What is the air speed velocity…

…of a fully laden swallow? This fearful question was posed to the intrepid band of Grail searchers. Their response of "African or European?" was partly correct. The air speed would most definitely depend on the sub-species of swallow. King Arthur, fearing more intense questioning in this vein, ordered his royal mathematicians to determine the air speed of a fully laden swallow – both African and European.

The mathematicians called upon the royal birders to capture a number of swallows of both types, lade them fully, and then release them from point A and time their arrival at point B. Since they didn't want to confuse their figures, the European and African swallows were each started from a different location, so that each group flew a different distance, but all swallows in the same group flew the same distance. They then asked the royal map-makers to determine the distance (measured in furlongs) between the two starting points and the finish point. Using 10 swallows of each type, the royal mathematicians would then compute the average air speed for each group.

However, the royal mathematicians were somewhat lazy. After gathering all the data, they decided it was MUCH too hard to do all those nasty calculations by hand. So, they quickly constructed a time machine and have come into the future to enlist your help: they need you to write a program to do the calculations, which they will then take back into the past with them. Thus the searchers of the Grail will be saved from certain doom, (should this dastardly question be posed again), and you will go down in history as a hero. (Well, maybe not history, since they are from the past, so maybe you'll go down in futurory?)

There's one tricky bit (you knew it was coming): the royal mathematicians cannot agree on exactly how the average should be calculated. Some believe that, for each group, one should add up all of the times and then divide the total distance covered by all the swallows of that type by the total time (this is method 1). Others are of the opinion that the average speed is determined by computing the speed for each swallow, summing those values and then dividing that total by the total number of swallows (this is method 2). Your program should compute the average both ways, to avoid a nasty falling out among the royal mathematicians.

The input provided by the royal mathematicians is somewhat disorganized – the two breeds' times have been intermixed and they weren't too careful about capitalization. But each entry is on a separate line and marked with an 'A' or 'a' or 'E' or 'e' to aid in identification. Each line begins with this single letter, followed by a single space. The final datum on the line is the elapsed time the swallow flew, expressed in hours. Since the time-keeping of the era wasn't very accurate, this value is simply a real number (> 0) with a single level of precision, such as 1.5 (one and a half hours), or 0.4 (four-tenths of an hour).

Input

The very first line of the input file will consist of 2 integers, both greater than 0, separated by a single space. The first integer is the distance the African swallows flew and the second integer is the distance the European swallows flew. Next come the times for the swallows (20 lines total: 10 for African, 10 for European – NOT in this order). Thus the input file has a total of 21 lines.

Output

The format for the output should be grouped by methods, with Method 1 being displayed first. Each method will produce 3 lines of output:
- **Line 1:** the name of the method (capitalized, with a digit identifying it), e.g. "Method 1" (the quotes are not part of your output.)
- **Line 2:** the speed of a fully-laden African swallow (expressed in furlongs per hour), e.g. "African: 3.00 furlongs per hour" (the quotes are not part of your output.)
- **Line 3:** the speed of a fully-laden European swallow (expressed in furlongs per hour) e.g. "European: 3.00 furlongs per hour" (the quotes are not part of your output.)

The format for the data for each method is as follows:
- The full name of the breed of swallow (capitalized), beginning with African
- A colon
- A single space
- The speed (to two digits of accuracy, with leading 0 for values < 1.0)
- A single space
- The phrase "furlongs per hour" (the quotes are not part of the output).

See the Sample output section below for any clarifications you require.

Sample Input
```
6 5
a 1.0
A 1.0
E 2.0
E 2.0
A 1.0
e 2.0
a 1.0
A 1.0
E 2.0
E 2.0
A 1.0
e 2.0
a 1.0
A 1.0
E 2.0
E 2.0
A 1.0
e 2.0
e 1.0
a 2.0
```

Sample Output
```
Method 1
African: 5.45 furlongs per hour
European: 2.63 furlongs per hour
Method 2
African: 5.70 furlongs per hour
European: 2.75 furlongs per hour
```

The Brave Sir Robin's cAsE cOrReCtOr

Dissatisfied with the loud and constant pronouncements of his alleged misdeeds by a trio of indefatigable minstrels, the brave knight Sir Robin wishes to exercise his authority by modifying their lyrics. The minstrels were happy to provide printed transcripts of their songs, and cheerfully announced that they would not change a word of them.

