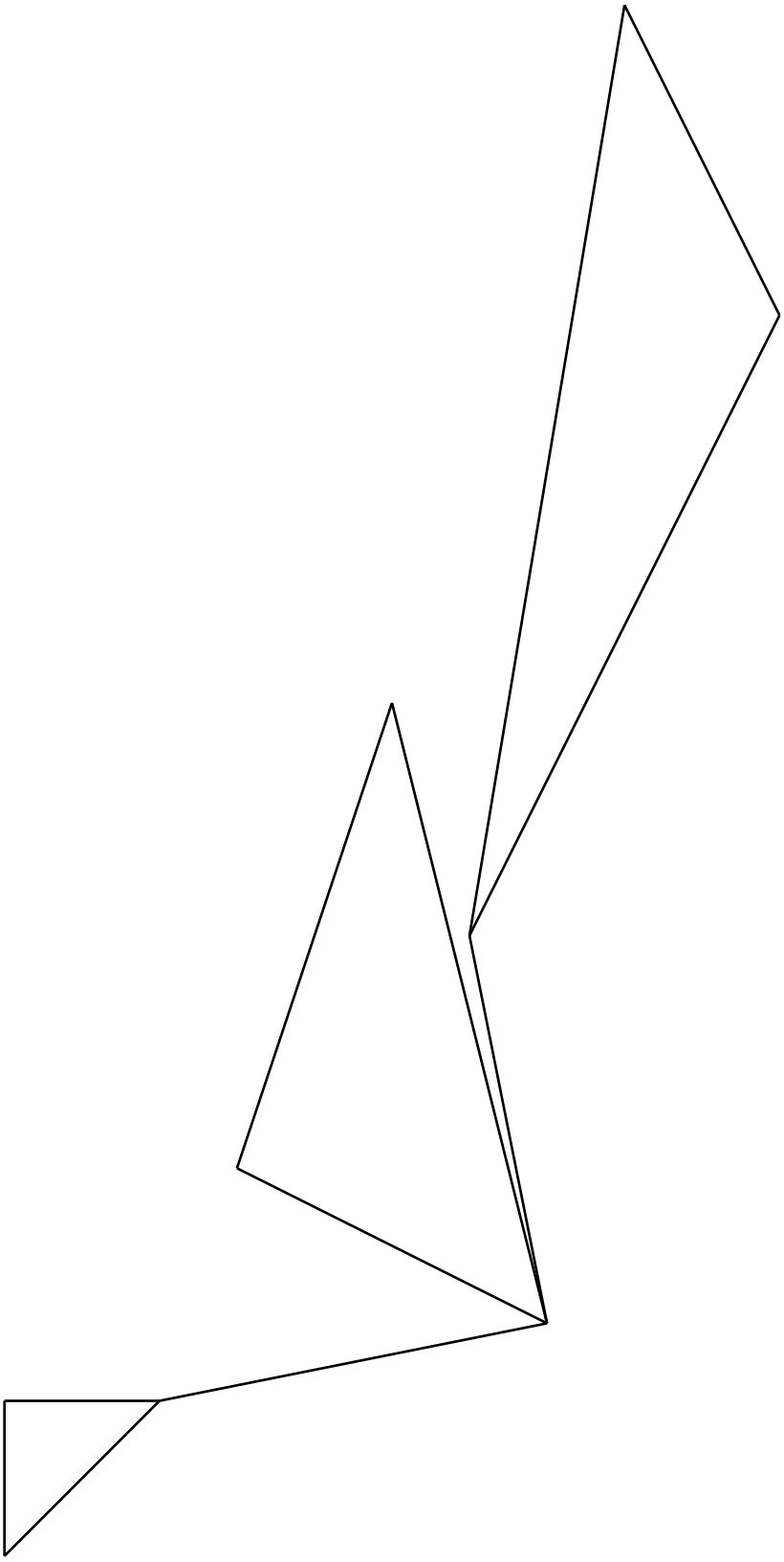
```
-- SAMPLE 1 --
36.0555
-- SAMPLE 2 --
50.9902
```

```
File:      trimare.txt
Author:    Bob Walton <walton@seas.harvard.edu>
Date:      Sat Oct  7 14:12:00 EDT 2017

The authors have placed this file in the public domain;
they make no warranty and accept no liability for this
file.
```

