Problem A
Lost in Space
Input: lisp.txt

William Robinson was completely puzzled in the music room; he could not find his triangle in his bag. He was sure that he had prepared it the night before. He remembered its clank when he had stepped on the school bus early that morning. No, not in his dream. His triangle was quite unique: no two sides had the same length, which made his favorite peculiar jingle. He insisted to the music teacher, Mr. Smith, that his triangle had probably been stolen by those aliens and thrown away into deep space.

Your mission is to help Will find his triangle in space. His triangle has been made invisible by the aliens, but candidate positions of its vertices are somehow known. You have to tell which three of them make his triangle. Having gone through worm-holes, the triangle may have changed its size. However, even in that case, all the sides are known to be enlarged or shrunk equally, that is, the transformed triangle is *similar* to the original.

Input

The very first line of the input has an integer which is the number of data sets. Each data set gives data for one incident such as that of Will's. At least one and at most ten data sets are given.

The first line of each data set contains three decimals that give lengths of the sides of the original triangle, measured in centimeters. Three vertices of the original triangle are named P, Q, and R. Three decimals given in the first line correspond to the lengths of sides QR, RP, and PQ, in this order. They are separated by one or more space characters.

The second line of a data set has an integer which is the number of points in space to be considered as candidates for vertices. At least three and at most thirty points are considered.

The rest of the data set are lines containing coordinates of candidate points, in light years. Each line has three decimals, corresponding to x, y, and z coordinates, separated by one or more space characters. Points are numbered in the order of their appearances, starting from one.

Among all the triangles formed by three of the given points, only one of them is *similar* to the original, that is, ratios of the lengths of any two sides are equal to the corresponding ratios of the original allowing an error of less than 0.01 percent. Other triangles have some of the ratios different from the original by at least 0.1 percent.

The origin of the coordinate system is not the center of the earth but the center of our galaxy. Note that negative coordinate values may appear here. As they are all within or close to our

galaxy, coordinate values are less than one hundred thousand light years. You don't have to take relativistic effects into account, i.e., you may assume that we are in a Euclidean space. You may also assume in your calculation that one light year is equal to 9.461×10^{12} kilometers.

A succeeding data set, if any, starts from the line immediately following the last line of the preceding data set.

Output

For each data set, one line should be output. That line should contain the point numbers of the three vertices of the similar triangle, separated by a space character. They should be reported in the order P, Q, and then R.

Sample Input

```
2
  50.36493   81.61338   79.96592
5
 -10293.83  -4800.033  -5296.238
  14936.30   6964.826   7684.818
 -4516.069   25748.41  -27016.06
  18301.59  -11946.25   5380.309
  27115.20   43415.93  -71607.81
  11.51547   13.35555   14.57307
5
 -56292.27   2583.892   67754.62
 -567.5082  -756.2763  -118.7268
 -1235.987  -213.3318  -216.4862
 -317.6108  -54.81976  -55.63033
  22505.44  -40752.88   27482.94
```

Output for the Sample Input

```
1 2 4
3 4 2
```

Problem B
Family Tree
Input: family.txt

A professor of anthropology was interested in people living in isolated islands and their history. He collected their family trees to conduct some anthropological experiment. For the experiment, he needed to process the family trees with a computer. For that purpose he translated them into text files. The following is an example of a text file representing a family tree.

```
John
 Robert
  Frank
  Andrew
 Nancy
  David
```

Each line contains the given name of a person. The name in the first line is the oldest ancestor in this family tree. The family tree contains only the descendants of the oldest ancestor. Their husbands and wives are not shown in the family tree. The children of a person are indented with one more space than the parent. For example, Robert and Nancy are the children of John, and Frank and Andrew are the children of Robert. David is indented with one more space than Robert, but he is not a child of Robert, but of Nancy. To represent a family tree in this way, the professor excluded some people from the family trees so that no one had both parents in a family tree.

For the experiment, the professor also collected documents of the families and extracted the set of statements about relations of two persons in each family tree. The following are some examples of statements about the family above.

```
John is the parent of Robert.
Robert is a sibling of Nancy.
David is a descendant of Robert.
```

For the experiment, he needs to check whether each statement is true or not. For example, the first two statements above are true and the last statement is false. Since this task is tedious, he would like to check it by a computer program.

Input

The input contains several data sets. Each data set consists of a family tree and a set of statements. The first line of each data set contains two integers n ($0 < n < 1000$) and

m ($0 < m < 1000$) which represent the number of names in the family tree and the number of statements, respectively. Each line of the input has less than 70 characters.

As a name, we consider any character string consisting of only alphabetic characters. The names in a family tree have less than 20 characters. The name in the first line of the family tree has no leading spaces. The other names in the family tree are indented with at least one space, *i.e.*, they are descendants of the person in the first line. You can assume that if a name in the family tree is indented with k spaces, the name in the next line is indented with at most $k + 1$ spaces. This guarantees that each person except the oldest ancestor has his or her parent in the family tree. No name appears twice in the same family tree. Each line of the family tree contains no redundant spaces at the end.

Each statement occupies one line and is written in one of the following formats, where X and Y are different names in the family tree.

X `is a child of` Y.
X `is the parent of` Y.
X `is a sibling of` Y.
X `is a descendant of` Y.
X `is an ancestor of` Y.

Names not appearing in the family tree are never used in the statements. Consecutive words in a statement are separated by a single space. Each statement contains no redundant spaces at the beginning and at the end of the line.

The end of the input is indicated by two zeros.

Output

For each statement in a data set, your program should output one line containing `True` or `False`. The first letter of `True` or `False` in the output must be a capital. The output for each data set should be followed by an empty line.