Undaunted, the brave (and crafty) Sir Robin scrutinized the documents and noticed that their loudest inflections were indicated by capital letters and realized that he could at least lower their voices. This, he reasoned, could be accomplished by replacing upper case letters with lower case letters ("Case correction", from his perspective). These modifications could be forced upon the singers by insistence upon proper usage of the King's English. Not all letters can be lower case, however, as the King's English mandates some letters must be upper case.

Strangely hesitant about performing "case correction" personally, the brave, crafty (and managerially capable) Sir Robin humbly requests you write a program to perform a first pass of case correction for the songs. There will still be some corrections required after this program is used.

As your program reads the file, it must force to upper case all alphabetic characters that follow terminal punctuation marks (period, question mark, and exclamation point) with only white space or parentheses characters following. All other alphabetic characters are to be forced to lower case. Note that decimal numbers are not to be followed by an upper case character unless the number itself is followed by a terminal punctuation mark.

Input

The input file contains the text that you are converting. Your conversions should be based on the rules given by Brave Sir Robin above.

Output

The output is to be the converted text. All characters are transferred to the output. Some will have cAsE cOrReCtiOn, others will be directly copied.

Sample input

```
The Brave Sir Robin took a short walk in a dark forest where rabbits did stalk.  a
ray of sunlight made him jump from his own shadow with A FACE AS PALE AS CHALK.
```

Sample output

```
the brave sir robin took a short walk in a dark forest where rabbits did stalk.  A
ray of sunlight made him jump from his own shadow with a face as pale as chalk.
```

Sir Bedavere's Bogus Division Solutions

The wise Sir Bedavere often uses non-standard logic, yet achieves positive results[1]. Well, it seems he has been at it again, this time with division. He has determined that canceling the common digit of a numerator and denominator produces the correct answer. Of course, Sir Bedavere only tried this on a small sample of three digit numbers. An example of what he did is shown in the following division problem (in which he canceled the common 6):

$$\frac{166}{664} = \frac{16\cancel{6}}{\cancel{6}64} = \frac{16}{64}$$

Your task is to find all three digit number combinations with the following property:

number combinations where removing the rightmost digit from the top number (numerator) and the identical leftmost digit from the bottom number (denominator) leaves the result of the calculation unchanged.

Omit all of the trivial cases — xxx/xxx = xx/xx (222/222 = 22/22). The solutions are to be shown in increasing order of the top number (the numerator).

The Input

NONE! There is no input for this problem.

The Output

Show the bogus division problems one to a line in the format shown below (which gives a *sample* merely to show the format) — single spaces separate the non-blank characters.

```
217 / 775 = 21 / 75
249 / 996 = 24 / 96
```

---

[1] Please see the scene "How do you know she's a witch?" or recall the quote "…how sheep's bladders may be employed to prevent earthquakes."

One…Two…Five!

The set of integers has rarely been a domain of error in everyday conversation. The king, however, is "three blind" and cannot visualize any number containing the digit '3' in its base 10 representation. He does intuitively sense the number between 2 and 4 and compensates for his blindness in the following manner: whenever he wants to state any number containing the digit '3', he will speak a series of numbers until they can all be combined (in the order given) via addition, subtraction, multiplication and division to produce the desired value which contains the digit '3'. Mathematical operators work from left to right without any other regard for order of precedence (i.e. 6 + 7 * 11 = 143 → a number with a '3').

For example, if the king says "1 2 5", then a knight will say "3" using the following logic:

```
1 2 5 = 1 * 2 – 5 = 2 – 5 = |2 – 5| = 3
```

Note that there are no negative numbers in optimistic Camelot. Every subtraction will produce a nonnegative result by what is called, in these enlightened times, the absolute value. All division is integer division, i.e. 7/5 = 1. Obviously, if the number zero appears as a divisor, then division will not be attempted.

