10284 Chessboard in FEN

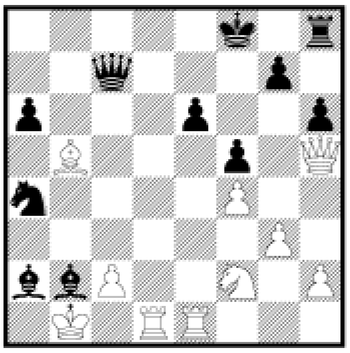
In the **FEN** (Forsyth-Edwards Notation), a chessboard is described as follows:

- The Board-Content is specified starting with the top row and ending with the bottom row.
- Character '/' is used to separate data of adjacent rows.
- Each row is specified from left to right.
- White pieces are identified by uppercase piece letters: `PNBRQK`.
- Black pieces are identified by lowercase piece letters: `pnbrqk`.
- Empty squares are represented by the numbers one through eight.
- A number used represents the count of contiguous empty squares along a row.
- Each row's sum of numbers and characters must equal 8.

As for example:

`5k1r/2q3p1/p3p2p/1B3p1Q/n4P2/6P1/bbP2N1P/1K1RR3`,

is the **FEN** notation description of the following chessboard:



The chessboard of the beginning of a chess game is described in **FEN** as:

`rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR`

Your task is simple: given a chessboard description in a **FEN** notation you are asked to compute the number of unoccupied squares on the board which are not attacked by any piece.

Input

Input is a sequence of lines, each line containing a **FEN** description of a chessboard. Note that the description does not necessarily give a legal chess position. Input lines do not contain white space.

Output

For each line of input, output one line containing an integer which gives the number of unoccupied squares which are not attacked.

Sample Input

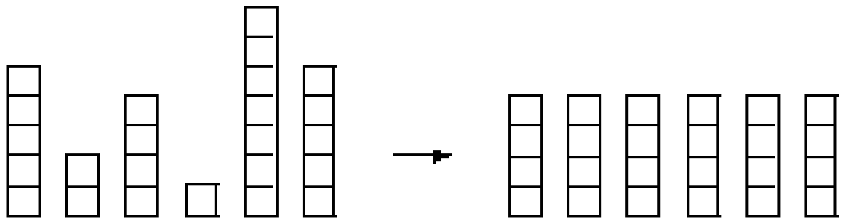
```
5k1r/2q3p1/p3p2p/1B3p1Q/n4P2/6P1/bbP2N1P/1K1RR3
rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR
```

Sample Output

```
3
16
```

591 Box of Bricks

Little Bob likes playing with his box of bricks. He puts the bricks one upon another and builds stacks of different height. “Look, I’ve built a wall!”, he tells his older sister Alice. “Nah, you should make all stacks the same height. Then you would have a real wall.”, she retorts. After a little con- sideration, Bob sees that she is right. So he sets out to rearrange the bricks, one by one, such that all stacks are the same height afterwards. But since Bob is lazy he wants to do this with the minimum number of bricks moved. Can you help?



Input

The input consists of several data sets. Each set begins with a line containing the number n of stacks Bob has built. The next line contains n numbers, the heights h_i of the n stacks. You may assume $1 \le n \le 50$ and $1 \le h_i \le 100$.

The total number of bricks will be divisible by the number of stacks. Thus, it is always possible to rearrange the bricks such that all stacks have the same height.

The input is terminated by a set starting with $n = 0$. This set should not be processed.

Output

For each set, first print the number of the set, as shown in the sample output. Then print the line “`The minimum number of moves is` k.”, where k is the minimum number of bricks that have to be moved in order to make all the stacks the same height.

Output a blank line after each set.

Sample Input

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

Sample Output

```
Set #1
The minimum number of moves is 5.
```

10341 Solve It

Solve the equation:

$$p * e^{-x} + q * \sin(x) + r * \cos(x) + s * \tan(x) + t * x^2 + u = 0$$

where $0 \le x \le 1$.

Input

Input consists of multiple test cases and terminated by an EOF. Each test case consists of 6 integers in a single line: p, q, r, s, t and u (where $0 \le p, r \le 20$ and $-20 \le q, s, t \le 0$). There will be maximum 2100 lines in the input file.

Output

For each set of input, there should be a line containing the value of x, correct up to 4 decimal places, or the string 'No solution', whichever is applicable.

Sample Input

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

Sample Output

```
0.7071
No solution
0.7554
```

10000 Longest Paths

It is a well known fact that some people do not have their social abilities completely enabled. One example is the lack of talent for calculating distances and intervals of time. This causes some people to always choose the longest way to go from one place to another, with the consequence that they are late to whatever appointments they have, including weddings and programming contests. This can be highly annoying for their friends.

