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


PRACTICE PROBLEMS FOR BOSPRE 2017
-------- -------- --- ------ ----

These DO NOT COUNT and DO NOT APPEAR ON THE SCOREBOARD.

You may get help on any aspect of these problems during
the contest, including algorithm design and code, from
the host site staff, as long as they have time for you.


    practice/pangram
        A message that has it all.
        Harvard Selection Contest Fall 2017
        Author: Bob Walton

    practice/relativeneighbor
        I’m closer than he is.
        Boston Preliminary 2011
        Author: Bob Walton

    practice/btlabels
        Where in the tree are we?
        Author: Bob Walton
```

Pangram

A pangram is an English sentence that uses every letter of the alphabet at least once. An example is

```
    The quick brown fox jumps over a lazy dog.
```

You are given a sentence and asked to say if it is a pangram, and if not, to list the letters it does not have in alphabetical order.

Input

For each of several test cases, one line containing the sentence. No line is longer than 80 characters. Input ends with an end of file.

Output

For each test case, first an exact copy of the test case input line, and then one line that contains either just the word ‘PANGRAM’ (you must capitalize this) or has the format

```
        MISSING <letters>
```

where <letters> consists of all the letters missing from the sentence. Punctuation should be ignored on input. Letter case should also be ignored on input, e.g., ‘I’ and ‘i’ should be considered the same. On output all letters in <letters> must be lower case, and the letters in <letters> MUST BE IN ALPHABETICAL ORDER.

Sample Input

```
The quick brown fox jumps over a lazy dog.
The quick brown fox jumps over the dog.
Help, I’m being eaten by a zephyr!
The five boxing wizards jump quickly.
The cow got tangled up with the moon and a spoon.
Five quacking wizards jump below my high wax statue.
```

Sample Output

```
The quick brown fox jumps over a lazy dog.
PANGRAM
The quick brown fox jumps over the dog.
MISSING alyz
Help, I’m being eaten by a zephyr!
MISSING cdfjkoqsuvwx
The five boxing wizards jump quickly.
PANGRAM
The cow got tangled up with the moon and a spoon.
MISSING bfjkqrvxyz
Five quacking wizards jump below my high wax statue.
PANGRAM
```