The court, however, has a problem. Some of the computations produce more than 1 number containing the digit '3'! You have been appointed to write a program which computes and displays the most frequently appearing number containing the digit '3'. In the event of a tie, use the largest number.

Input

The input will consist of an unspecified number of lines. Each line will contain at least 1 and at most 9 integers. Every number will be nonnegative and less than 100. A line with a single '#' character will be the end of input.

Output

For every line of input, print a line showing the number most probably meant by the king as described above. If there is no such number, print "No result".

Sample Input	Sample Output
1 2 5	3
1 1	No result
6 5 1	30
#	

Taunt Exposure Estimation

The brave knights *(kə' nig' əts)* of Camelot are constantly exposed to French taunting while assaulting the castle occupied by the French. Consequently, the taunting to which they are exposed varies with their distance from the castle during their assault, as well as variations in French taunting activity. We need to estimate the total amount of taunting that they are exposed to during a certain time period. Unfortunately, we only have access to a set of measurements at random times — we do not have a continuous reading — and, because of flaws in our archaic equipment, the measurements of taunting occur at unpredictable intervals.

The total amount of taunting will be given by the integral of the taunting intensity during the time period, as held in the observation data file. The amount of random noise, though, is fairly high, so that a simple trapezoid-rule integration is all that is merited.
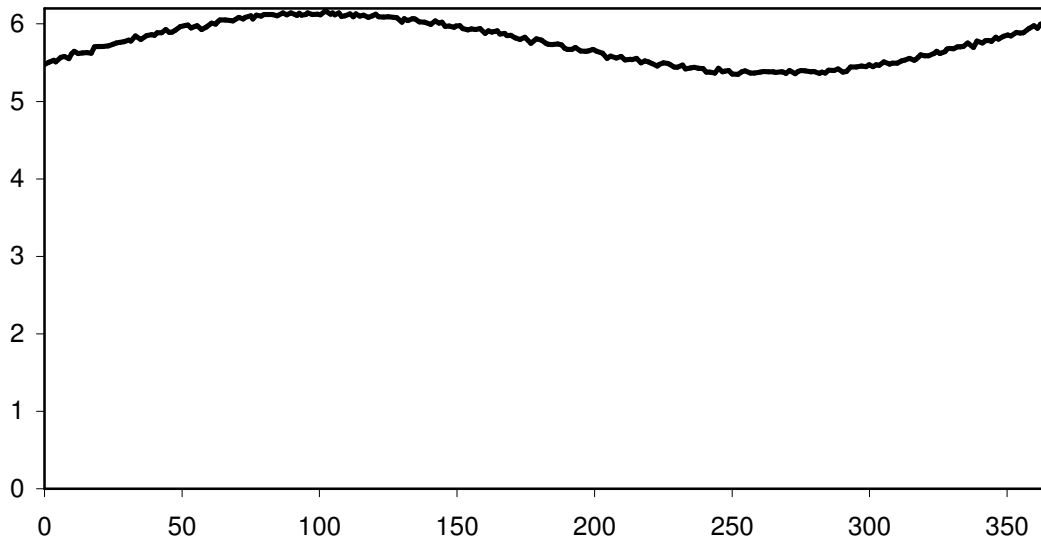
The Input

- A single number, *n*, specifying the number of data points in the file
- *n* pairs of floating point numbers (given in increasing x order), separated by a comma — in other words, a CSV file that *could* be input for a spreadsheet program [the first number is the x coordinate (time specification), the second is the y coordinate (the radiation reading)][1]

The Output

A single line of text giving the first and last x values (with two digits to the right of the decimal point), and the computed integral (with four digits to the right of the decimal point), in the fashion shown below (which reflects the data shown in the graph):

```
0.00 to 365.25: 2099.8021
```

[A reasonable value for the given input, (shown in the graph above), since the values range around 5 ¾, and 365.25 * 5.75 gives 2100.1875.]

