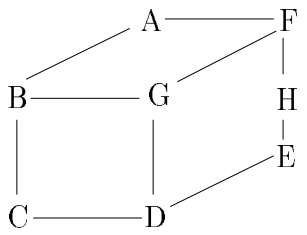
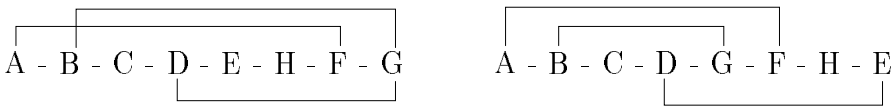
Problem A: Bandwidth

Given a graph (V,E) where V is a set of nodes and E is a set of arcs in VxV, and an *ordering* on the elements in V, then the *bandwidth* of a node v is defined as the maximum distance in the ordering between v and any node to which it is connected in the graph. The bandwidth of the ordering is then defined as the maximum of the individual bandwidths. For example, consider the following graph:



This can be ordered in many ways, two of which are illustrated below:



For these orderings, the bandwidths of the nodes (in order) are 6, 6, 1, 4, 1, 1, 6, 6 giving an ordering bandwidth of 6, and 5, 3, 1, 4, 3, 5, 1, 4 giving an ordering bandwidth of 5.

Write a program that will find the ordering of a graph that minimises the bandwidth.

Input will consist of a series of graphs. Each graph will appear on a line by itself. The entire file will be terminated by a line consisting of a single #. For each graph, the input will consist of a series of records separated by ';'. Each record will consist of a node name (a single upper case character in the the range 'A' to 'Z'), followed by a ':' and at least one of its neighbours. The graph will contain no more than 8 nodes.

Output will consist of one line for each graph, listing the ordering of the nodes followed by an arrow (->) and the bandwidth for that ordering. All items must be separated from their neighbours by exactly one space. If more than one ordering produces the same bandwidth, then choose the smallest in lexicographic ordering, that is the one that would appear first in an alphabetic listing.

Sample input

```
A:FB;B:GC;D:GC;F:AGH;E:HD
#
```

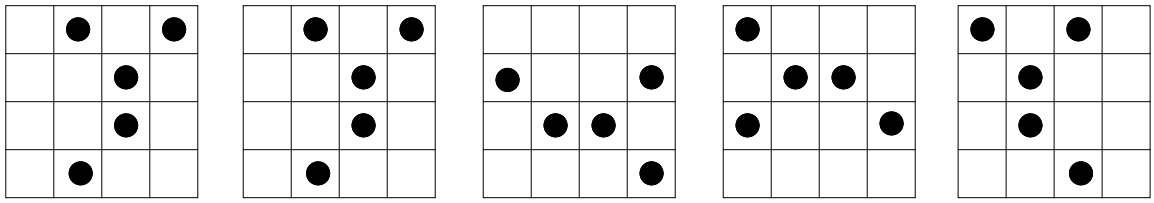
Sample output

```
A B C F G D H E -> 3
```

Problem B: The Spot Game

The game of Spot is played on an NxN board as shown below for N = 4. During the game, alternate players may either place a black counter (spot) in an empty square or remove one from the board, thus producing a variety of patterns. If a board pattern (or its rotation by 90 degrees or 180 degrees) is repeated during a game, the player producing that pattern loses and the other player wins. The game terminates in a draw after 2N moves if no duplicate pattern is produced before then.

Consider the following patterns:



If the first pattern had been produced earlier, then any of the following three patterns (plus one other not shown) would terminate the game, whereas the last one would not.

Input will consist of a series of games, each consisting of the size of the board, N ($2 \leq N \leq 50$) followed, on separate lines, by 2N moves, whether they are all necessary or not. Each move will consist of the coordinates of a square (integers in the range 1..N) followed by a blank and a character '+' or '-' indicating the addition or removal of a spot respectively. You may assume that all moves are legal, that is there will never be an attempt to place a spot on an occupied square, nor to remove a non-existent spot. Input will be terminated by a zero (0).

Output will consist of one line for each game indicating which player won and on which move, or that the game ended in a draw.

