A. Mia

Mia is a dice game for two players. Each roll consist of two dice. Mia involves bluffing about what a player has rolled, but in this problem we focus only on its scoring rules. Unlike most other dice games, the score of a roll is not simply the sum of the dice.

Instead, a roll is scored as follows:

- Mia (12 or 21) is always highest.
- Next come doubles (11, 22, and so on). Ties are broken by value, with 66 being highest.
- All remaining rolls are sorted such that the highest number comes first, which results in a two-digit number. The value of the roll is the value of that number, e.g. 3 and 4 becomes 43.

Input

The input will contain multiple test cases. Each test case contains on a single line four integer numbers s_0 s_1 r_0 r_1 where s_0 s_1 represent the dice rolled by player 1 and r_0 r_1 represents the dice rolled by player 2. The input will be terminated by a line containing 4 zeros.

Output

For each test case, output which player won, or whether there was a tie, using exactly the format shown below.

Sample Input

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

Sample Output

```
Player 1 wins.
Player 2 wins.
Player 1 wins.
Player 2 wins.
Tie.
```

B. Odds of Mia

This problem relates to the game of Mia introduced earlier. We recommend that you first solve Mia to understand how the game is scored. In this problem, you are asked to compute the odds that player 1 will win given partial knowledge of both rolls.

Input

The input will contain multiple test cases. Each test case contains on a single line four symbols s_0 s_1 r_0 r_1 where s_0 s_1 represent the dice rolled by players 1 and r_0 r_1 represents the dice rolled by player 2. A * represents that the value is not known, otherwise a digit represents the value of the dice. The input will be terminated by a line containing 4 zeros.

Output

For each test case output the odds that player 1 will win. If the odds are 0 or 1, output `0` or `1`. Otherwise, output the odds in the form `a/b` where a and b represent the nominator and denominator of a reduced fraction (i.e., in lowest terms).

Sample Input

```
* * 1 2
1 2 * *
1 2 1 3
3 1 2 1
6 6 6 6
* 2 2 2
* 2 * 6
* * * *
0 0 0 0
```

Sample Output

```
0
17/18
1
0
0
1/6
1/3
205/432
```

Explanation: for `* * 1 2`, the best player 1 can do is tie, so his chance of winning is 0. For `1 2 * *`, player 1 wins unless player 2 rolls a Mia, which happens 1 out of 18 times. For `1 2 1 3`, `3 1 2 1`, and `6 6 6 6` the result is already known. For `* 2 2 2`, player 1 wins only if she rolls a 1. For `* 2 * 6`, player 1 wins if he rolls a 1. If he rolls a 2, he wins with probability 5/6. He loses if he rolls a 3, 4, or 5. If he rolls a 6 he wins only if player 2 rolls a 1. Thus, his chance of winning is 1/6 + 5/6 * 1/6 + 1/6 * 1/6 = 12/36 = 1/3. When no dice are known, Player 1 will win in 615 of all possible 1,296 outcomes. Player 2 will lose in 615 cases, and there are 66 possible ties. Thus, her chance of winning is 615/1296 = 205/432.

C. Beehives

Bill the beekeeper has a problem! His bees like to fight with each other instead of producing honey. If the bees fight each other, then the honey turns out sour. The only way to stop the bees from fighting and keep the honey sweet is to ensure that the hives are not too close together. Help Bill figure out how many of this season's hives will produce sour honey.

Bill's hives all exist on a plane, and he knows some distance *d* such that two hives within *d* of each other will fight, and both will produce sour honey. If a hive does not fight with any other hives, it produces sweet honey. Given *d* and the positions of *N* beehives, output how many hives will produce sweet honey and how many hives will produce sour honey.

Input

Input will be provided on multiple lines. Each case will begin with a floating point number d ($0 < d < 1000.0$), the distance within which hives will fight. On the next line will be N ($1 \le N \le 100$), the number of hives in that case. The next N lines will contain two floating point numbers separated by a single space, x and y ($-1000.0 \le x,y \le 1000.0$), which give the position of each hive. No two hives will be at the exact same location. Input will be terminated by a line containing the string `0.0 0`.