---

[1] Sample input and output on the following page (according to Ralph The Wonder Llama)

Sample Input[2]

```
9
 0.0000, 0.5176
 0.9869,  1.000
  1.596,  1.114
  2.370,  1.006
  2.904, 0.8481
  3.506, 0.5760
  3.996, 0.4775
  5.004, 0.3945
  6.283,  1.004
```

Sample Output[3]

```
0.00 to 6.28: 4.7288
```

---

[2] A Møøse once bit my sister...

[3] We apologise for the fault in the footnotes. Those responsible have been sacked.

Ye Holy Hand Grenades!

Holy Hand Grenades have found multiple uses throughout history. Most notably, King Arthur did use a Holy Hand Grenade (HHG) to dispatch a vicious rabbit which was guarding the entrance to a cave. The instructions for its use are in the Book of Armaments (Chapter 2, verses 9-21), and are read as follows:

"First shalt thou take out the Holy Pin. Then, shalt thou count to three. No more, no less. Three shall be the number thou shalt count, and the number of the counting shall be three. Four shalt thou not count, neither count thou two, excepting that thou then proceed to three. Five is right out. Once the number three, being the third number, be reached, then lobbest thou thy Holy Hand Grenade of Antioch towards thy foe, who, being naughty in My sight, shall snuff it." Amen.

Unfortunately, the Book of Armaments faileth to mention that the HHG must come to a complete stop (i.e. rest stably) before it can explodeth and blow a rabbit to tiny bits. As long as it rolleth or wobbeleth, it refuseth to explodeth. Unknown is the reason why the authors of the Book of Armaments neglected to mention this pertinent fact. Further complicating matters, different HHGs have different radii of destruction (*RD*). Once a HHG resteth stably, and explodeth, everything within or equal to its *RD* shalt be snuffed out. Presumably, this is what happened to the rabbit with big, pointy teeth; however, there remaineth some controversy over this matter. Because of this controversy, the king hath decreed a simulation to be run that may possibly answer this question.
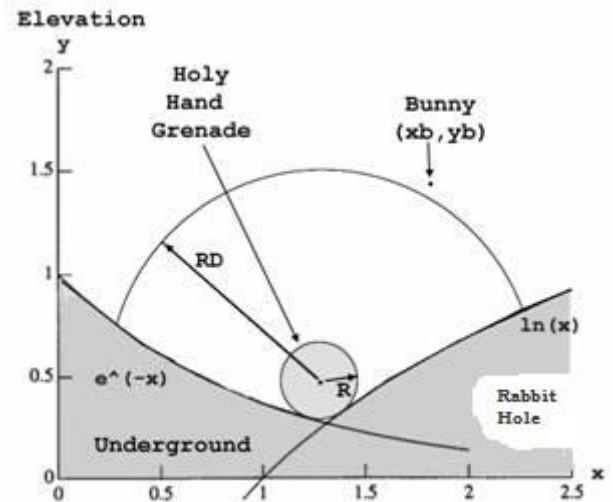
- Your task is to determine the result of the grenade's explosion.
- The grenade must always be at rest in order to detonate. All HHGs have a radius R=1 and a variable radius of destruction (RD). At the instant of detonation, the rabbit could be mid-leap, on the ground, or underground.
- For each simulation the terrain remaineth unchanged. It consisteth of two intersecting curves, $y = e^{-x}$ and $y = ln(x)$ as we have revealed unto you in the illustrious illustration. The curves reside in the plane formed by the rabbit, the grenade, and the center of the earth. The terrain is impervious to detonations, and changeth not between tests.
- If a bunny's center of mass is strictly below the ground level (as denoted by the intersecting curves) when a HHG explodeth, that bunny shalt remaineth un-snuffedeth and thus shalt live to bite another day.
- If the bunny resideth strictly outside of the range of a stably resting HHG's radius of destruction, that bunny shalt also remain un-snuffedeth out.
- Each result shalt be dependent upon computations accurate even unto 8 significant figures.
- There shalt not be an input value smaller even than 1.0E-15 or greater even than 1.0E+15.
- Because the bunny's center of mass doth be its location, a bunny might possibly end up inside both the *RD* and the R=1 of a HHG. This simply means the bunny is curled around the grenade. If the center of mass of such a bunny be strictly inside the *RD* it shall most surely be snufféd out. Amen.