Sample input

```
2
1 1 +
2 2 +
2 2 -
1 2 +
2
1 1 +
2 2 +
1 2 +
2 2 -
0
```

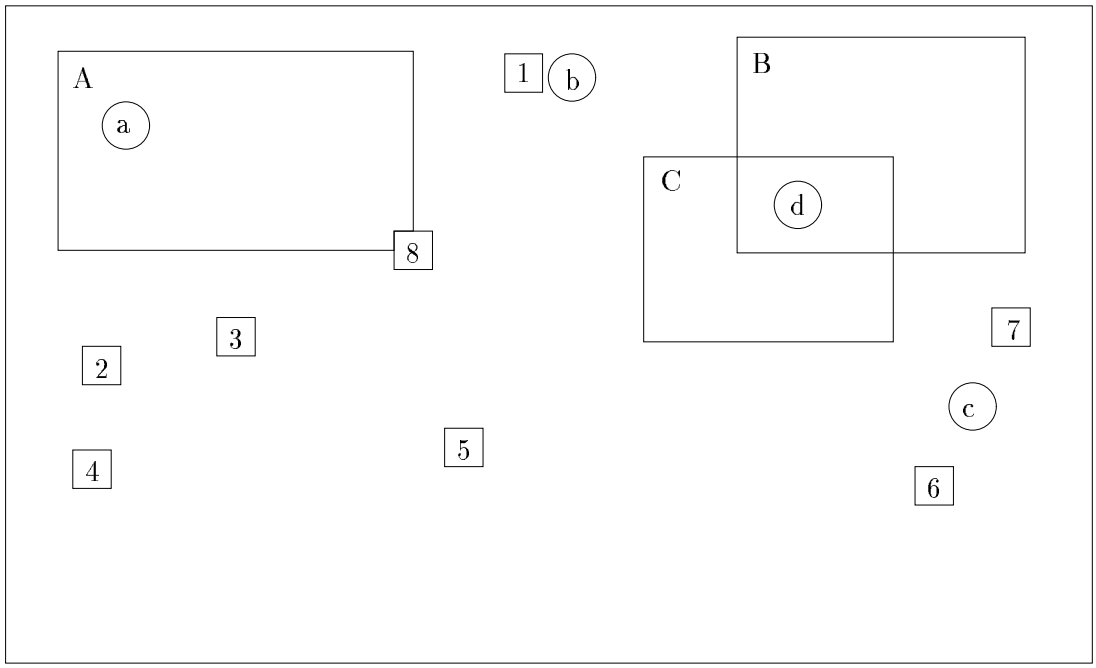
Sample output

```
Player 2 wins on move 3
Draw
```

Problem C: Mouse Clicks

A typical windowing system on a computer will provide a number of icons on the screen as well as some defined regions. When the mouse button is clicked, the system has to determine where the cursor is and what is being selected. For this problem we assume that a mouse click in (or on the border of) a region selects that region, otherwise it selects the closest visible icon (or icons in the case of a tie).

Consider the following screen:



A mouse click at 'a' will select region A. A mouse click at 'b' will select icon 1. A mouse click at 'c' will select icons 6 and 7. A mouse click at 'd' is ambiguous. The ambiguity is resolved by assuming that one region is in front of another. In the data files, later regions can be assumed to be in front of earlier regions. Since regions are labelled in order of appearance (see later) 'd' will select C. Note that regions always overlap icons so that obscured icons need not be considered and that the origin (0,0) is at the top left corner.

Write a program that will read in a series of region and icon definitions followed by a series of mouse clicks and return the selected items. Coordinates will be given as pairs of integers in the range 0..499 and you can assume that all icons and regions lie wholly within the screen. Your program must number all icons (even invisible ones) in the order of arrival starting from 1 and label regions alphabetically in the order of arrival starting from 'A'.

Input will consist of a series of lines. Each line will identify the type of data: I for icon, R for region and M for mouse click. There will be no separation between the specification part and the event part, however no icon or region specifications will follow the first mouse click. An I will be followed by the coordinates of the centre of the icon, R will be followed by the coordinates of the top left and bottom right corners respectively and M will be followed by the coordinates of the cursor at the time of the click. There will always be at least one visible icon and never more than 25 regions and 50 icons. The entire file will be terminated by a line consisting of a single #.

Output will consist of one line for each mouse click, containing the selection(s) for that click. Regions will be identified by their single character identifier, icon numbers will be written out right justified in a field of width 3, and where there is more than one icon number they will appear in increasing numerical order.