-- SAMPLE 1 --



-- SAMPLE 2 --



Separation

In a network, a node that can disconnect the network if it fails is called a ‘cut vertex’ or ‘articulation point’. An edge that can disconnect the network if it fails is called a ‘cut edge’ or ‘bridge’. More specifically, an articulation point is a vertex of a connected undirected graph which when removed, along with all the edges of which it is an end point, disconnects the graph. And a bridge is an edge of a connected undirected graph which when removed disconnects the graph.

You have been asked to find all the articulation points and bridges of a very large network (undirected graph).

Input

For each test case, first a line that gives the test case name. Then a line of the form

M N

giving the number of vertices M and number of edges N. Vertices are given identifiers 1, 2, 3, ..., M. Then N lines, one per edge, each of the form

i j

specifying the the edge connects vertex i to vertex j.

Each edge appears only once.

2 <= M <= 50,000
1 <= N <= 1,000,000

Input terminates with an end of file. The test case name line is at most 80 characters.

Output

For each test case, first an exact copy of the test case name line. Then a line containing

m n

where m is the number of articulation points and n is the number of bridges. This is followed by m lines, one for each articulation point, each of the form

i

where i is the vertex ID of the articulation point. These lines MUST BE SORTED in order of increasing i. Lastly there are n lines, one for each bridge, each of the form

i j

where the bridge connects vertices with IDs i and j. Here i < j is REQUIRED, and the bridge lines MUST BE SORTED in order of increasing i and for equal i in order of increasing j.

Sample Input
------ -----

```
-- SAMPLE 1 --
2 1
1 2
-- SAMPLE 2 --
4 4
1 2
2 3
3 4
4 1
-- SAMPLE 3 --
4 3
1 2
2 3
3 4
-- SAMPLE 4 --
7 8
1 2
1 3
2 3
3 4
4 5
5 6
6 7
7 4
-- SAMPLE 5 --
9 9
1 2
2 3
3 4
4 5
5 6
6 1
7 1
8 3
9 5
```

Sample Output
------ ------

```
-- SAMPLE 1 --
0 1
1 2
-- SAMPLE 2 --
0 0
-- SAMPLE 3 --
2 3
2
3
1 2
2 3
3 4
-- SAMPLE 4 --
2 1
3
4
3 4
-- SAMPLE 5 --
3 3
1
3
5
1 7
3 8
5 9
```

File: separation.txt
Author: Bob Walton <walton@seas.harvard.edu>
Date: Sat Sep 16 01:28:30 EDT 2017

Energy Coupons

You are a computer living on the planet E++, and you buy your energy from Dave's Energy Emporium (DEE). Everything is fast on E++, and energy charges are computed for each second of use. DEE has somewhat high prices, but excellent rebates, delivered as follows. The rebate period is P seconds. Every P seconds DEE gives you P rebate coupons. You can use these any time in the next P seconds, but only one can be used per second. Often the total coupon worth is 20% of what you paid DEE in the last P seconds, so DEE's is a good price if you use the coupons.

However, you have heard rumors of change at the top of DEE, and at the end of the current period, there is a surprise. Apparently Dave has been bought out by the Devil, and the new coupons are different. First, different coupons are frequently for different amounts. Second, instead of each coupon expiring at the end of the new P second period, each expires after a different number of seconds in that period, with most expiring in less than P seconds.

You find that if you simply used the coupons in order of expiration date, some of the coupons will expire before you can use them. So you need an algorithm to find the order in which you should use the coupons so that your total rebate is maximized.