César has this kind of problem. When he has to go from one point to another he realizes that he has to visit many people, and thus always chooses the longest path. One of César's friends, Felipe, has understood the nature of the problem. Felipe thinks that with the help of a computer he might be able to calculate the time that César is going to need to arrive to his destination. That way he could spend his time in something more enjoyable than waiting for César.

Your goal is to help Felipe developing a program that computes the length of the longest path that can be constructed in a given graph from a given starting point (César's residence). You can assume that there is at least one path from this starting point to any destination. Also, the graph has no cycles (there is no path from any node to itself), so César will reach his destination in a finite time. In the same line of reasoning, nodes are not considered directly connected to themselves.

Input

The input consists of a number of cases. The first line on each case contains a positive number n ($1 < n \leq 100$) that specifies the number of points that César might visit (i.e., the number of nodes in the graph).

A value of $n = 0$ indicates the end of the input.

After this, a second number s is provided, indicating the starting point in César's journey ($1 \leq s \leq n$). Then, you are given a list of pairs of places p and q, one pair per line, with the places on each line separated by white-space. The pair "p q" indicates that César can visit q after p.

A pair of zeros ('`0 0`') indicates the end of the case.

As mentioned before, you can assume that the graphs provided will not be cyclic and that every place will be reachable from the starting place.

Output

For each test case you have to find the length of the longest path that begins at the starting place. You also have to print the number of the final place of such longest path. If there are several paths of maximum length, print the final place with smallest number.

Print a new line after each test case.

Sample Input

```
2
1
1 2
0 0
5
3
1 2
```

```
3 5
3 1
2 4
4 5
0 0
5
5
5 1
5 2
5 3
5 4
4 1
4 2
0 0
0
```

Sample Output

```
Case 1: The longest path from 1 has length 1, finishing at 2.

Case 2: The longest path from 3 has length 4, finishing at 5.

Case 3: The longest path from 5 has length 2, finishing at 1.
```

12541 Birthdates

Write a program to identify the youngest person and the oldest person in a class.

Input

The number n ($1 \leq n \leq 100$) in the first line determines the number of people in a class. The following n lines contain person's name and his/her birthdate.

The information in each line is of this format:

$personName\ dd\ mm\ yyyy$

where $personName$ is a single word less than 15 letters, $dd\ mm\ yyyy$ are date, month and year of the birthdate.

Suppose that no one has the same name or the same birthdate.

Output

Print out 2 lines containing the name of youngest person and oldest person, respectively.

Sample Input

```
5
Mickey 1 10 1991
Alice 30 12 1990
Tom 15 8 1993
Jerry 18 9 1990
Garfield 20 9 1990
```

Sample Output

```
Tom
Jerry
```

101 The Blocks Problem

Background

Many areas of Computer Science use simple, abstract domains for both analytical and empirical studies. For example, an early AI study of planning and robotics (STRIPS) used a block world in which a robot arm performed tasks involving the manipulation of blocks.

In this problem you will model a simple block world under certain rules and constraints. Rather than determine how to achieve a specified state, you will "program" a robotic arm to respond to a limited set of commands.

The Problem

The problem is to parse a series of commands that instruct a robot arm in how to manipulate blocks that lie on a flat table. Initially there are n blocks on the table (numbered from 0 to $n-1$) with block b_i adjacent to block b_{i+1} for all $0 \leq i < n-1$ as shown in the diagram below:

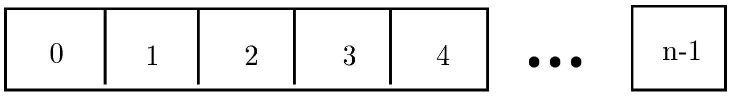


Figure 1: Initial Blocks World

The valid commands for the robot arm that manipulates blocks are:

- move a onto b

  where a and b are block numbers, puts block a onto block b after returning any blocks that are stacked on top of blocks a and b to their initial positions.

- move a over b

  where a and b are block numbers, puts block a onto the top of the stack containing block b, after returning any blocks that are stacked on top of block a to their initial positions.

- pile a onto b

  where a and b are block numbers, moves the pile of blocks consisting of block a, and any blocks that are stacked above block a, onto block b. All blocks on top of block b are moved to their initial positions prior to the pile taking place. The blocks stacked above block a retain their order when moved.

- pile a over b

  where a and b are block numbers, puts the pile of blocks consisting of block a, and any blocks that are stacked above block a, onto the top of the stack containing block b. The blocks stacked above block a retain their original order when moved.

- quit

  terminates manipulations in the block world.

Any command in which $a = b$ or in which a and b are in the same stack of blocks is an illegal command. All illegal commands should be ignored and should have no affect on the configuration of blocks.