Sample Input

```
6 5
John
 Robert
  Frank
  Andrew
 Nancy
  David
Robert is a child of John.
Robert is an ancestor of Andrew.
Robert is a sibling of Nancy.
Nancy is the parent of Frank.
```

```
John is a descendant of Andrew.
2 1
abc
 xyz
xyz is a child of abc.
0 0
```

Output for the Sample Input

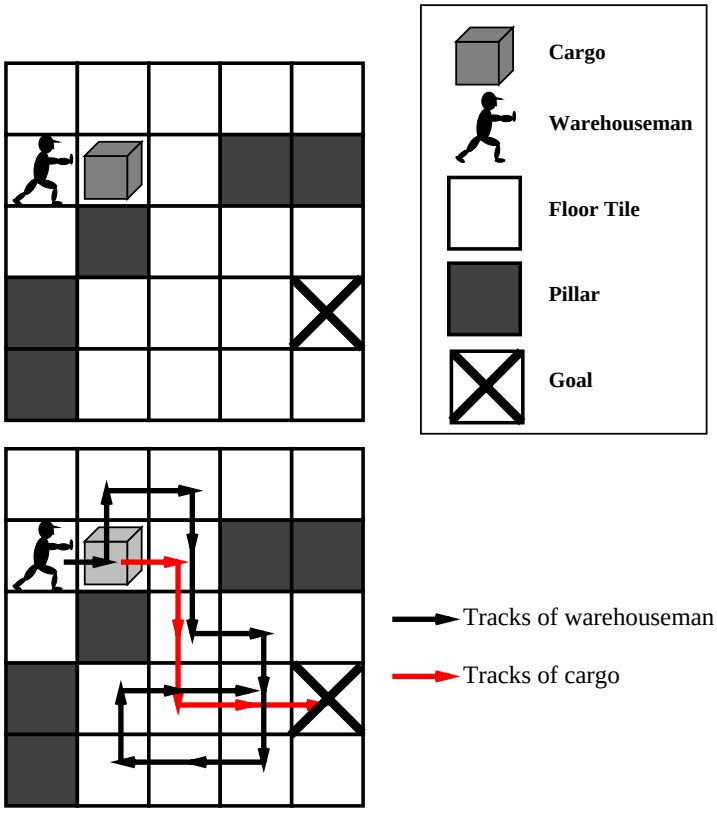
```
True
True
True
False
False

True
```

Problem C
Push!!
Input: push.txt

Mr. Schwarz was a famous powerful pro wrestler. He starts a part time job as a warehouseman. His task is to move a cargo to a goal by repeatedly pushing the cargo in the warehouse, of course, without breaking the walls and the pillars of the warehouse.

There may be some pillars in the warehouse. Except for the locations of the pillars, the floor of the warehouse is paved with square tiles whose size fits with the cargo. Each pillar occupies the same area as a tile.



Initially, the cargo is on the center of a tile. With one push, he can move the cargo onto the center of an adjacent tile if he is in proper position. The tile onto which he will move the cargo must be one of (at most) four tiles (i.e., east, west, north or south) adjacent to the tile where the cargo is present.

To push, he must also be on the tile adjacent to the present tile. He can only push the cargo in the same direction as he faces to it and he cannot pull it. So, when the cargo is on the tile next to a wall (or a pillar), he can only move it along the wall (or the pillar). Furthermore, once he places it on a corner tile, he cannot move it anymore.

He can change his position, if there is a path to the position without obstacles (such as the cargo and pillars) in the way. The goal is not an obstacle. In addition, he can move only in the four directions (i.e., east, west, north or south) and change his direction only at the center of a tile.

As he is not so young, he wants to save his energy by keeping the number of required pushes as small as possible. But he does not mind the count of his pedometer, because walking is very light exercise for him.

Your job is to write a program that outputs the minimum number of pushes required to move the cargo to the goal, if ever possible.

Input

The input consists of multiple maps, each representing the size and the arrangement of the warehouse. A map is given in the following format.

$$
\begin{array}{ccccc}
w & h & & & \\
d_{11} & d_{12} & d_{13} & \cdots & d_{1w} \\
d_{21} & d_{22} & d_{23} & \cdots & d_{2w} \\
 & & \cdots & & \\
d_{h1} & d_{h2} & d_{h3} & \cdots & d_{hw}
\end{array}
$$

The integers w and h are the lengths of the two sides of the floor of the warehouse in terms of widths of floor tiles. w and h are less than or equal to 7. The integer d_{ij} represents what is initially on the corresponding floor area in the following way.

0: nothing (simply a floor tile)
1: a pillar
2: the cargo
3: the goal
4: the warehouseman (Mr. Schwarz)

Each of the integers 2, 3 and 4 appears exactly once as d_{ij} in the map. Integer numbers in an input line are separated by at least one space character. The end of the input is indicated by a line containing two zeros.

Output

For each map, your program should output a line containing the minimum number of pushes. If the cargo cannot be moved to the goal, -1 should be output instead.

Sample Input

```
5 5
0 0 0 0 0
4 2 0 1 1
0 1 0 0 0
1 0 0 0 3
1 0 0 0 0
5 3
4 0 0 0 0
2 0 0 0 0
0 0 0 0 3
7 5
1 1 4 1 0 0 0
1 1 2 1 0 0 0
3 0 0 0 0 0 0
0 1 0 1 0 0 0
0 0 0 1 0 0 0
6 6
0 0 0 0 0 3
0 0 0 0 0 0
0 0 0 0 0 0
0 0 0 0 0 0
0 2 0 0 0 0
4 0 0 0 0 0
0 0
```

Output for the Sample Input

```
5
-1
11
8
```

Problem D
Pump up Batteries
Input: pattern.txt

Bill is a boss of security guards. He has pride in that his men put on wearable computers on their duty. At the same time, it is his headache that capacities of commercially available batteries are far too small to support those computers all day long. His men come back to the office to charge up their batteries and spend idle time until its completion. Bill has only one battery charger in the office because it is very expensive.