Input

For each test case, first a line that gives the test case name. Then a line containing just

```
P
```

giving the number of seconds in a period, which is also the number of coupons issued at the beginning of the period. Then P coupon lines, each of the form:

```
V E
```

where V is the value of the coupon and E is the expiration time. More specifically, you can use the coupon in second t of the new period if and only if t <= E, where the seconds are numbered t = 1, 2, ..., P.

DEE has kindly sorted the coupons so their values are non-decreasing.

```
2 <= P <= 10,000,000
1 <= V <= 1,000,000
1 <= E <= P
```

Input terminates with an end of file. The test case name line is at most 80 characters.

Output

For each test case, first an exact copy of the test case name line. Then a line containing just

best-rebate OUT OF total-value

where ‘best-rebate’ is the maximum rebate you can achieve given the coupon values and expiration times, and ‘total-value’ is the sum of the values of all the coupons, what you would have received if there were no expiration times. Note that your rebate is not limited by the amount of energy you actually buy, as that is always more than the rebate.

Sample Input
------ -----

```
-- SAMPLE 1 --
3
2 2
3 2
4 2
-- SAMPLE 2 --
3
2 2
3 1
4 3
-- SAMPLE 3 --
10
5 6
5 2
6 4
6 7
6 3
7 9
8 10
8 1
9 3
9 2
```

Sample Output
------ ------

```
-- SAMPLE 1 --
7 OUT OF 9
-- SAMPLE 2 --
9 OUT OF 9
-- SAMPLE 3 --
58 OUT OF 69
```

```
File:      energy.txt
Author:    Bob Walton <walton@seas.harvard.edu>
Date:      Sat Oct  7 14:14:04 EDT 2017

The authors have placed this file in the public domain;
they make no warranty and accept no liability for this
file.
```

Tour Bets

Tourists come to the colorful Kingdom of Fate to tour its quaint villages and spin the fantastic wheels of fortune in each village. There are rewards for those who do so, both emotional and monetary, but we need not go into those here.

There are also rewards for the bookies of Fate who take on the following interesting kind of bet. Every tourist records the villages they visit and the result of their spinning the wheel of fortune in that village. The bookies publish the lists of spin results for each tourist and take bets on the list of villages visited by the tourist, which of course is not published until the day the bets are paid.

Each wheel of fortune is divided into 50 parts each labeled with a capital letter. The tourists visit the villages by traveling roads between them, taking a whole day on each road, and resting one night in the village at the end of the road before traveling on to the next village the next day. They spin the wheel at a village as soon as they arrive at the village. Tours start and end at the kingdom's only port, which has no wheel.

A friend of yours has made a model of Fate that gives the probability that a tourist who traveled a road one day will travel another road on the next day. It is also possible for the tourist to double back. The 50 possible outcomes of each wheel are equally probable and known.

You have been asked to compute a list of the most probable tours for a tourist given the spin results of the tourist and your friend's model.

Input

For each test case, first a line that gives the test case name. Then a line of the form

```
V L B
```

giving the number of villages V, length of output list L, and number of bets B. This is followed by V lines each containing 50 characters that list the wheel labels for village 1, 2, 3, ..., V in order.

Next are lines of the form:

```
w1 w2 w3 p(w1,w2,w3)
```

Here w1, w2, and w3 are location IDs. These can be village IDs over the range 1, 2, 3, ..., V, or they can be the port ID 0. In the following v1, v2, v3 are village IDs, but NEVER the port ID. Then p(w1,w2,w3) is a probability defined as follows:

```
p(w1,w2,v3) is the probability of going to village
            v3 if the last two places the tour was
            at are w1 and w2, respectively, and
            the tour does not go back to the port.
            Note that p(0,0,v3) is the probability
            the tour starts at v3.

p(w1,v2,0) = 1 if and only if there is a road from
               v2 to the port and w1 != v2,
           = 0 otherwise,
```

Also it is true that:

```
    p(w1,v2,v2) = 0

    p(v1,v1,w2) = 0

    p(v1,0,v3) = 0

    p(v1,0,0) = 1

    p(0,0,0) = 0

    if ( p(0,0,v2) > 0 ) and w1 != v2,
    then p(w1,v2,0) = 1
```

Probabilities not explicitly specified are zero; none of the explicitly specified probabilities are zero.