Input
The **first line** of input doth contain yea verily a **single integer** indicating the **number of holy hand grenades** in the data file. Each **line of the file** doth contain the radius of destruction of this particular holy hand grenade, and finally doth contain the rabbit's x coordinate (xb) and y coordinate (yb) during this particular test run.

Output
For each test run, thy program shalt print out the words "**Bunny Bits**" (if the rabbit is blown to bits by the grenade) or "**Bunny Biteth Knights**" (if the rabbit liveth to fight on another day anon).

Sample Input	Sample Output
2 1.5 3.0 3.0 5.5 3.0 3.0	Bunny Biteth Knights Bunny Bits

Nested Shrubbery Boxes

After each commission to install a shrubbery, Roger the Shrubber has to transport many empty planting boxes with a drawn cart. In this instance, a planting box is a wooden box with one open side.

Given a set of n planting boxes, compute the largest number of boxes that can be nested. Specifically, report the number of the largest subset of boxes which may be nested such that the smallest box of the subset fits within the second smallest, the second smallest of the subset fits within the third smallest, the third smallest of the subset fits within the fourth smallest, and so forth.

A box **i** (b_i) fits into box **j** (b_j) if there exists some rotation of b_i such that each dimension of b_i is less than the corresponding dimension of b_j. Any box can be rotated to nest inside another box.

Input

The input will consist of an unspecified number of box sets. Each set will begin with a line containing n, $0 \leq n \leq 500$, the number of boxes in the set. Each box will be described on its own line by three positive integers representing length, width and height (Each value will not exceed 1000). The first two numbers of each box description will be followed by a space, the letter '**x**', and a space. The end of input occurs when $n = -1$.

Output

For each set of boxes, print a line containing the largest number of boxes that can be selected from the original set to form a fully nesting subset of boxes.

Sample Input	Sample Output
5 145 x 472 x 812 827 x 133 x 549 381 x 371 x 900 271 x 389 x 128 718 x 217 x 491 4 432 x 123 x 139 942 x 844 x 783 481 x 487 x 577 677 x 581 x 701 -1	2 4

And Now For Something Completely Different!

Bart and Lisa Simpson have many chores, but they don't always do them well. So their father began keeping score, giving them credit only if a chore was done right. After a month, he showed them his record keeping.

Assigned chores for first month							Chores done correctly	
Bart	Bart	Bart	Lisa	Lisa	Lisa	Bart	Bart	Lisa
1	0	1	0	1	1	1	75.00 %	66.67 %

This showed that Bart had done 3 out of 4 chores correctly and Lisa had done 2 out of 3 chores correctly. Then for a second month, Bart and Lisa did chores and the record keeping looked like this:

Assigned chores for second month									Chores done correctly	
Bart	Bart	Lisa	Lisa	Lisa	Lisa	Lisa	Lisa	Lisa	Bart	Lisa
0	1	0	1	1	0	1	0	0	50.00 %	42.86 %

The father made the following chart to support the hypothesis that Bart was doing the most chores:

	Month 1	Month 2	Computation	Total
Bart	75.00 %	50.00 %	125.00 / 234.53	53.30 %
Lisa	66.67 %	42.86 %	109.53 / 234.53	46.70 %

Lisa looked at the tables then said "No no no, that's not right, I've done more." Bart pointed at the statistics and said, "Numbers don't lie." Lisa said, "But look, I've done 5 chores and you've only done 4. You just look good because you always run away before somebody asks you to do something. The total should give me 55% of the total chores done right." Bart laughed and ran away. Now Lisa needs your help to show the unfairness of these chore evaluations. She wants a program which computes the averages over the entire dataset and displays where the before and after averages support the opposite hypothesis than would be indicated by considering the total dataset. To be fair, she only wants to consider record keeping times when both she and Bart have chores assigned both before and after the record keeping.