```
File:      pangram.txt
Author:    Bob Walton <walton@seas.harvard.edu>
Date:      Fri Aug 25 03:56:16 EDT 2017
```

Relative Neighbor Graphs

Given a set of points in a plane, the associated relative neighbor graph has an edge between two points P1 and P2 if and only if there is NO point P3 such that

$$d(P3,P1) < d(P1,P2) \quad \text{and } d(P3,P2) < d(P1,P2)$$

where d(Px,Py) is the distance between Px and Py.

You have been asked to compute the relative neighbor graph of a set of points.

Input

For each test case, first a line containing just the test case name, and then one line containing just

        N

the number of points. Then N lines each containing

        X Y

which describes the point (X,Y). The points are given identifiers 1, 2, ..., N in the order that their coordinate lines appear in the input. Coordinates are all integers.

        $3 <= N <= 100$
        $-1,000,000 <= X,Y <= +1,000,000$

The test case name contains at most 80 characters. Input ends with an end of file.

Output

For each test case, first a line that is an exact copy of the test case name input line. Then one line for each edge in the relative neighbor graph, this line having the format

        i j

to specify that there is an edge from point i to point j. Here 1 <= i,j <= N.

This output MUST BE SORTED so that i < j and the edges are in order of increasing i, and for equal i, in the order of increasing j.

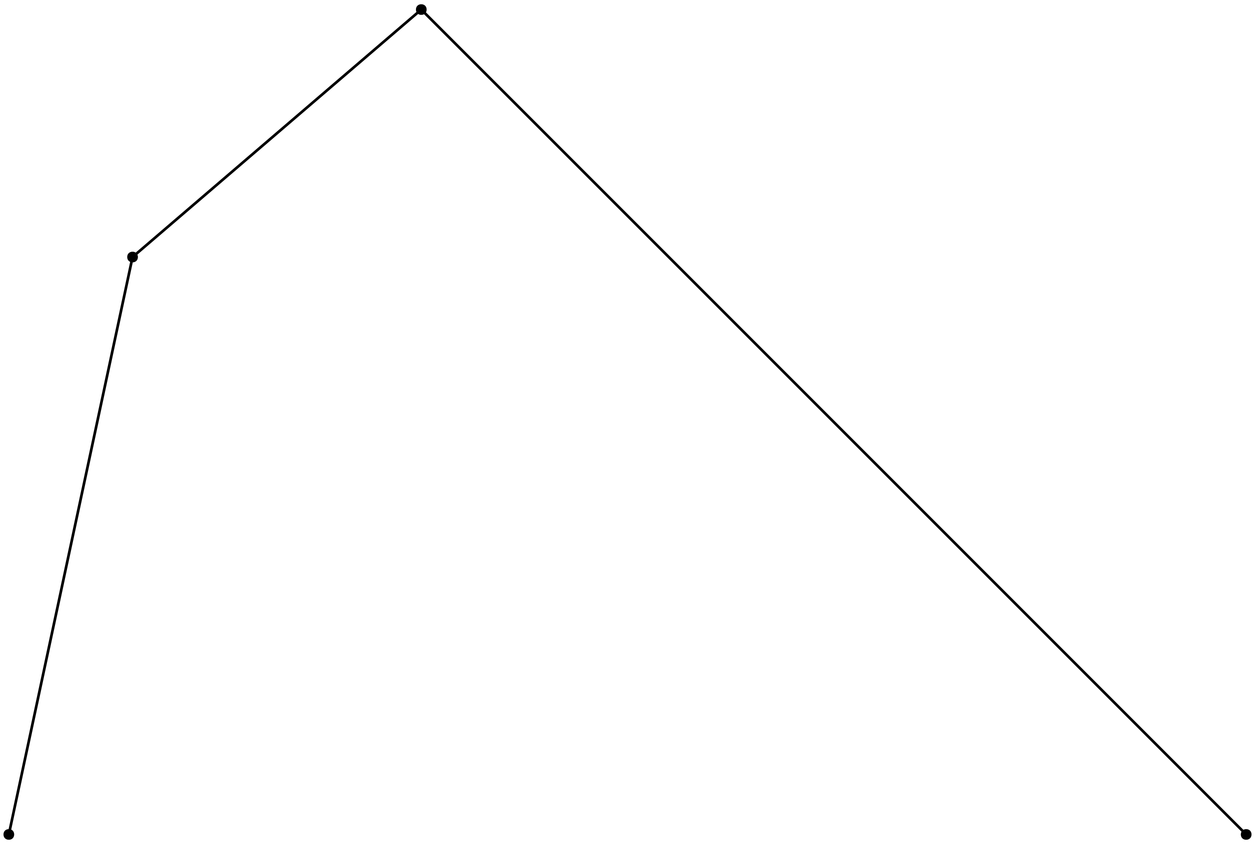
The test case output is ended with a line containing just ‘*’.

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

-- SAMPLE 1 --
4
-1000     0
+2000     0
    0 +2000
 -700  1400
-- SAMPLE 2 --
5
-1000 -1000
+1000 -1000
+1000 +1000
-1000 +1000
 -700     0
-- SAMPLE 3 --
5
-1000 -1000
+1000 -1000
+1000 +1000
-1000 +1000
-1300     0
```