Lastly there are B lines each representing one bet. Each of these contains just a string of capital letters denoting the results of the tourist wheel spins. The number of letters in one line is the number of villages T visited by the tourist (excluding the port), and the i'th letter is the result of the tourist's wheel spin in the i'th village visited.

```
    2 <= V <= 100
    1 <= L <= 10
    1 <= B <= 100
    1 <= T <= 50
    0 <=  p(w1,w2,w3) <= 1.0
    if w1 != w2 != 0 or w1 == w2 == 0:
        sum { p(w1,w2,v3) : 1 <= v3 <= V } = 1
    Also see special cases above.
```

Input ends with an end of file. Test case name lines are at most 80 characters.

Output

For each test case, first an exact copy of the test case name line. Then for each of the B bets the following lines.

First, a line copying the bet input. Then L lines of the form

```
        p v1 v2 ... vT
```

where v1 v2 ... vT are the villages visited by the tour in order (T is the number of villages visited, i.e., the length of the bet line), and p is the probability of the tour according to the model. The L lines must be in order of descending p, and must be the L most probable tours given the bet.

p is the conditional probability of the tour being v1, v2, ..., vT given that the spins are those of the bet. For example, in SAMPLE 1 below the joint probability of the bet ABB and the path 1 2 1 is 0.064, and the sum of the joint probability of the bet ABB over all paths (ALL PATHS, not just the 2 paths output) is 0.08, so the conditional probability of the path 1 2 1 given the bet ABB is 0.064 / 0.08 = 0.8. Similarly in SAMPLE 2 the joint probability of the bet ABC and the path 1 2 3 is 0.03024 and the sum of the joint probability of the bet ABC over all paths is 0.042775 so the conditional probability of the path 1 2 3 given the bet ABC is 0.03024/0.042775 = 0.706955.

The probabilities must be accurate to 5 digits of precision and may be printed in scientific notation (use of double precision numbers with default %g format for C, C++, and JAVA suffices). The judge's input will be such that no two of the L+1 most probable tours have the same probability to within this accuracy. The probabilities may be very small and require exponential notation when printed.

Sample Input

------ -----

-- SAMPLE 1 --

```
2 2 4
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAABBBBBBBBBB
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBAAAAAAAAAA
0 0 1 0.5
0 0 2 0.5
1 0 0 1
2 0 0 1
0 1 0 1
0 2 0 1
1 2 0 1
2 1 0 1
0 1 2 1.0
0 2 1 1.0
1 2 1 1.0
2 1 2 1.0
AB
BA
ABB
ABABABAB
```

-- SAMPLE 2 --

```
3 3 3
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAABBBBBBBBBBBBBBBCCCCC
BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBCCCCCCCCCCCCCCCCAAAAA
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCAAAAAAAAAAAAAAABBBBB
0 0 1 0.5
0 0 2 0.25
0 0 3 0.25
1 0 0 1
2 0 0 1
3 0 0 1
0 1 0 1
0 2 0 1
0 3 0 1
1 2 0 1
1 3 0 1
2 1 0 1
2 3 0 1
3 1 0 1
3 2 0 1
0 1 2 0.4
0 1 3 0.6
0 2 1 0.2
0 2 3 0.8
0 3 1 1
1 2 1 0.3
1 2 3 0.7
1 3 1 1
2 1 2 0.6
2 1 3 0.4
2 3 1 0.2
2 3 2 0.8
3 1 2 0.9
3 1 3 0.1
3 2 1 1
AAA
ABC
ACB
```

```
Sample Output
------ ------

-- SAMPLE 1 --
AB
0.941176 1 2
0.0588235 2 1
BA
0.941176 2 1
0.0588235 1 2
ABB
0.8 1 2 1
0.2 2 1 2
ABABABAB
0.999985 1 2 1 2 1 2 1 2
1.52586e-05 2 1 2 1 2 1 2 1
-- SAMPLE 2 --
AAA
0.7327 1 3 1
0.0915875 3 1 2
0.0569878 1 2 3
ABC
0.706955 1 2 3
0.142022 3 1 2
0.0504968 1 2 1
ACB
0.661697 1 3 1
0.117635 2 3 2
0.0827121 3 1 2


File:      tourbets.txt
Author:    Bob Walton <walton@seas.harvard.edu>
Date:      Tue Oct  3 01:49:04 EDT 2017

The authors have placed this file in the public domain;
they make no warranty and accept no liability for this
file.
```