Bill suspects that his men spend much idle time waiting in a queue for the charger. If it is the case, Bill had better introduce another charger. Bill knows that his men are honest in some sense and blindly follow any given instructions or rules. Such a simple-minded way of life may lead to longer waiting time, but they cannot change their behavioral pattern.

Each battery has a data sheet attached on it that indicates the best pattern of charging and consuming cycle. The pattern is given as a sequence of pairs of consuming time and charging time. The data sheet says the pattern should be followed cyclically to keep the battery in quality. A guard, trying to follow the suggested cycle strictly, will come back to the office exactly when the consuming time passes out, stay there until the battery has been charged for the exact time period indicated, and then go back to his beat.

The guards are quite punctual. They spend not a second more in the office than the time necessary for charging up their batteries. They will wait in a queue, however, if the charger is occupied by another guard, exactly on first-come-first-served basis. When two or more guards come back to the office at the same instance of time, they line up in the order of their identification numbers, and, each of them, one by one in the order of that line, judges if he can use the charger and, if not, goes into the queue. They do these actions in an instant.

Your mission is to write a program that simulates those situations like Bill's and reports how much time is wasted in waiting for the charger.

Input

The input consists of one or more data sets for simulation.

The first line of a data set consists of two positive integers separated by a space character: the number of guards and the simulation duration. The number of guards does not exceed one hundred. The guards have their identification numbers starting from one up to the number of guards. The simulation duration is measured in minutes, and is at most one week, i.e., 10080 (min.).

Patterns for batteries possessed by the guards follow the first line. For each guard, in the order of identification number, appears the pattern indicated on the data sheet attached to his battery. A pattern is a sequence of positive integers, whose length is a multiple of two and does not exceed fifty. The numbers in the sequence show consuming time and charging time alternately. Those times are also given in minutes and are at most one day, i.e., 1440 (min.). A space character or a newline follows each number. A pattern is terminated with an additional zero followed by a newline.

Each data set is terminated with an additional empty line. The input is terminated with an additional line that contains two zeros separated by a space character.

Output

For each data set your program should simulate up to the given duration. Each guard should repeat consuming of his battery (i.e., being on his beat) and charging of his battery according to the given pattern cyclically. At the beginning, all the guards start their cycle simultaneously, that is, they start their beats and, thus, start their first consuming period.

For each data set, your program should produce one line containing the total wait time of the guards in the queue up to the time when the simulation duration runs out. The output should not contain any other characters.

For example, consider a data set:

```
3 25
3 1 2 1 4 1 0
1 1 0
2 1 3 2 0
```

The guard 1 tries to repeat 3 min. consuming, 1 min. charging, 2 min. consuming, 1 min. charging, 4 min. consuming, and 1 min. charging, cyclically. Yet he has to wait sometimes to use the charger, when he is on his duty together with the other guards 2 and 3. Thus, the actual behavior of the guards looks like:

```
          0         10        20
          |    |    |    |    |    |
guard 1: ***.**.****.***.**-.****.
guard 2: *.*-.*-.*-.*.*.*.*--.*.*-
guard 3: **.***--..**-.***..**.***
```

where "*" represents a minute spent for consuming, "." for charging, and "-" for waiting in the queue. At time 3, the guards 1 and 2 came back to the office and the guard 1 started charging while the guard 2 went into the queue. At time 6, all the guards came back to the office and the guard 1 started charging while the others went to the queue. When the charger got available at time 7, the guard 2 started charging, leaving the guard 3 in the queue. All those happened

are consequences of rules stated above. And the total time wasted in waiting for the charger becomes 10 minutes.

Sample Input

```
3 25
3 1 2 1 4 1 0
1 1 0
2 1 3 2 0

4 1000
80 20 80 20 80 20 80 20 0
80
20
0
80 20 90
10 80
20
0
90 10
0

0 0
```

Output for the Sample Input

```
10
110
```

Problem E
The Devil of Gravity
Input: gravity.txt

Haven't you ever thought that programs written in Java, C++, Pascal, or any other modern computer languages look rather sparse? Although most editors provide sufficient screen space for at least 80 characters or so in a line, the average number of significant characters occurring in a line is just a fraction. Today, people usually prefer readability of programs to efficient use of screen real estate.

Dr. Faust, a radical computer scientist, believes that editors for real programmers shall be more space efficient. He has been doing research on saving space and invented various techniques for many years, but he has reached the point where no more essential improvements will be expected with his own ideas.

After long thought, he has finally decided to take the ultimate but forbidden approach. He does not hesitate to sacrifice anything for his ambition, and asks a devil to give him supernatural intellectual powers in exchange with his soul. With the transcendental knowledge and ability, the devil provides new algorithms and data structures for space efficient implementations of editors.

The editor implemented with those evil techniques is beyond human imaginations and behaves somehow strange. The mighty devil Dr. Faust asks happens to be the devil of gravity. The editor under its control saves space with magical magnetic and gravitational forces.

Your mission is to defeat Dr. Faust by re-implementing this strange editor without any help of the devil. At first glance, the editor looks like an ordinary text editor. It presents texts in two-dimensional layouts and accepts editing commands including those of cursor movements and character insertions and deletions. A text handled by the devil's editor, however, is partitioned into text segments, each of which is a horizontal block of non-blank characters. In the following figure, for instance, four text segments "`abcdef`", "`ghijkl`", "`mnop`", "`qrstuvw`" are present and the first two are placed in the same row.

```
abcdef  ghijkl
     mnop
qrstuvw
```

The editor has the following unique features.

1. A text segment without any supporting segments in the row immediately below it falls by the evil gravitational force.

2. Text segments in the same row and contiguous to each other are concatenated by the evil magnetic force.

For instance, if characters in the segment "`mnop`" in the previous example are deleted, the two segments on top of it fall and we have the following.

```
abcdef
qrstuvw ghijkl
```

After that, if "`x`" is added at the tail (i.e., the right next of the rightmost column) of the segment "`qrstuvw`", the two segments in the bottom row are concatenated.

```
abcdef
qrstuvwxghijkl
```

Now we have two text segments in this figure. By this way, the editor saves screen space but demands the users' extraordinary intellectual power.