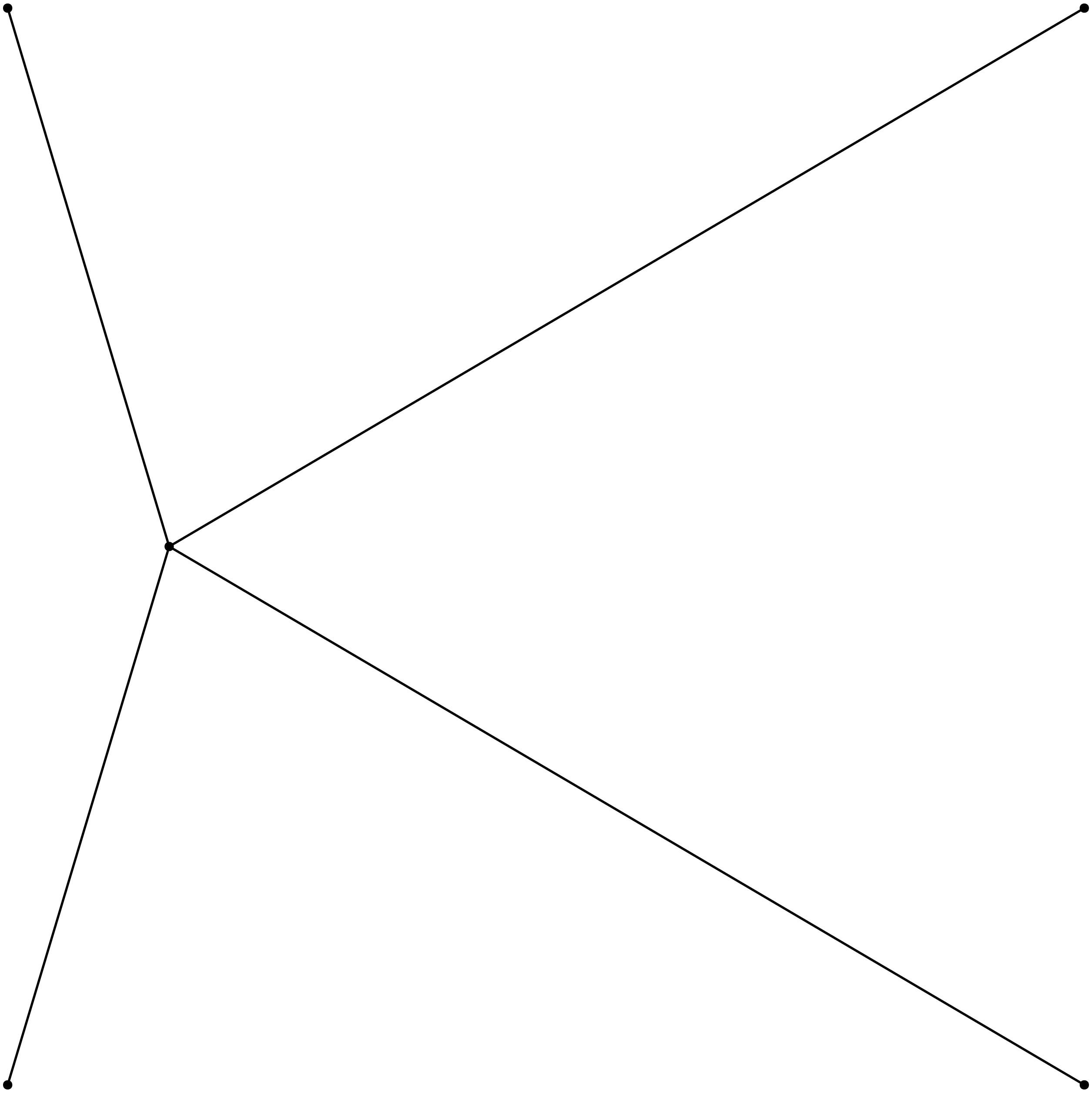
```
Sample Output
------ ------

-- SAMPLE 1 --
1 4
2 3
3 4
*
-- SAMPLE 2 --
1 5
2 5
3 5
4 5
*
-- SAMPLE 3 --
1 2
1 5
2 3
3 4
4 5
*
```

```
File:      relativeneighbor.txt
Author:    Bob Walton <walton@seas.harvard.edu>
           Upgraded by Bob Walton in 2017
Date:      Sat Oct  7 10:48:45 EDT 2017
           Original: Sun Oct  2 03:59:50 EDT 2011
```