The Input

The input begins with an integer n on a line by itself representing the number of blocks in the block world. You may assume that $0 < n < 25$.

The number of blocks is followed by a sequence of block commands, one command per line. Your program should process all commands until the `quit` command is encountered.

You may assume that all commands will be of the form specified above. There will be no syntactically incorrect commands.

The Output

The output should consist of the final state of the blocks world. Each original block position numbered i ($0 \leq i < n$ where n is the number of blocks) should appear followed immediately by a colon. If there is at least a block on it, the colon must be followed by one space, followed by a list of blocks that appear stacked in that position with each block number separated from other block numbers by a space. Don't put any trailing spaces on a line.

There should be one line of output for each block position (i.e., n lines of output where n is the integer on the first line of input).

Sample Input

```
10
move 9 onto 1
move 8 over 1
move 7 over 1
move 6 over 1
pile 8 over 6
pile 8 over 5
move 2 over 1
move 4 over 9
quit
```

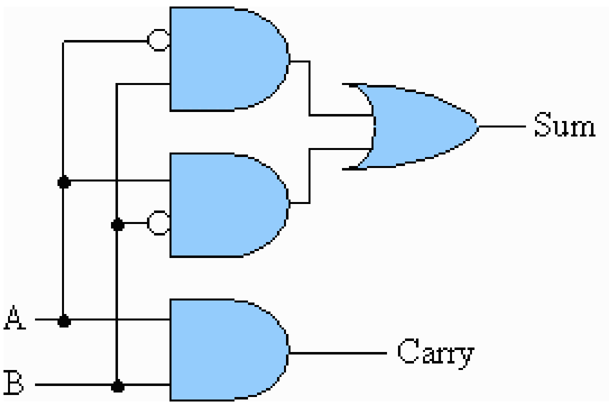
Sample Output

```
 0: 0
 1: 1 9 2 4
 2:
 3: 3
 4:
 5: 5 8 7 6
 6:
 7:
 8:
 9:
```

10469 To Carry or not to Carry

6+9=15 seems okay. But how come 4+6=2?

You see, Mofiz had worked hard throughout his digital logic course, but when he was asked to implement a 32 bit adder for the laboratory exam, he did some mistake in the design part. After tracing the design for half an hour, he found his flaw!! He was doing bitwise addition but his carry bit always had zero output. Thus,



```
   4  =  00000000  00000000  00000000  00000100
  +6  =  00000000  00000000  00000000  00000110
  ----------------------------------------------
   2  =  00000000  00000000  00000000  00000010
```

Its a good thing that he finally found his mistake, but it was too late. Considering his effort throughout the course, the instructor gave him one more chance. Mofiz has to write an efficient program that would take *2 unsigned 32 bit decimal numbers* as input, and produce an *unsigned 32 bit decimal number* as the output adding in the same was as his circuit does.

Input

In each line of input there will be a pair of integer separated by a single space. Input ends at EOF.

Output

For each line of input, output one line — the value after adding the two numbers in the '*Mofiz way*'.

Sample Input

```
4 6
6 9
```

Sample Output

```
2
15
```

490 Rotating Sentences

In “Rotating Sentences,” you are asked to rotate a series of input sentences 90 degrees clockwise. So instead of displaying the input sentences from left to right and top to bottom, your program will display them from top to bottom and right to left.

Input and Output

As input to your program, you will be given a maximum of 100 sentences, each not exceeding 100 characters long. Legal characters include: newline, space, any punctuation characters, digits, and lower case or upper case English letters. (NOTE: Tabs are not legal characters.)

The output of the program should have the last sentence printed out vertically in the leftmost column; the first sentence of the input would subsequently end up at the rightmost column.

Sample Input

```
Rene Decartes once said,
"I think, therefore I am."
```

Sample Output

```
"R
Ie
 n
te
h
iD
ne
kc
,a
 r
tt
he
es
r
eo
fn
oc
re
e
 s
Ia
 i
ad
m,
.
"
```

11958 Coming Home

John is going back home after a party. Currently he is standing on a bus station and waiting for a bus to arrive. There is a timetable of arriving buses near the station. Beside this, John also knows the amount of time that is needed to travel with specific bus. As he has only one ticket, there are no possibilities to change the bus somewhere in the middle of a trip in order to make it shorter. Can you help John to calculate minimal time that he needs to get home?

Input

There is a number of tests T ($T \leq 100$) on the first line. Each test case contains the number of buses K ($1 \leq K \leq 100$) and current time (in format 'HH:MM'. Each of the next K lines contain arrival time of the bus (in the same format as current time) and travelling time $0 \leq Q \leq 1000$ needed for John to get home (in minutes). Refer to the sample input as an example.