Output

For each case output a line of the following form: *a* `sour,` *b* `sweet` where *a* and *b* are the number of hives producing sour and sweet honey, respectively.

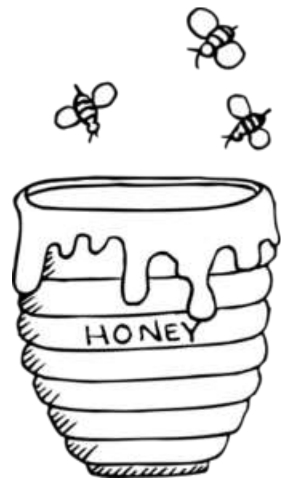
Sample Input

```
10.0 3
3.0 3.0
3.5 4.5
20.0 20.0
5.0 4
2.0 1.0
2.0 4.0
2.0 8.0
2.0 12.0
0.0 0
```

Sample Output

```
2 sour, 1 sweet
4 sour, 0 sweet
```

D. Left Beehind



Beekeeper Bill is ready to go to the annual beekeeper's convention with his friends. Bill packed his honey in jars, but unfortunately the honey in some of the jars has turned sour. Bill's friends are mean, and if too much of Bill's honey is sour they will go to the convention without him.

If Bill has more jars of sour honey than sweet, he will be left “beehind”. If Bill has more jars of sweet honey than sour he will go to the convention. If Bill has the same number of sweet and sour jars, his friends are undecided. Lastly, Bill's friends are superstitious, if he has exactly 13 jars they will never speak to him again. Bill needs new friends.

Input

Input consists of multiple cases, each on its own line. Each case consists of two numbers *x* and *y* ($0 \le x, y \le 1000$), which are the number of sweet and sour jars Bill has, respectively. Input is terminated by a line containing two zeros.

Output

For each case, output one of the following results on its own line (without quotes).

- “Left beehind.” (note the spelling, those punsters)
- “Undecided.”
- “To the convention.”
- “Never speak again.”

If Bill's friends will never speak to him again, that is most important and should be the only output.

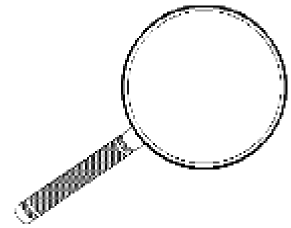
Sample Input

```
17 3
13 14
8 5
44 44
0 0
```

Sample Output

```
To the convention.
Left beehind.
Never speak again.
Undecided.
```

E. Beekeeper



It's Bill the beekeeper again. In this problem, we learn that didn't choose his career for the best reason. Bill chose to be a beekeeper because he really, really likes the look of the double e's in the word "beekeeper". Your task is to help identify other words Bill will really like.

Bill's affinity for a word is determined by how many pairs of double vowels it has, where vowels are the letters {a,e,i,o,u,y}. Given a list of words, find the word that has the most pairs of double vowels, as it will be Bill's favorite word.

Notes:

- It is guaranteed that there will be a unique correct answer. That is, one word in the list will have strictly more pairs of vowels than all other words in the list.
- No word in the input will have three or more of the same vowel consecutively, because that would just be too much for poor Bill.

Input

The first line of each case contains *N*, the number of words in that case ($0 < N \leq 1000$). The next *N* lines will contain the words that make up that case. Each word will contain only lowercase English letters `[a-z]`, and no word will be longer than 80 characters. Words will not necessarily be actual English words, but they will contain at least one letter.
The input will be terminated by a line containing a single zero.

Output

For each case, print Bill's favorite word. Each output should be printed on its own line.

Sample Input

```
4
artist
engineer
beekeeper
programmer
3
bookkatt
jailaikia
yeehaaw
0
```

Sample Output

```
beekeeper
yeehaaw
```

F. Carousel Rides

Carl likes to ride the carousel. Carousel operators often offer discounts for buying multiple rides. He wonders which of the discounts provides the best value.

Write a program to help him.

Input

The input will contain multiple test cases. A test case starts with a line containing two numbers n (1 ≤ n ≤ 10) and `m` (1 ≤ `m` ≤ 20). Carl will not take advantage of offers that require him to buy more than `m` tickets. Following this are n lines, each with numbers a and b which each represent an offer to buy a tickets for `$b`.