Input

The first line of each test case contains one integer N (2 <= N <= 50000) the number of chore records. Each of the next N lines in each test case contains either "Bart" or "Lisa" followed by an integer, 0 or 1. The name shows who was assigned a chore, and the integer is a value of 1 if it was done correctly, or a 0 if it was not. Both Bart and Lisa will have chores assigned to them in each dataset. Input ends when $N = 0$.

Output

Write on the first line of the output for each test case, the test case number, and the number of chores correctly performed by Bart (**CB**) and Lisa (**CL**) in the following format:

    Case **<case number>**: Bart did **<CB>** and Lisa did **<CL>**

If Bart and Lisa do the same number of chores over the entire dataset, there is no trend to oppose. In that case, print "Bart and Lisa accomplished same number of chores".

If Bart and Lisa do a different number of chores over the entire dataset, determine if there are any places in the dataset where taking the averages of all results before and after a record keeping break would produce results which would indicate a trend opposing the trend over the entire dataset. If there are no such cases, print on the next line, "Simpson's paradox not detected". Otherwise, print on the next line "Trend measured in 2 parts is reversed" and on every following line print the following:

    After chore **<chore number>**: **<B_{Before}>**% **<L_{Before}>**% **<B_{After}>**% **<L_{After}>**%

where **B_{Before}** is Bart's correctly performed chore percentage before the record keeping, **L_{Before}** is Lisa's correctly performed chore percentage before the recordkeeping, **B_{After}** is Bart's correctly performed chore percentage after the first record keeping until the end of the dataset, and **L_{After}** is Lisa's correctly performed chore percentage after the first record keeping until the end of the dataset. Note that chores are numbered from 1 to N in order of appearance in the dataset. All percentages must be rounded to the nearest .0001. Print a blank line between the outputs for two consecutive test cases.

Clarification: Because Lisa wants a program which "displays where the before and after averages support the opposite hypothesis than would be indicated by considering the total dataset", then any averages which, after rounding, are printed as equal must be disregarded.

Sample Input

```
16
Bart 1
Bart 0
Bart 1
Lisa 0
Lisa 1
Lisa 1
Bart 1
Bart 0
Bart 1
Lisa 0
Lisa 1
Lisa 1
Lisa 0
Lisa 1
Lisa 0
Lisa 0
2
Lisa 1
Bart 1
2
Lisa 0
Bart 1
0
```

Sample Output

```
Case 1: Bart did 4 and Lisa did 5
Trend measured in 2 parts is reversed
After chore 4: 66.6667% 0.0000% 66.6667% 55.5556%
After chore 5: 66.6667% 50.0000% 66.6667% 50.0000%
After chore 7: 75.0000% 66.6667% 50.0000% 42.8571%

Case 2: Bart did 1 and Lisa did 1
Bart and Lisa accomplished same number of chores

Case 3: Bart did 1 and Lisa did 0
Simpson's paradox not detected
```

Taunt Generation Simulator

In all the annals of knighthood, no personality trait has been in more dire need than the fortitude
to withstand the diplomatically deleterious effects of a vicious, relentless taunting. Tasked with
strengthening the mental endurance of Camelot's knights, King Arthur's court decided that
instructional taunting must be applied, yet it could not be delivered by the chivalrous
administration. Thus, Sir Lancelot commanded a local anarcho-syndicalist peasant to write a
program that generates taunts (a.k.a. mudslinging) thereby producing a script to test the
patience of knights in a training environment. To prevent unbridled creativity in taunting from
spoiling the otherwise stately proceedings of a nobleman's education, the following rules
designed by committee (The Round Table) must be adhered to:

```
<taunt> ::= <sentence> | <taunt> <sentence> | <noun>! | <sentence>
<sentence> ::= <past-rel> <noun-phrase> | <present-rel> <noun-phrase> | <past-rel> <article> <noun>
<noun-phrase> ::= <article> <modified-noun>
<modified-noun> ::= <noun> | <modifier> <noun>
<modifier> ::= <adjective> | <adverb> <adjective>
<present-rel> ::= your <present-person> <present-verb>
<past-rel> ::= your <past-person> <past-verb>
<present-person> ::= steed | king | first-born
<past-person> ::= mother | father | grandmother | grandfather | godfather
<noun> ::= hamster | coconut | duck | herring | newt | peril | chicken | vole | parrot | mouse | twit
<present-verb> ::= is | "masquerades as"
<past-verb> ::= was | personified
<article> ::= a
<adjective> ::= silly | wicked | sordid | naughty | repulsive | malodorous | ill-tempered
<adverb> ::= conspicuously | categorically | positively | cruelly | incontrovertibly
```