Output

For each test case output a single line '`Case` T`:` N'. Where T is the test case number (starting from 1) and N minimal time (in minutes) needed for John to go back home.

Sample Input

```
2
1 18:00
19:30 30
2 18:00
19:00 100
20:00 30
```

Sample Output

```
Case 1: 120
Case 2: 150
```

696 How Many Knights

The knight is a piece used in chess, a game played on a board with squares arranged in rows and columns. A knight attacks pieces that are either (a) two rows and one column away from its position, or (b) one row and two columns away from its position. The following diagram illustrates this. The square marked **N** represents the position of the knight, and the squares marked **X** indicate the squares that are under attack.

	X		**X**	
X				**X**
		N		
X				**X**
	X		**X**	

In this problem you are to determine the largest number of knights that can be placed on a board with M rows and N columns so that no knight is attacking any other. M and N will each be no larger than 500.

Input

The input consists of pairs of integers giving values for M and N, followed by a pair of zeroes.

Output

For each input pair, display the number of rows and columns in the board, and the number of knights that can be appropriately placed.

Sample Input

```
2 3
5 5
4 7
0 0
```

Sample Output

```
4 knights may be placed on a 2 row 3 column board.
13 knights may be placed on a 5 row 5 column board.
14 knights may be placed on a 4 row 7 column board.
```

481 What Goes Up

Write a program that will select the longest *strictly* increasing subsequence from a sequence of integers.

Input

The input file will contain a sequence of integers (positive, negative, and/or zero). Each line of the input file will contain one integer.

Output

The output for this program will be a line indicating the length of the longest subsequence, a newline, a dash character ('-'), a newline, and then the subsequence itself printed with one integer per line. If the input contains more than one longest subsequence, the output file should print the one that occurs last in the input file.

Notice that the second 8 was not included — the subsequence must be strictly increasing.

Sample Input

```
-7
10
9
2
3
8
8
6
```

Sample Output

```
4
-
-7
2
3
8
```

12577 Hajj-e-Akbar

Labayk Allahuma Labayk. Labayk La shareeka laka Labayk. Innal hamda wannimata laka wal mulk. La shareeka Lak

(Here I am at your service, oh Lord, here I am - here I am. No partner do you have. Here I am. Truly, the praise and the favor are yours, and the dominion. No partner do you have.)

These are the words chanted by some two million people from world heading, as if pulled by a magnet, to one single spot on Earth. As has happened every year for 14 centuries, Muslim pilgrims gather in Makkah to perform rituals based on those conducted by the Prophet Muhammad during his last visit to the city.

Performing these rituals, known as the Hajj, is the fifth pillar of Islam and the most significant manifestation of Islamic faith and unity. Undertaking the Hajj at least once is a duty for Muslims who are physically and financially able to make the journey to Makkah. The emphasis on financial ability is meant to ensure that a Muslim takes care of his family first. The requirement that a Muslim be healthy and physically capable of undertaking the pilgrimage is intended to exempt those who cannot endure the rigors of extended travel.

The pilgrimage is the religious high point of a Muslim's life and an event that every Muslim dreams of undertaking. Umrah, the lesser pilgrimage, can be undertaken at any time of the year; Hajj, however, is performed during a five-day period from the ninth through the thirteenth of Dhu Al-Hijjah, the twelfth month of the Muslim lunar calendar.

It is generally presumed that the Hajj performed on Friday is called 'Hajj-e-Akbar' and it is a superior kind of Hajj as compared with the Hajj performed on other days of the week.

But, the correct meaning of the term, as explained by a large number of the commentators of the Holy Quran is that the Umrah, which can be performed at any time throughout the year, was generally called 'Hajj-e-Asghar' (the minor Hajj). In order to distinguish hajj from Umrah the former was named 'Hajj-e-Akbar' (the greater hajj). Therefore, each and every hajj is Hajj-e-Akbar, no matter whether it is performed on Friday or on any other day. The word 'Akbar' (greater) is used only to distinguish it from Umrah which is a minor Hajj.

Input

There will be several lines in the input terminated with a line containing a single '`*`'. This last line should not be processed. Each of the lines will contain either Hajj or Umrah.

Output

For each line of the input, output either '`Hajj-e-Akbar`' or '`Hajj-e-Asghar`' in separate lines without quotations. For exact format refer to the sample.

Sample Input

```
Hajj
Umrah
```

```
Hajj
Umrah
*
```

Sample Output

```
Case 1: Hajj-e-Akbar
Case 2: Hajj-e-Asghar
Case 3: Hajj-e-Akbar
Case 4: Hajj-e-Asghar
```