The input will be terminated by a line containing the characters `0 0`.

Output

For each test case, print `Buy a tickets for $b` for the best offer that matches his requirements. If there are multiple best offers, print the one which buys more tickets. If there is no suitable offer, print `No suitable tickets offered`.

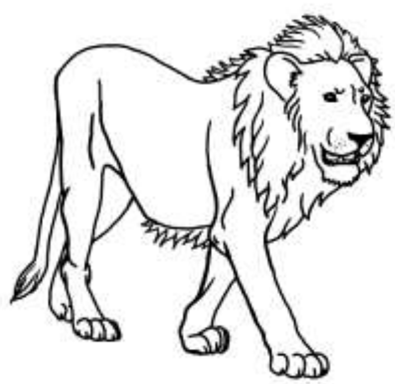
Sample Input

```
3 5
1 3
3 5
4 7
3 2
3 5
1 3
4 7
3 2
3 6
1 2
2 4
1 3
4 10
0 0
```

Sample Output

```
Buy 3 tickets for $5
Buy 1 tickets for $3
Buy 2 tickets for $4
No suitable tickets offered
```

G. Un-bear-able Zoo



In his free time, when he's not busy hacking computers, Dr. Back runs a zoo. Every morning he gets up and makes sure that none of the animals have escaped. He has a huge list of all the animals and checks each animal off as he sees it, but thinks this is really inefficient. He only cares about what animal they are, since all similar animals share a cage. So, if he has a white tiger and a siberian tiger, Dr. Back only wants to know that he has 2 tigers.

Given an integer n, and n lines describing animals, output in alphabetical order the animals Dr. Back has in his zoo, followed by their count.

Input

The input will contain multiple test cases. Each test case contains a line containing a single integer n ($0 \le n \le 10^3$), followed by n lines of animals with at least one word on every line. An animal name may consist of multiple lowercase or uppercase words, with the last one describing the kind of animal. The input is terminated when n is 0.

Output

For each test case, output the list number, followed by the animals Dr. Back has in his zoo in lowercase and alphabetical order, with each animal followed by one space and the delimiter | and then another space and their count.

Sample Input

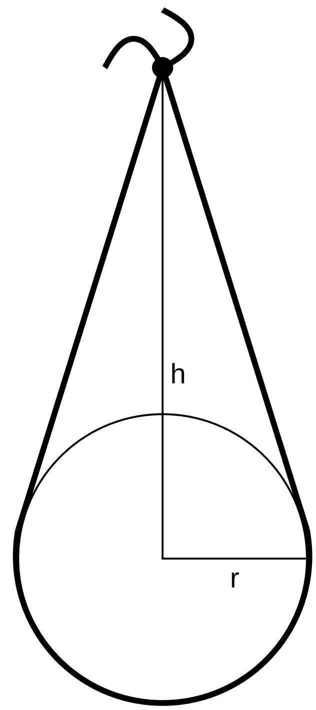
```
6
African elephant
White tiger
Indian elephant
Siberian tiger
Tiger
Panda bear
1
Blue Russian Penguin
0
```

Sample Output

```
List 1:
bear | 1
elephant | 2
tiger | 3
List 2:
penguin | 1
```

H. Ornaments

A factory is making ornaments for the upcoming holidays. Ornaments consist of a 2D glass sphere (i.e., a circle) that is designed to hang off a hook or nail using a string that tightly wraps around the ornament, as shown in the figure to the right.



Your task is to write a program that calculates the length of the string, given the radius r of the circle, the distance h from the knot to the center of the circle, and some multiplier to account for the excess needed to tie the knot.

Input

The input will contain multiple test cases. Each test case contains on a single line three integer numbers r ($1 \le r \le 1000$), h ($r \le h \le 10000$), and s ($0 \le s \le 100$). s denotes the necessary excess string in %. The percentage is relative to what would be needed without accounting for the knot, so if s = 25, you should increment the necessary length by 1/4.

The input will be terminated by a line containing 3 zeros.

Output

For each test case, output the length of the needed string, rounded to two decimals.