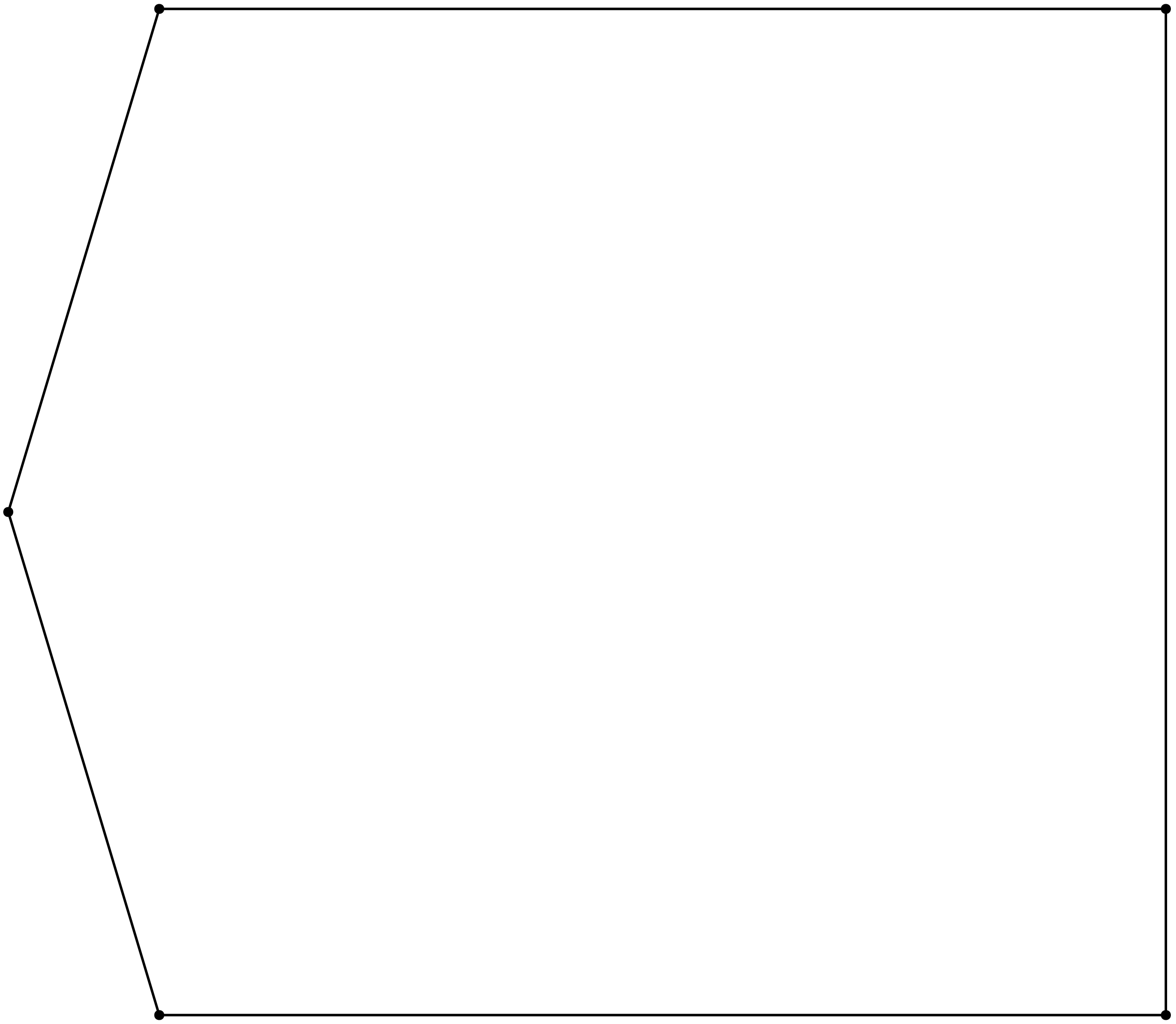
-- SAMPLE 1 --



-- SAMPLE 2 --



-- SAMPLE 3 --



Binary Tree Labels
------ ---- ------

There are several ways to label the nodes of a binary tree. For example, they can be labeled 1, 2, 3, ..., N from left to right and top to bottom, as in:

```
                        (root)
                          1
                         / \
                ---------   ---------
               /                     \
              2                       3
             / \                     / \
          ---   ---               ---   ---
         /         \             /         \
        4           5           6           7
       / \         / \         / \         / \
      /   \       /   \       /   \       /   \
     8     9     10   11     12   13     14   15
```

We will call this kind of labeling ‘lexical’, because of its left to right top to bottom order. With lexical labeling the left child of node n is 2n and the right child is 2n+1. Also the parent of child n is n/2, where we use integer division and discard the remainder.

Another kind of labeling is by binary strings, as in:

```
                        (root)
                         / \
                ---------   ---------
               /                     \
              0                       1
             / \                     / \
          ---   ---               ---   ---
         /         \             /         \
        00         01           10         11
       /  \       /  \         /  \       /  \
      /    \     /    \       /          /    \
    000    001 010    011    100   101  110   111
```

Here the root is labeled by the empty string, the left child of a node labeled s has label s0, and the right child has label s1. The parent of a node with label s has as label s with its last binary digit removed.

You are being asked to store values into or read values from nodes of a binary tree, using either lexical or binary string labels.

Input

For each test case, first a line that gives the test case name. Then lines each of which have one of the 4 forms:

```
        L n v
        B s v
        L n ?
        B s ?
```

Lastly a line containing just ‘*’.

In the above a line beginning with ‘L n’ gives a lexical label n for a binary tree node, and a line beginning with ‘B s’ gives a binary string label s for a tree node. v is an integer > 0 that is a value to be stored in the node. ? is the character ‘?’ which means the node is to be queried to determine its value.

```
        2 <= n <= 2047
        1 <= length s <= 10
        1 <= v <= 10,000
```

Note the root is never referenced.

The lines beginning with ‘L’ or ‘B’ should be thought of as executing in the order they are given, starting with a sufficiently large binary tree none whose nodes have a value at the beginning of the test case. With this in mind, the input will be such that no query reads a node without a value, or in other words, each query gives the label of a node whose label was given previously in the same test case by a non-query that stored a value in the labeled node.

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

Output

For each test case, an exact copy of the input of the test case but with each ? replaced by the value of the node at the time the query was read.

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

```
-- SAMPLE 1 --
L 7 10
B 01 20
L 5 ?
B 11 ?
L 5 60
B 11 50
L 7 ?
B 01 ?
*
-- SAMPLE 2 --
L 14 1400
L 15 1500
L 30 3000
L 31 3100
B 110 ?
B 111 ?
B 1110 ?
B 1111 ?
*
```

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

-- SAMPLE 1 --
L 7 10
B 01 20
L 5 20
B 11 10
L 5 60
B 11 50
L 7 50
B 01 60
*
-- SAMPLE 2 --
L 14 1400
L 15 1500
L 30 3000
L 31 3100
B 110 1400
B 111 1500
B 1110 3000
B 1111 3100
*



File:      btlabels.txt
Author:    Bob Walton <walton@seas.harvard.edu>
Date:      Sat Oct  7 11:05:15 EDT 2017
```