Sample input

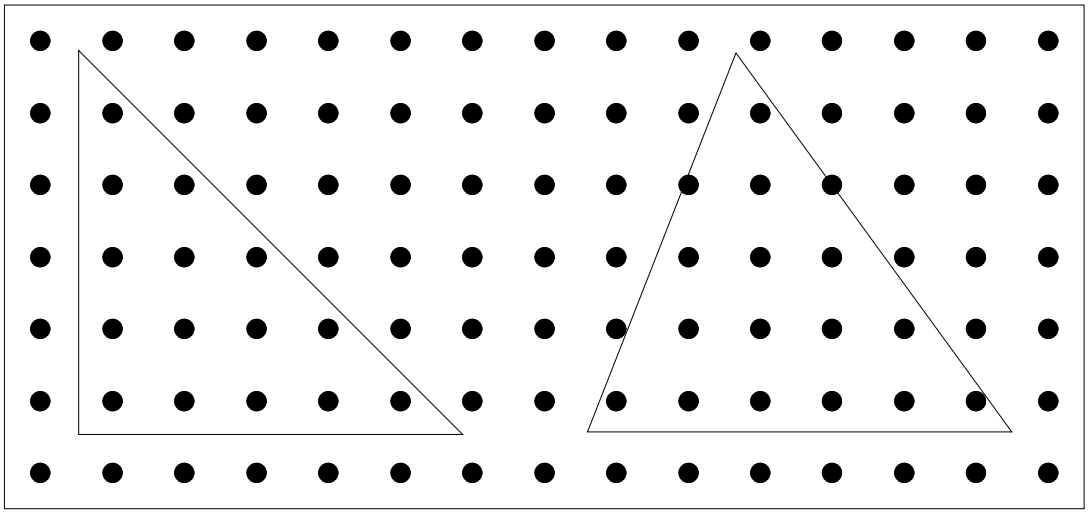
```
I 216 28
R 22 19 170 102
I 40 150
I 96 138
I 36 193
R 305 13 425 103
I 191 184
I 387 200
R 266 63 370 140
I 419 134
I 170 102
M 50 50
M 236 30
M 403 167
M 330 83
#
```

Sample output

```
A
  1
  6  7
C
```

Problem D: Orchard Trees

An Orchardist has planted an orchard in a rectangle with trees uniformly spaced in both directions. Thus the trees form a rectangular grid and we can consider the trees to have integer coordinates. The origin of the coordinate system is at the bottom left of the following diagram:



Consider that we now overlay a series of triangles on to this grid. The vertices of the triangle can have any real coordinates in the range 0.0 to 100.0, thus trees can have coordinates in the range 1 to 99. Two possible triangles are shown.

Write a program that will determine how many trees are contained within a given triangle. For the purposes of this problem, you may assume that the trees are of point size, and that any tree (point) lying exactly on the border of a triangle is considered to be in the triangle.

Input will consist of a series of lines. Each line will contain 6 real numbers in the range 0.00 to 100.00 representing the coordinates of a triangle. The entire file will be terminated by a line containing 6 zeroes (0 0 0 0 0 0).

Output will consist of one line for each triangle, containing the number of trees for that triangle right justified in a field of width 4.

Sample input

```
1.5 1.5  1.5 6.8  6.8 1.5
10.7 6.9  8.5 1.5  14.5 1.5
0 0 0 0 0 0
```

Sample output

```
  15
  17
```

Problem E: Student Grants

The Government of Impecunia has decided to discourage tertiary students by making the payments of tertiary grants a long and time-consuming process. Each student is issued a student ID card which has a magnetically encoded strip on the back which records the payment of the student grant. This is initially set to zero. The grant has been set at $40 per year and is paid to the student on the working day nearest to his birthday. (Impecunian society is still somewhat medieval and only males continue with tertiary education.) Thus on any given working day up to 25 students will appear at the nearest office of the Department of Student Subsidies to collect their grant.

The grant is paid by an Automatic Teller Machine which is driven by a reprogrammed $8085\frac{1}{2}$ chip originally designed to run the state slot machine. The ATM was built in the State Workshops and is designed to be difficult to rob. It consists of an interior vault where it holds a large stock of $1 coins and an output store from which these coins are dispensed. To limit possible losses it will only move coins from the vault to the output store when that is empty. When the machine is switched on in the morning, with an empty output store, it immediately moves 1 coin into the output store. When that has been dispensed it will then move 2 coins, then 3, and so on until it reaches some preset limit k. It then recycles back to 1, then 2 and so on.

The students form a queue at this machine and, in turn, each student inserts his card. The machine dispenses what it has in its output store and updates the amount paid to that student by writing the new total on the card. If the student has not received his full grant, he removes his card and rejoins the queue at the end. If the amount in the store plus what the student has already received comes to more than $40, the machine only pays out enough to make the total up to $40. Since this fact is recorded on the card, it is pointless for the student to continue queuing and he leaves. The amount remaining in the store is then available for the next student.

Write a program that will read in values of N (the number of students, $1 \leq N \leq 25$) and k (the limit for that machine, $1 \leq k \leq 40$) and calculate the order in which the students leave the queue.