In general, after a command execution, the following rules are applied, where S is a text segment, left(S) and right(S) are the leftmost and rightmost columns of S, respectively, and row(S) is the row number of S.

1. If the columns from left(S) to right(S) in the row numbered row(S)-1 (i.e., the row just below S) are empty (i.e., any characters of any text segments do not exist there), S is pulled down to row(S)-1 vertically, preserving its column position. If the same ranges in row(S)-2, row(S)-3, and so on are also empty, S is further pulled down again and again. This process terminates sooner or later since the editor has the ultimate bottom, the row whose number is zero.
2. If two text segments S and T are in the same row and right(S) $+1$ = left(T), that is, T starts at the right next column of the rightmost character of S, S and T are automatically concatenated to form a single text segment, which starts at left(S) and ends at right(T). Of course, the new segment is still in the original row.

Note that any text segment has at least one character. Note also that the first rule is applied prior to any application of the second rule. This means that no concatenation may occur while falling segments exist. For instance, consider the following case.

```
   dddddddd
   cccccccc
bbbb
aaa
```

If the last character of the text segment "bbbb" is deleted, the concatenation rule is not applied until the two segments "cccccccc" and "dddddddd" stop falling. This means that "bbb" and "cccccccc" are not concatenated.

The devil's editor has a cursor and it is always in a single text segment, which we call the current segment. The cursor is at some character position of the current segment or otherwise at its tail. Note that the cursor cannot be a support. For instance, in the previous example, even if the cursor is at the last character of "bbbb" and it stays at the same position after the deletion, it cannot support "cccccccc" and "dddddddd" any more by solely itself and thus those two segments shall fall. Finally, the cursor is at the leftmost "d"

The editor accepts the following commands, each represented by a single character.

- F: Move the cursor forward (i.e., to the right) by one column in the current segment. If the cursor is at the tail of the current segment and thus it cannot move any more within the segment, an error occurs.
- B: Move the cursor backward (i.e., to the left) by one column in the current segment. If the cursor is at the leftmost position of the current segment, an error occurs.
- P: Move the cursor upward by one row. The column position of the cursor does not change. If the new position would be out of the legal cursor range of any existing text segment, an error occurs.
- N: Move the cursor downward by one row. The column position of the cursor does not change. If the cursor is in the bottom row or the new position is out of the legal cursor range of any existing text segment, an error occurs.
- D: Delete the character at the cursor position. If the cursor is at the tail of the current segment and so no character is there, an error occurs. If the cursor is at some character, it is deleted and the current segment becomes shorter by one character. Neither the beginning (i.e., leftmost) column of the current segment nor the column position of the cursor changes. However, if the current segment becomes empty, it is removed and the cursor falls by one row. If the new position would be out of the legal cursor range of any existing text segment, an error occurs.

  Note that after executing this command, some text segments may lose their supports and be pulled down toward the hell. If the current segment falls, the cursor also falls with it.
- C: Create a new segment of length one immediately above the current cursor position. It consists of a copy of the character under the cursor. If the cursor is at the tail, an error occurs. Also if the the position of the new segment is already occupied by another segment, an error occurs. After executing the command, the created segment becomes the new current segment and the column position of the cursor advances to the right by one column.
- Lowercase and numeric characters ('a' to 'z' and '0' to '9'): Insert the character at the current cursor position. The current segment becomes longer by one character. The cursor moves forward by one column. The beginning column of the current segment does not change.

Once an error occurs, the entire editing session terminates abnormally.

Input

The first line of the input contains an integer that represents the number of editing sessions. Each of the following lines contains a character sequence, where the first character is the initial character and the rest represents a command sequence processed during a session. Each session starts with a single segment consisting of the initial character in the bottom row. The initial cursor position is at the tail of the segment. The editor processes each command represented by a character one by one in the manner described above.

You may assume that each command line is non-empty and its length is at most one hundred. A command sequence ends with a newline.

Output

For each editing session specified by the input, if it terminates without errors, your program should print the current segment at the completion of the session in a line. If an error occurs during the session, just print "`ERROR`" in capital letters in a line.

Sample Input

```
3
12BC3BC4BNBBDD5
aaaBCNBBBCb
aaaBBCbNBC
```

Output for the Sample Input

```
15234
aba
ERROR
```

Problem F
Numoeba
Input: numoeba.txt

A scientist discovered a strange variation of amoeba. The scientist named it *numoeba*. A numoeba, though it looks like an amoeba, is actually a community of cells, which always forms a tree.

The scientist called the cell *leader* that is at the root position of the tree. For example, in Fig. 1, the leader is A. In a numoeba, its leader may change time to time. For example, if E gets new leadership, the tree in Fig. 1 becomes one in Fig. 2. We will use the terms root, leaf, parent, child and subtree for a numoeba as defined in the graph theory.

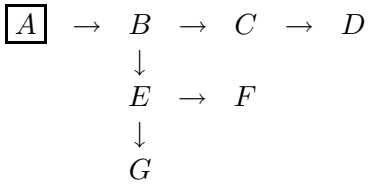


Fig. 1

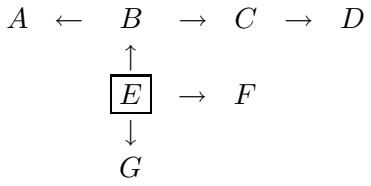


Fig. 2

Numoeba changes its physical structure at every biological clock by cell division and cell death. The leader may change depending on this physical change.