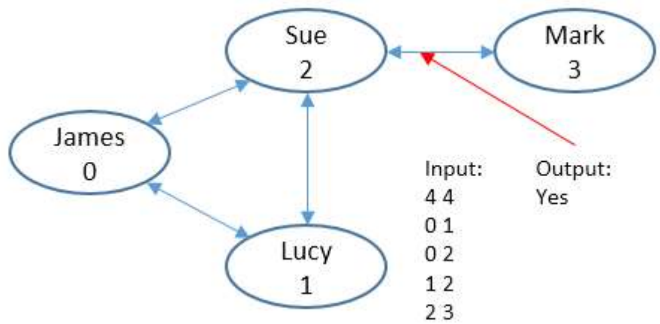
Sample Input

```
1 3 10
2 5 8
10 11 0
0 0 0
```

Sample Output

```
10.43
18.46
63.40
```

I. James's Birthday Party



James needs some help figuring out whether he can invite all of his friends to his birthday party. Luckily for James, he doesn't have to invite all of his friends by himself, because his friends will forward invitations to each other.

Consider this example: James has three friends: Lucy, Sue, and Mark. James is friends with all three of them, but unfortunately he doesn't have Mark's phone number so he can't invite him directly. However, Sue has Mark's number (and vice versa) so she can invite Mark to James's party.

The question that you need to answer is that if anyone lost one of the phone numbers for anyone else, would it be impossible to invite everybody to the party? Continuing the above example, if Sue lost Mark's number (the edge denoted by the red arrow above), then it would not be possible to invite everybody to the party. You should assume that if Sue loses Mark's number, then Mark will also lose Sue's number. Please help James figure out if he is at risk at having someone miss his party if any pair of friends loses contact with each other.

Input

The input will contain multiple test cases. Each test case contains on a single line two integer numbers p ($1 \le p \le 100$) and c ($0 \le c \le 5000$). p represents the number of people James wants to invite to the party, including himself. The next c lines represent the connections among James's friends represented as two integers *a* ($0 \le a < p$) and *b* ($0 \le b < p$), where *a* is not equal to *b*. This means that friend number *a* has friend number *b*'s phone number and vice versa.

The input will be terminated by a line containing 2 zeros.

Output

For each test case, if a pair could lose each other's numbers and make it so that not everybody can be invited to the party, print "Yes". Otherwise, print "No".

(continued on next page)

Sample Input

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

Sample Output

```
Yes
No
Yes
```

J. Cracker Barrel Game

The popular Cracker Barrel country store chain offers its clientele a peg game to pass the time until their food arrives. The game is played on a 15-peg board. The rules are simple: each move jumps one peg over another peg into a free hole. The peg that's jumped over is removed. If only one peg remains, the player wins. Usually, the game is started with 14 pegs (and one hole). Pegs may have different colors: blue, red, yellow, white, though the color of a peg usually does not matter.

In this problem, you are giving a start position of between 1 and 14 pegs of different colors, as well as a target color. You should output whether it is possible to remove all but one peg from the board using the usual rules and end up with a peg that is of the target color.

Input

The input will contain multiple test cases. Each test contains a target peg on a single line, followed by the initial position of the board. A peg is represented as a pair of two uppercase letters, e.g. BB represents a peg of a certain color (say blue). The board is given using appropriate indentation of 4, 3, 2, and 1 spaces for the 1^{st}, 2^{nd}, 3^{rd}, and 4^{th} row.

The input will be terminated by a line containing the characters **.

Output

For each test case print `Possible` if there exists a sequence of valid moves such that all but one peg can be removed and you end up with a peg of the given target color. Otherwise, print `Impossible`.

Sample Input

```
BB
    __
   RR__
  __YY__
 ____GG__
______WWBB
WW
    __
   RR__
  __YY__
 ____GG__
______WWBB
BB
    __
   BBBB
  BBRRRR
 RRRRWWYY
WWWWYYBBRR
YY
    __
   BBBB
  BBYYRR
 RRRRWWBB
WWWWRRBBRR
BB
    __
   BBBB
  YYYYYY
 RRRR__BB
WW__RRBBRR
YY
    __
   BBBB
  YYYYYY
 RRRR__BB
WW__RRBBRR
**
```

Sample Output

```
Possible
Impossible
Possible
Impossible
Possible
Impossible
```