Simple Polygons
------ --------

A polygon is a closed circular chain of line segments. A simple polygon is a polygon such that no two non-adjacent segments intersect each other.

A polygon's vertices are the ends of its line segments. A polygon may be defined by listing its vertices in the order that they are first encountered while traversing the chain of line segments, so each consecutive pair of vertices defines one of the line segments, and there is also a line segment connecting the first and last vertex.

Your friend Improbable George has come up with a new way to generate very large random simple polygons very fast. Unfortunately a very few of his polygons are not simple and have a few intersections. He wants you to come up with an equally fast way of checking whether a very large polygon is simple and identifying its few intersections if it is not. Here an intersection is defined as any pair of non-adjacent segments that touch each other.

Input

For each test case, first a line that gives the test case name. Then a line of the form

V

giving the number of vertices. Then V lines each giving the coordinates (Vx,Vy) of one vertex in the format:

Vx Vy

The vertices are given in the order they appear on the polygon boundary, with the last vertex being connected to the first vertex.

All input numbers are integers. No two vertices will have the same coordinates (this is a property of Improbable's polygon generator).

3 <= V <= 100,000
-1,000,000 <= Vx, Vy <= +1,000,000

Input ends with an end of file. Test case name lines are at most 80 characters.

Output

For each test case, first an exact copy of the test case name line. Then a line containing just

W

which is the number of intersections. Then W lines each containing

s1 s2

identifying pairs (s1,s2) of non-adjacent segments that intersect (i.e., that touch each other). Note you are outputting pairs of non-adjacent segments that touch each other, and not the intersection points themselves, so, for example, if 4 non-adjacent segments intersect at a single point you output 4*(4-1)/2 = 6 segment pairs.

A segment identifier s refers the segment whose first vertex is the s'th vertex input and whose second vertex is the s+1'st vertex input, except if s == V it is the 1'st vertex input instead.

In order to make judging easier, the intersections must be sorted. s1 < s2 is required. The intersections must be in order of increasing s1, and for those with equal s1, in order of increasing s2.

Segments may intersect at a point or by overlapping. The interior of a segment may contain a vertex of another segment, thus intersecting the other segment. However, two non-adjacent segments cannot intersect by sharing a vertex, as all vertices are unique.