The most astonishing fact about the numoeba cell is that it contains an organic unit called *numbosome*, which represents an odd integer within the range from 1 to 12,345,677. At every biological clock, the value of a numbosome changes from n to a new value as follows:

1. The maximum odd factor of $3n + 1$ is calculated. This value can be obtained from $3n + 1$ by repeating division by 2 while even.

2. If the resulting integer is greater than 12,345,678, then it is subtracted by 12,345,678.

For example, if the numbosome value of a cell is 13, $13 \times 3 + 1 = 40$ is divided by $2^3 = 8$ and a new numbosome value 5 is obtained. If the numbosome value of a cell is 11,111,111, it changes to 4,320,989, instead of 16,666,667. If $3n + 1$ is a power of 2, yielding 1 as the result, it signifies the death of the cell as will be described below.

At every biological clock, the next numbosome value of every cell is calculated and the fate of the cell and thereby the fate of numoeba is determined according to the following steps.

1. A cell that is a leaf and increases its numbosome value is designated as a *candidate* leaf.

   A cell dies if its numbosome value becomes 1. If the dying cell is the leader of the numoeba, the numoeba dies as a whole. Otherwise, all the cells in the subtree from the dying cell (including itself) die. However, there is an exceptional case where the cells in the subtree do not necessarily die; if there is only one child cell of the dying non-leader cell, the child cell will replace the dying cell. Thus, a straight chain simply shrinks if its non-leader constituent dies.

   For example, consider a numoeba with the leader A below.

$$\begin{array}{ccccccccccc} \boxed{A} & \rightarrow & B & \rightarrow & C & \rightarrow & D & \rightarrow & E & \rightarrow & F \\ & & & & & & & & \downarrow & & \\ & & & & & & & & G & \rightarrow & H \end{array} \tag{1}$$

   If the leader A dies in (1), the numoeba dies.

   If the cell D dies in (1), (1) will be as follows.

$$\begin{array}{ccccccccc} \boxed{\text{A}} & \rightarrow & B & \rightarrow & C & \rightarrow & E & \rightarrow & F \\ & & & & & & \downarrow & & \\ & & & & & & G & \rightarrow & H \end{array} \tag{2}$$

   And, if the cell E dies in (1), (1) will be as follows.

$$\boxed{\text{A}} \rightarrow \quad B \quad \rightarrow \quad C \quad \rightarrow \quad D \tag{3}$$

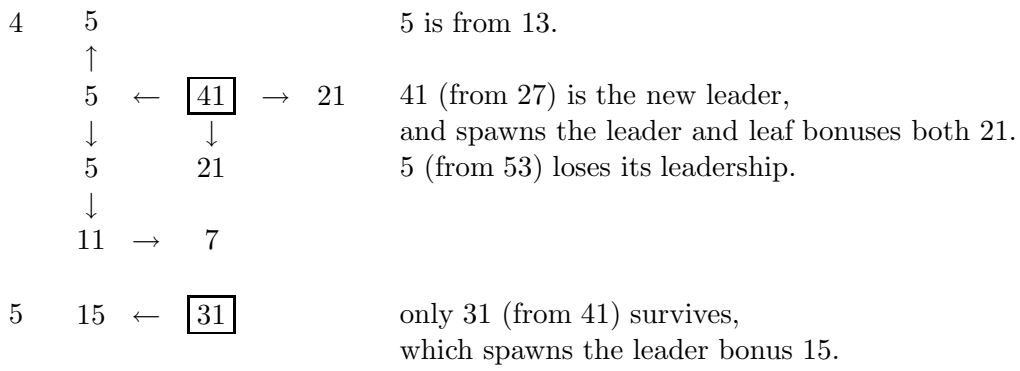
   Note that this procedure is executed sequentially, top-down from the root of the numoeba to leaves. If the cells E and F will die in (1), the death of F is not detected at the time the procedure examines the cell E. The numoeba, therefore, becomes (3). One should not consider in such a way that the death of F makes G the only child of E, and, therefore, G will replace the dying E.

2. If a *candidate* leaf survives with the numbosome value of n, it spawns a cell as its child, thereby a new leaf, whose numbosome value is the least odd integer greater than or equal to $(n + 1)/2$. We call the child *leaf bonus*.

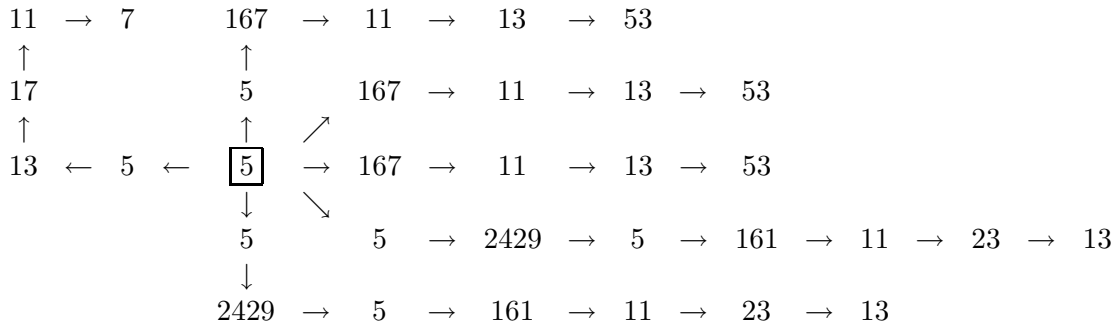
3. Finally, a new leader of the numoeba is selected, who has a unique maximum numbosome value among all the constituent cells. The tree structure of the numoeba is changed so that the new leader is its root, like what is shown in Fig. 1 and Fig. 2. Note that the parent-child relationship of some cells may be reversed by this leader change. When a new leader of a unique maximum numbosome value, say m, is selected (it may be the same cell as the previous leader), it spawns a cell as its child with the numbosome whose value is the greatest odd integer less than or equal to $(m+1)/2$. We call the child *leader bonus*. If there is more than one cell of the same maximum numbosome value, however, the leader does not change for the next period, and there is no leader bonus.