Input will consist of a series of lines each containing a value for N and k as integers. The list will be terminated by two zeroes (0 0).

Output will consist of a line for each line of input and will contain the list of students in the order in which they leave the queue. Students are ordered according to their position in the queue at the start of the day. All numbers must be right justified in a field of width 3.

Sample input

```
5 3
0 0
```

Sample output

```
  1  3  5  2  4
```

Problem F: Gondwanaland Telecom

Gondwanaland Telecom makes charges for calls according to distance and time of day. The basis of the charging is contained in the following schedule, where the charging step is related to the distance:

Charging Step (distance)	Day Rate 8am to 6pm	Evening Rate 6pm to 10pm	Night Rate 10pm to 8am
A	0.10	0.06	0.02
B	0.25	0.15	0.05
C	0.53	0.33	0.13
D	0.87	0.47	0.17
E	1.44	0.80	0.30

All charges are in dollars per minute of the call. Calls which straddle a rate boundary are charged according to the time spent in each section. Thus a call starting at 5:58 pm and terminating at 6:04 pm will be charged for 2 minutes at the day rate and for 4 minutes at the evening rate. Calls less than a minute are not recorded and no call may last more than 24 hours.

Write a program that reads call details and calculates the corresponding charges.

Input lines will consist of the charging step (upper case letter 'A'..'E'), the number called (a string of 7 digits and a hyphen in the approved format) and the start and end times of the call, all separated by exactly one blank. Times are recorded as hours and minutes in the 24 hour clock, separated by one blank and with two digits for each number. Input will be terminated by a line consisting of a single #.

Output will consist of the called number, the time in minutes the call spent in each of the charge categories, the charging step and the total cost in the format shown below.

Sample input

```
A 183-5724 17 58 18 04
#
```

Sample output

```
          10   16   22   28   31       39
 183-5724   2    4    0    A   0.44
```

Problem G: ID Codes

It is 2084 and the year of Big Brother has finally arrived, albeit a century late. In order to exercise greater control over its citizens and thereby to counter a chronic breakdown in law and order, the Government decides on a radical measure—all citizens are to have a tiny microcomputer surgically implanted in their left wrists. This computer will contains all sorts of personal information as well as a transmitter which will allow people's movements to be logged and monitored by a central computer. (A desirable side effect of this process is that it will shorten the dole queue for plastic surgeons.)

An essential component of each computer will be a unique identification code, consisting of up to 50 characters drawn from the 26 lower case letters. The set of characters for any given code is chosen somewhat haphazardly. The complicated way in which the code is imprinted into the chip makes it much easier for the manufacturer to produce codes which are rearrangements of other codes than to produce new codes with a different selection of letters. Thus, once a set of letters has been chosen all possible codes derivable from it are used before changing the set.

For example, suppose it is decided that a code will contain exactly 3 occurrences of 'a', 2 of 'b' and 1 of 'c', then three of the allowable 60 codes under these conditions are:

abaabc
abaacb
ababac

These three codes are listed from top to bottom in alphabetic order. Among all codes generated with this set of characters, these codes appear consecutively in this order.

Write a program to assist in the issuing of these identification codes. Your program will accept a sequence of no more than 50 lower case letters (which may contain repeated characters) and print the successor code if one exists or the message 'No Successor' if the given code is the last in the sequence for that set of characters.

Input will consist of a series of lines each containing a string representing a code. The entire file will be terminated by a line consisting of a single #.

Output will consist of one line for each code read containing the successor code or the words 'No Successor'.

Sample input

```
abaacb
cbbaa
#
```

Sample output

```
ababac
No Successor
```

Problem H: Dollars

New Zealand currency consists of $100, $50, $20, $10, and $5 notes and $2, $1, 50c, 20c, 10c and 5c coins. Write a program that will determine, for any given amount, in how many ways that amount may be made up. Changing the order of listing does not increase the count. Thus 20c may be made up in 4 ways: 1x20c, 2x10c, 10c+2x5c, and 4x5c.

Input will consist of a series of real numbers no greater than $50.00 each on a separate line. Each amount will be valid, that is will be a multiple of 5c. The file will be terminated by a line containing zero (0.00).

Output will consist of a line for each of the amounts in the input, each line consisting of the amount of money (with two decimal places and right justified in a field of width 5), followed by the number of ways in which that amount may be made up, right justified in a field of width 12.

Sample input

```
0.20
  2.00
0.00
```

Sample output

```
 0.20           4
 2.00         293
```