Importantly, you may assume that

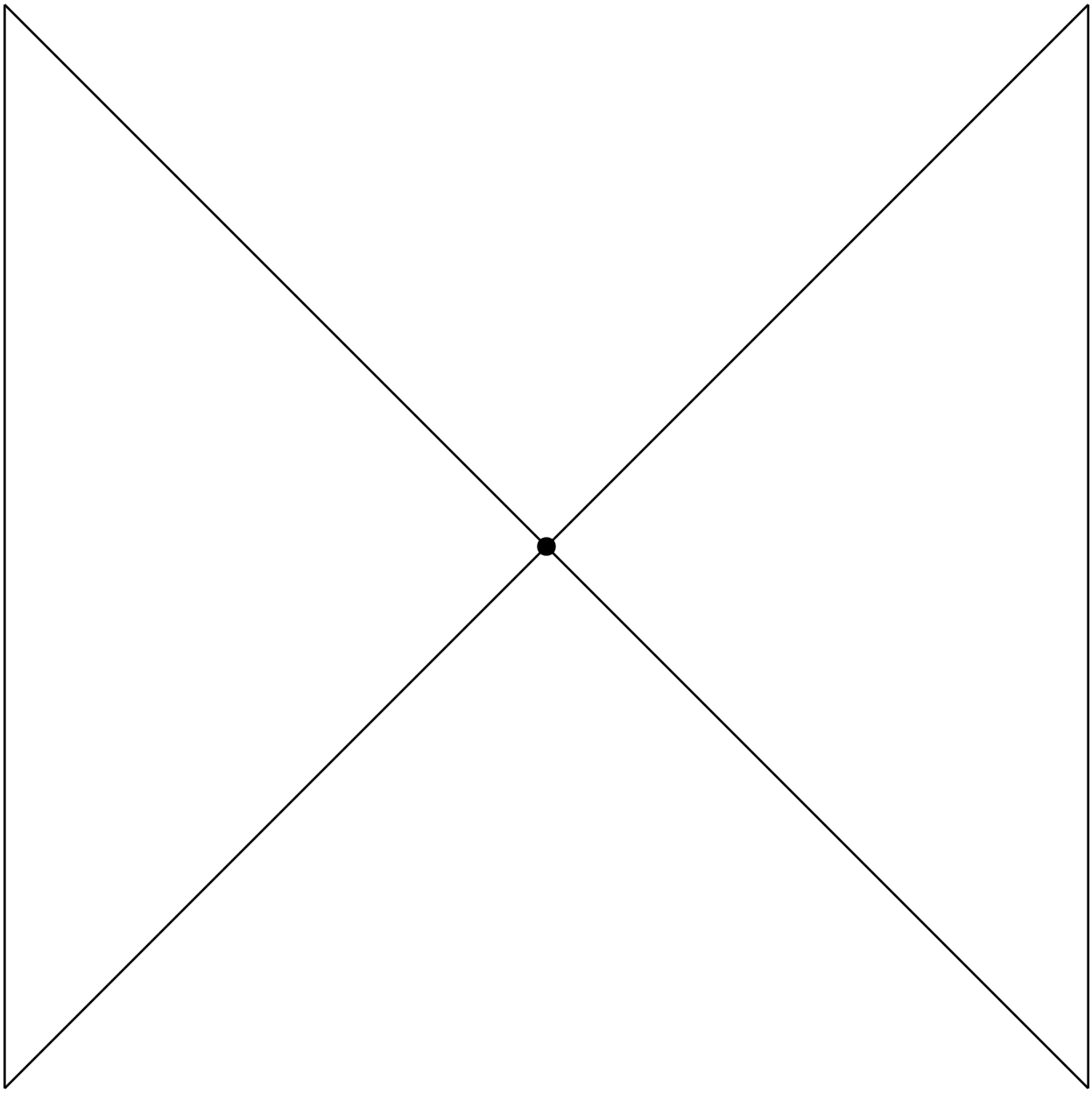
W <= 100

Sample Input

```
-- SAMPLE 1 --
4
 0  0
 0 10
10 00
10 10
-- SAMPLE 2 --
14
  0   0
100   0
 20  20
100  40
 60  60
100  80
 20 100
100 120
  0 120
 80 100
  0  80
 40  60
  0  40
 80  20
```

```
Sample Output
------ ------

-- SAMPLE 1 --
1
2 4
-- SAMPLE 2 --
4
2 14
3 13
6 10
7 9


File:      simplepoly.txt
Author:    Bob Walton <walton@seas.harvard.edu>
Date:      Sat Oct  7 14:16:36 EDT 2017

The authors have placed this file in the public domain;
they make no warranty and accept no liability for this
file.
```

-- SAMPLE 1 --



-- SAMPLE 2 --