The following illustrates the growth and death of a numoeba starting from a single cell seed with the numbosome value 15, which plays both roles of the leader and a leaf at the start. In the figure, a cell is nicknamed with its numbosome value. Note that the order of the children of a parent is irrelevant.

clock	structure			comments
0	$\boxed{15}$			15 plays both roles of the leader and a leaf.
1	$\boxed{23}$	$\rightarrow$	13	23 (from 15) is the new leader (again).
	$\downarrow$			13 is the leaf bonus,
	11			and 11 is the leader bonus.
2	17			17 is the leader bonus.
	$\uparrow$			
	$\boxed{35}$	$\rightarrow$	5	35 (from 23) is the new leader (again).
	$\downarrow$			5 is from 13.
	17			
	$\downarrow$			17 (from 11) spawns
	9			the leaf bonus 9.
3	13			13 is from 17.
	$\uparrow$			
	$\boxed{53}$	$\rightarrow$	27	53 (from 35) is the new leader (again).
	$\downarrow$			27 is the leader bonus.
	13			Note that 5 dies out.
	$\downarrow$			
	7			



The numoeba continues changing its structure, and at clock 104, it looks as follows.



Here, two ambitious 2429's could not become the leader. The leader 5 will die without promoting these *talented* cells at the next clock. This alludes the fragility of a big organization.

And, the numoeba dies at clock 105.

Your job is to write a program that outputs statistics about the life of numoebae that start from a single cell seed at clock zero.

Input

A sequence of odd integers, each in a line. Each odd integer k_i ($3 \leq k_i \leq 9,999$) indicates the initial numbosome value of the starting cell. This sequence is terminated by a zero.

Output

A sequence of pairs of integers: an integer that represents the numoeba's life time and an integer that represents the maximum number of constituent cells in its life. These two integers should be separated by a space character, and each pair should be followed immediately by a newline. Here, the lifetime means the clock when the numoeba dies.

You can use the fact that the life time is less than 500, and that the number of cells does not exceed 500 in any time, for any seed value given in the input. You might guess that the program would consume a lot of memory. It is true in general. But, don't mind. Referees will use a test data set consisting of no more than 10 starting values, and, starting from any of the those values, the total numbers of cells spawned during the lifetime will not exceed 5000.

Sample Input

```
3
5
7
15
655
2711
6395
7195
8465
0
```

Output for the Sample Input

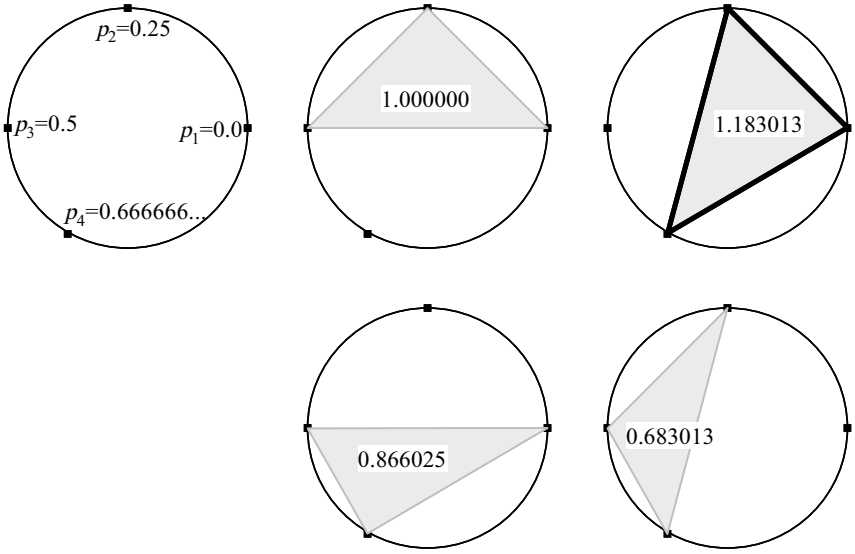
```
2 3
1 1
9 11
105 65
398 332
415 332
430 332
428 332
190 421
```

Problem G
Telescope
Input: scope.txt

Dr. Extreme experimentally made an extremely precise telescope to investigate extremely curious phenomena at an extremely distant place. In order to make the telescope so precise as to investigate phenomena at such an extremely distant place, even quite a small distortion is not allowed. However, he forgot the influence of the internal gas affected by low-frequency vibration of magnetic flux passing through the telescope. The cylinder of the telescope is not affected by the low-frequency vibration, but the internal gas is.

The cross section of the telescope forms a perfect circle. If he forms a coil by putting extremely thin wire along the (inner) circumference, he can measure (the average vertical component of) the temporal variation of magnetic flux: such measurement would be useful to estimate the influence. But points on the circumference at which the wire can be fixed are limited; furthermore, the number of special clips to fix the wire is also limited. To obtain the highest sensitivity, he wishes to form a coil of a polygon shape with the largest area by stringing the wire among carefully selected points on the circumference.

Your job is to write a program which reports the maximum area of all possible m-polygons (polygons with exactly m vertices) each of whose vertices is one of the n points given on a circumference with a radius of 1. An example of the case $n = 4$ and $m = 3$ is illustrated below.



In the figure above, the equations such as "$p_1 = 0.0$" indicate the locations of the n given points, and the decimals such as "1.000000" on m-polygons indicate the areas of m-polygons.

Parameter p_i denotes the location of the i-th given point on the circumference ($1 \leq i \leq n$). The location p of a point on the circumference is in the range $0 \leq p < 1$, corresponding to the range of rotation angles from 0 to 2π radians. That is, the rotation angle of a point at p to the point at 0 equals $2\pi p$ radians. (π is the circular constant 3.14159265358979323846....)

