14th Denison Spring Programming Contest
5 April, 2003

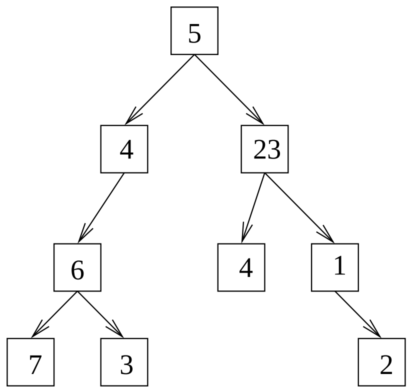
Rules:

1. All questions require you to read the test data from standard input and write results to standard output. You cannot use files for input or output.
2. All programs will be re-compiled prior to testing with the judges' data.
3. Non-standard libraries cannot be used in your solutions. The Standard Template Library (STL) and C++ string libraries are allowed.
4. Programming style is not considered in this contest. You are free to code in whatever style you prefer. Documentation is not required.
5. All communication with the judges will be handled by the PC^2 environment.
6. The allowed programming languages are C, C++ and Java.
7. Judges' decisions are to be considered final. No cheating will be tolerated.
8. There are **five** questions to be completed in **four hours**.

Problem A: Tree Summing

Binary trees are fundamental in computer science. This problem involves building and trvaersing trees.

Given a binary tree (inputted in the manner described below), you are to determine if there exists a root-to-leaf path that sums to a specified integer. For example, the sum of each of the root-to-leaf paths in the tree below are 22, 18, 32 and 31.



In this problem, the binary tree will be given by a sequence of pairs (n, s) where n is the positive integer giving the value of a node and s is a string indicating the path from the root to that node. The string s will be a string of `L`'s and `R`'s where `L` indicates a left branch and `R` indicates a right branch. The root node will have the empty string s and the information for a binary tree will terminate with `()`. The nodes may be given in any order.

The binary tree in the example above could be given by:

```
(4,L)(1,RR)(23,R)
(5,)(4,RL)
(6,LL)(7,LLL)(3,LLR)(2,RRR)()
```

Input

There will be multiple problem instances for input. Each problem instance will start on a new line and will consist of an integer *target* followed by a space, followed by a description of a tree. The tree description will be in the form above, will contain no more than 18 nodes and may be on more than one line. A *target* value of 0 indicates end of input.

Output

For each problem instance output one line of output, either:

`There is a path of sum` *target*.

or

`There is no path of sum` *target*.

accordingly. (You fill in the given value for *target*, of course.)

Sample Input

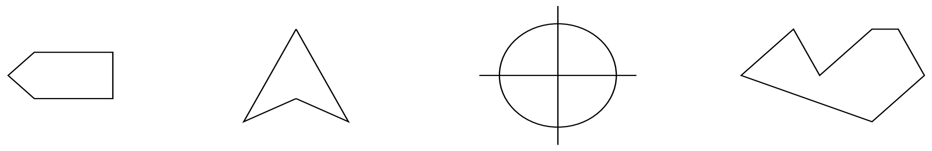
```
22 (4,L)(1,RR)(23,R)
(5,)(4,RL)
(6,LL)(7,LLL)(3,LLR)(2,RRR)()
6 (1,)(3,L)(5,LL)()
0
```

Sample Output

```
There is a path of sum 22.
There is no path of sum 6.
```

Problem B: Symmetry

Ida makes patterns for designs given by customers. Material is expensive, so Ida wants to take advantage of symmetries in the designs and not have to cut a whole pattern if possible. There are two symmetries Ida wishes to take advantage of: horizontal and vertical. In the four figures below, the first figure is vertically symmetric, the second is horizontally symmetric, the third is both horizontally and vertically symmetric and the last figure is neither vertically nor horizontally symmetric.



Ida receives from her customers a two-color scan of their patterns. The pixels are black or white. The black pixels represent the pattern. There may be extra white rows or columns surrounding the pattern, which should be ignored as far as the pattern goes. You are to determine whether the pattern is symmetric horizontally, vertically or both.

Input

Input will consist of one or more data sets. The first line of each data set will contain the number of columns c ($c \leq 20$) and the number of rows r ($r \leq 20$) in the scan. There will be r lines that follow, each line will consist of c characters, each either `b` or `w`, representing a black or white pixel. There will be at least one black pixel in each data set.

The last line of input will have $c = r = 0$ and should not be processed.