You may rely on the fact that the area of an isosceles triangle ABC (AB = AC = 1) with an interior angle BAC of α radians ($0 < \alpha < \pi$) is $\frac{1}{2}\sin\alpha$, and the area of a polygon inside a circle with a radius of 1 is less than π.

Input

The input consists of multiple subproblems followed by a line containing two zeros that indicates the end of the input. Each subproblem is given in the following format.

$n\ m$
$p_1\ p_2\ \cdots\ p_n$

n is the number of points on the circumference ($3 \leq n \leq 40$). m is the number of vertices to form m-polygons ($3 \leq m \leq n$). The locations of n points, p_1, p_2, ..., p_n, are given as decimals and they are separated by either a space character or a newline. In addition, you may assume that $0 \leq p_1 < p_2 < \cdots < p_n < 1$.

Output

For each subproblem, the maximum area should be output, each in a separate line. Each value in the output may not have an error greater than 0.000001 and its fractional part should be represented by 6 decimal digits after the decimal point.

Sample Input

```
4 3
0.0 0.25 0.5 0.66666666666666666667
4 4
0.0 0.25 0.5 0.75
30 15
0.00 0.03 0.06 0.09 0.12 0.15 0.18 0.21 0.24 0.27
0.30 0.33 0.36 0.39 0.42 0.45 0.48 0.51 0.54 0.57
0.61 0.64 0.66 0.69 0.72 0.75 0.78 0.81 0.84 0.87
40 20
0.351 0.353 0.355 0.357 0.359 0.361 0.363 0.365 0.367 0.369
0.371 0.373 0.375 0.377 0.379 0.381 0.383 0.385 0.387 0.389
0.611 0.613 0.615 0.617 0.619 0.621 0.623 0.625 0.627 0.629
0.631 0.633 0.635 0.637 0.639 0.641 0.643 0.645 0.647 0.649
0 0
```

Output for the Sample Input

```
1.183013
2.000000
3.026998
0.253581
```

Problem H
Ether Geometry
Input: ether.txt

Euclid, dwelt in Alexandria, still being convinced firmly that there is no Royal Road to Euclidean Geometry yet, began to feel gradually, reading newspaper articles, watching TV programs, or brainwashed by young students, that it might be possible to create such a discipline as e-Geometry or Geometry on the Internet. In an e-Geometry environment, he could timely discuss geometry issues with many friends of his including Pythagoras, or he could upload new versions of his unparalleled famous archive "Elements of Geometry (Stoicheia)" on his WWW site. He instantly made up his mind to join the Internet.

He started laying ether cable himself from the hub to his terminal in his uniquely shaped study. He designed the cable route as short as possible.

Figure 1 is a plan of his study. Integers in a pair of parentheses like `(0 0)` are x- and y-coordinates of the nearest pillar or hub or terminal. Between two adjacent pillars are walls indicated by the solid lines. Suppose A is the position of the hub and B his terminal, then the shortest cable may run as drawn with a broken line.

Your mission is to answer the sequence of the coordinates of turning points of the shortest cable route including those of both ends A and B for each plan. So, you are to answer as:
`(1 1) (2 2) (8 4) (9 5)`
for the example of Figure 1.

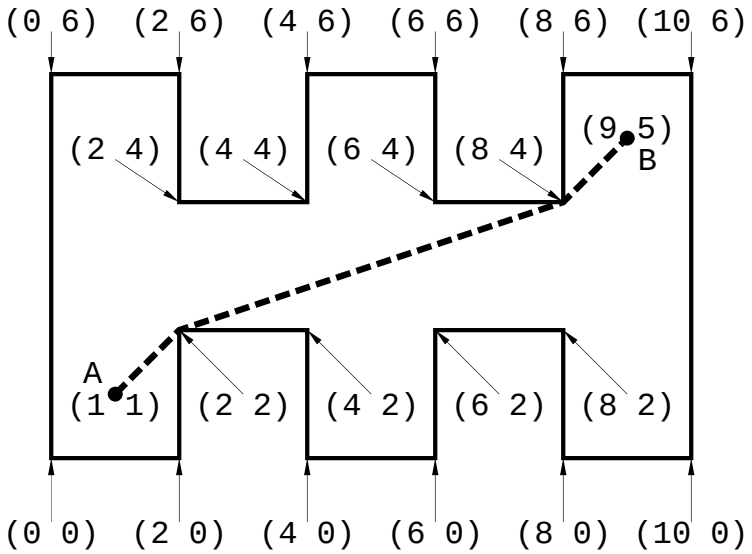


Figure 1

Remember, however, that this is a house of Euclid, the founder and master of Euclidean Geometry. The pillars, hub and terminal are ideal points of Euclidean Geometry. The walls and cable are line segments in the sense of geometry. Accordingly, the cable may overlap the walls or the pillars. Hence for Figures 2, 3 and 4, the answers should be as follows: (Hereafter, the layouts of the study are given by means of the lists of the pillar coordinates.)

Figure 2: Pillars:
`(0 0) (2 0) (2 2) (4 2) (4 0) (6 0) (6 6) (4 6) (4 4) (2 4) (2 6) (0 6)`
Hub: `(1 1)`, Terminal: `(5 1)`
The answer: `(1 1) (2 2) (4 2) (5 1)`

Figure 3: Pillars:
`(0 0) (2 0) (2 2) (4 2) (4 0) (6 0) (6 6) (4 6) (4 4) (2 4) (2 6) (0 6)`
Hub: `(1 1)`, Terminal: `(5 5)`
The answer: `(1 1) (5 5)`
Warning: `(2 2) (4 4)` are not the turning points.

Figure 4: Pillars:
`(0 0) (2 0) (2 2) (4 2) (4 0) (6 0) (6 2) (8 2) (8 0) (10 0) (10 6) (8 6)`
`(8 4) (6 4) (6 6) (4 6) (4 4) (2 4) (2 6) (0 6)`
Hub: `(1 1)`, Terminal: `(9 1)`
The answer: `(1 1) (2 2) (8 2) (9 1)`

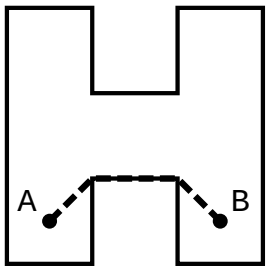


Figure 2

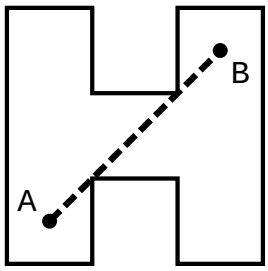


Figure 3

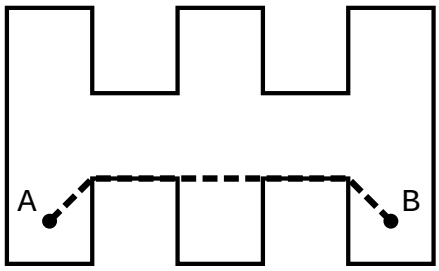


Figure 4

All values of coordinates are integers. Walls are parallel to either x- or y-axis. Walls turn right or left at each pillar for which the coordinates are given. The coordinates of the pillars are given in the order to look the inside of study to the left. From the last pillar in the list, the next wall stretches to the first pillar. Every position of the pillar appears only once in the list. Each pillar is connected to exactly two walls, one of which is parallel to x-axis and the other to y-axis. The whole wall surrounds the study with single stroke closing line. No walls cross. No wall branches exist. It is known that no independent walls or other obstacles are found inside the study. Only one hub and one terminal exist inside the study for each plan, and their positions are always different.

Submit your program when it could solve for the study of complicated shape like Figure 5. This layout is included in the sample input.

Pillars:
(0 0) (2 0) (2 4) (4 4) (4 0) (14 0) (14 6) (10 6) (10 8) (14 8) (14 14)
(4 14) (4 10) (2 10) (2 14) (0 14) (0 8) (6 8) (6 12) (12 12) (12 10)
(8 10) (8 4) (12 4) (12 2) (6 2) (6 6) (0 6)
Hub: (1 1), Terminal: (1 13)
The answer:
(1 1) (2 4) (4 4) (6 2) (12 2) (12 4) (10 6) (10 8) (12 10) (12 12)
(6 12) (4 10) (2 10) (1 13)

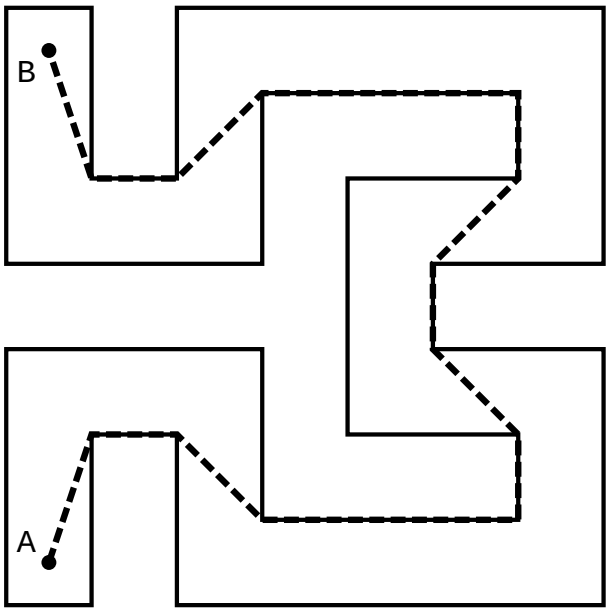


Figure 5

Input

The input contains one or more data sets of the coordinates of the pillars, hubs and terminals. Each set begins with a line of an integer n ($4 \leq n \leq 208$), which is the number of the pillars of the study. The following lines contain the coordinates of pillars, five pairs of coordinates per line. After these lines comes one line that contains the coordinates of the hub $(a_x\ a_y)$ and the terminal $(b_x\ b_y)$.

The structure of one complete data set is as follows: (Input/output data do not use parentheses.)

n
x_1 y_1 x_2 y_2 ... x_5 y_5
x_6 y_6 x_7 y_7 ... x_{10} y_{10}
x_{11} y_{11} ...
... x_n y_n
a_x a_y b_x b_y

To make the situation simple, the coordinates of the pillars are all even integers, while those of

the hub and terminal are all odd. It is also known that the room size is less than 100 coordinate units in both x and y directions and the cable length is shorter than 1000 coordinate units.

The input is terminated by a line having one 0 after the last set.

Output

For each data set, print the sequence of the coordinates of turning points of the shortest cable with those of the hub at the beginning and those of the terminal at the end. Each line should contain five pairs of coordinates separated by a single space, except the last of these lines which may contain fewer than five. In each coordinate pair, x and y values should be separated also by a single space. The last coordinate of each line may be followed by a single space. The answer of each data set must be followed by a single empty line.

Sample Input

The sample input corresponds to Figures 1 and 5.

```
20
 0  0  2  0  2  2  4  2  4  0
 6  0  6  2  8  2  8  0 10  0
10  6  8  6  8  4  6  4  6  6
 4  6  4  4  2  4  2  6  0  6
 1  1  9  5
28
 0  0  2  0  2  4  4  4  4  0
14  0 14  6 10  6 10  8 14  8
14 14  4 14  4 10  2 10  2 14
 0 14  0  8  6  8  6 12 12 12
12 10  8 10  8  4 12  4 12  2
 6  2  6  6  0  6
 1  1  1 13
0
```

Output for the Sample Input

```
1 1 2 2 8 4 9 5

1 1 2 4 4 4 6 2 12 2
12 4 10 6 10 8 12 10 12 12
6 12 4 10 2 10 1 13
```