Output

Each data set should generate a line like one of the following:

`Pattern` n `is asymmetrical.`

`Pattern` n `is horizontally symmetrical.`

`Pattern` n `is vertically symmetrical.`

`Pattern` n `is both horizontally and vertically symmetrical.`

where n is the number of the data set (starting at 1).

Sample Input

```
10 4
wwwwbbbbbw
wwwwbwwwbw
wwwwbwbwbw
wwwwbwbwbw
5 3
bbwbw
bwbww
wbbwb
7 6
wwwwwww
wwwwbbw
wwwbbbb
wwwwbbw
wwwwwww
wwwwwww
0 0
```

Sample Output

```
Pattern 1 is horizontally symmetrical.
Pattern 2 is asymmetrical.
Pattern 3 is both horizontally and vertically symmetrical.
```

Problem C: Speeders

The Ohio Toll Roads Commission has decided to fine drivers speeding on the toll roads. This should be fairly easy since when entering a toll road, each driver has a ticket with the entry point marked by time and location. When they get off the toll road, it should be a simple calculation to see if their average speed exceeded the speed limit. They've decided to access a fine based on how many miles per hour the average speed of a car exceeds the speed limit. To avoid bickering, all speeds will be truncated to integer values. (So, a speed of 79.8 MPH would be considered a speed of 79 MPH, for purposes of fining.) Thus a driver entering the road at mile marker 300 at 3:05 AM and gets off at mile marker 149 at 5:06 AM has an average speed of 74 MPH (when truncated). If the speed limit is 70 MPH with a fine of $5 per MPH over the speed limit, then the driver would have a fine of $20.

They've asked you to write a program to compute these fines.

Input

There will be multiple data sets for this problem. Each data set will consist of information about a particular toll road, followed by information about one or more vehicles traveling the road.

Each data set starts with a line with two integers, *speedlimit* and *fine*, giving the speed limit and the fine (in dollars) to be assessed for each mile per hour a driver exceeds the speed limit. If *speedlimit* is 0, end of input is indicated.

Following the first line will be multiple lines containing four integers: *sm*, *st*, *em*, and *et*, indicating starting mile marker, starting time, ending mile marker, and ending time, respectively. Time will be given on a 24-hour clock, so 730 is 7:30 AM and 1930 is 7:30 PM. You may assume no one trip will be longer than a day nor will it span two calendar days. The list of vehicle information will end with a line of four 0's.

Output

For each data set print, in the format shown below in the sample, the number of the data set (preceded by the word `Road`), and the total amount of the fine, if any, for each vehicle on the road. The vehicles should be numbered, as shown, and information for a vehicle should be on its own line. Note the output is indented 2 spaces for each vehicle. If there is no fine, the appropriate line, as shown, should be printed. A blank line should follow each data set.

Sample Input

```
75 10
300 305 149 505
250 200 42 430
0 0 0 0
55 45
0 1410 40 1540
0 100 120 300
120 300 0 500
0 737 324 1214
0 0 0 0
0 50
```

Sample Output

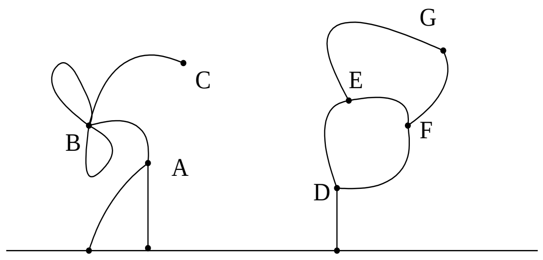
```
Road 1
  Vehicle 1: no fine
  Vehicle 2: $80

Road 2
  Vehicle 1: no fine
  Vehicle 2: $225
  Vehicle 3: $225
  Vehicle 4: $675
```

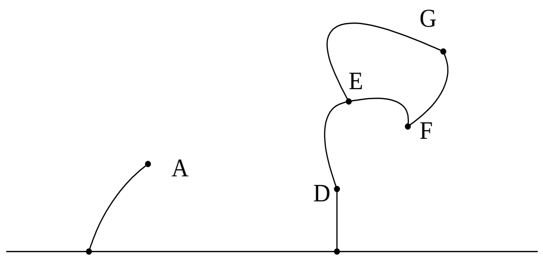
Problem D: Hakenbush

The game of Hakenbush was invented by John Conway and goes as follows. You start with a collection of "stick figures" drawn where every figure is touching the "ground" at at least one point. In the drawing below, there are two figures. The left one is connected to the ground at two points and the right figure is connected at one. The players alternate erasing edges. If the result of erasing an edge causes a portion of the remaining figure to be disconnected from the ground, that disconnected portion is eliminated too. Play continues, with the last player to erase an edge losing. Here, we won't play a game but are only concerned with the figures that remain after erasure.

In our version we'll allow any number of edges to be drawn from a point to itself (as in the B-to-B edges in the drawing below) or between points and no edges to be drawn connecting ground to ground. There are 12 edges in the drawing below. ("Ground" is permanent and doesn't count as an edge.) Many edges may be drawn from a point to ground. (For example, there are two edges from A to ground, here.)



For example, if edges A-to-B, A-to-ground, and D-to-F are erased, the result is the figure below.



Input

There will be multiple data sets form this problem. Each data set will consist of two input lines. The first integer n ($n \leq 20$) on line one will be the number of edges to be drawn. There will then follow n pairs of uppercase letters on this first input line, indicating the endpoints of each drawn edge. We'll use the letter `Z` to indicate the ground. A value of $n = 0$ will indicate the end of input. The second line of input will consist of m followed by m pairs of uppercase letters indicating edges to erase. There may be illegal directions to erase edges — that is, directions to erase egdes that have in fact not been drawn or directions to erase edges in portions that have already been eliminated.

Output

Each data set should generate one line of input which will consist of one integer, giving the number of edges remaining in the drawing, if all the directions to erase edges were legal. If a direction to erase an edge is illegal, print the line

```
Illegal move.
```

and ignore the remaining part of that data set.

Sample Input

```
12 A Z B B A B E F Z A B B C B D Z D E D F G E F G
3 B A D F A Z
4 Z B B Z B D D D
3 B D B B B Z
0
```

Sample Output

```
6
Illegal move.
```

Problem E: Voting

There are many schemes used when trying to pick one winner from a slate of many candidates. Many of the schemes have bad effects. Here's one that works well. It's a (potentially) two-phased method. Suppose there are n candidates. Each voter ranks the n candidates from best to worst. Note that a ballot could be thought of as how a person would vote on each pair of candidates: a voter prefers candidate A to candidate B if A is ranked before B on the ballot.

Phase one of the voting is to tally each pairwise vote. That is, for each pair of candidates, find the results of the election as if just those two candidates were involved, ignoring the other candidates. (Note if there are n candidates, there will be $n(n-1)/2$ such pairwise elections.) If there is one candidate who beats every other candidate on the pairwise voting, then that candidate is declared the winner.

If there is no such candidate, we go to Phase two: The candidates are given points for where they appear on each ballot, with n points for being ranked first, $n-1$ points for being ranked second, and so on until 1 point for being ranked last. The points are tallied for each candidate and the highest total is declared the winner.

For example, suppose there are 4 candidates, A, B, C, and D, and 5 voters whose ballots are: ABCD, CBDA, ACDB, DCBA, and BCDA. (Here, the most preferred candidates are listed first.) In the pairwise phase, C beats A, B, and D and so is the winner. If the 4th ballot were changed to DACB, then A beats B and C, B beats D, C beats B and D, and D beats A. Thus there is no pairwise winner and we proceed to phase two. The counts would be A:13, B:12, C:14 and D:11. Thus C would be declared the winner. Note there may be a tie in phase two.

Input

There will be multiple data sets, each data set will start with a line containing the integers n and m, where n ($3 \leq n \leq 8$) is the number of candidates (indicated by the first n uppercase letters) and m ($m \leq 100$) is the number of ballots. m lines follow, each containing a string of n uppercase characters (no spaces) indicating a ballot from highest preference to lowest. A value of $n = 0$ indicates end of input.

Output

For each data set you will output one line of output of the form:

`Election` e`:  `X` won the pairwise voting.`

`Election` e`:  `X` won the voting with `p` points.`

or

`Election` e`:  there is a tie among `c` candidates with `p` points.`

according to the outcome. (You fill in the appropriate values for e, X, c and p, of course.) Here, e is the number of the data set, starting at 1.

Sample Input

```
4 5
ABCD
CBDA
ACDB
DCBA
BCDA
4 5
ABCD
CBDA
ACDB
DACB
BCDA
3 3
ABC
BCA
CAB
0 0
```

Sample Output

```
Election 1: C won the pairwise voting.
Election 2: C won the voting with 14 points.
Election 3: there is a tie among 3 candidates with 6 points.
```