Note that all phrases in double quotes are to be treated as one word for taunt simulation output.

The number of taunts elicited at any given time is derived from the number of words spoken by
the knight. For every three words (or fraction thereof) delivered by the knight, the generator
produces one or more taunts in a theater-style script format. In the event that 2 taunts must be
produced on a single line, they will be counted as 2 taunts towards the total required. By a
mandate from the masses, a word will always contain at least one alphabetic character, and will
be separated from other words by at least 1 space.

In exception to the above rules, whenever the program finds the holy grail, which is to say, the
letters t-h-e-h-o-l-y-g-r-a-i-l (case insensitive) in that order in a line of input, then the first taunt
generated will be displayed by the program as "(A childish hand gesture)".

To ensure all royal quality assurance criteria are met, the program must be demonstrated by a
simulation showing the taunts produced from a series of inputs. Each taunt is generated by
applying the taunt generation rules until all of the <…> have been replaced with appropriate
words. In most cases, you will face a choice of alternate rules for expanding a phrase name. In
these cases, you should make a choice as follows: Suppose that this is the kth such choice that
you have faced for that rule since the start of program execution, and that you must choose one
of n rules for expanding a given kind of phrase. Let the rules for that phrase be numbered from
1…n in the order of appearance above, and then choose rule number **((k-1) mod n) + 1**.

Well, get on with it!

Input

The input will consist of an unspecified number of lines. Each line will contain a statement uttered by a knight consisting of letters, digits, the characters ",.-!?" and whitespace. Each line of input will be more than 1 character and less than 72 characters in length. All words will be separated by whitespace. Each statement will contain at least one word.

Output

For every line of input, print a block of output containing the following:

- A single line containing "Knight:", a space, and the input. Any appearance of whitespace inside the input will be replaced by a single space.
- All taunts (as explained in the above rules) prefaced by "Taunter:" and a space, the taunt, and a period. Each word should be separated from neighboring words by a single space.
- A blank line

Each taunt should begin with a capital letter, and no extra characters should be added.

Sample Input

```
Hello!
Are you feeling alright?
Is there someone else I could talk to?
Anyone at     all?
We seek the holy grail . . .
```

Sample Output

```
Knight: Hello!
Taunter: Your mother was a hamster.

Knight: Are you feeling alright?
Taunter: Coconut! Your steed is a silly duck.

Knight: Is there someone else I could talk to?
Taunter: Your father personified a herring.
Taunter: Your grandmother was a newt.
Taunter: Peril! Your king masquerades as a conspicuously wicked chicken.

Knight: Anyone at all?
Taunter: Your grandfather personified a vole.

Knight: We seek the holy grail . . .
Taunter: (A childish hand gesture).
Taunter: Your godfather was a parrot.
```

No Left Turns

ALL HEADS: You're a Knight of the Round Table?
ROBIN: I am.
LEFT HEAD: In that case I shall have to kill you.
MIDDLE HEAD: Shall I?
RIGHT HEAD: Oh, I don't think so.
MIDDLE HEAD: Well, what do I think?
LEFT HEAD: I think kill him.
RIGHT HEAD: Well let's be nice to him.
MIDDLE HEAD: Oh shut up.

As the story goes, the Knight scarpers off. Right Head has taken it upon himself to search the grounds for the knight so he, Left, and Middle can go extinguish him (and then have tea and biscuits.)

Consider the following 8 by 12 maze, where shaded squares are walls that can't be entered.
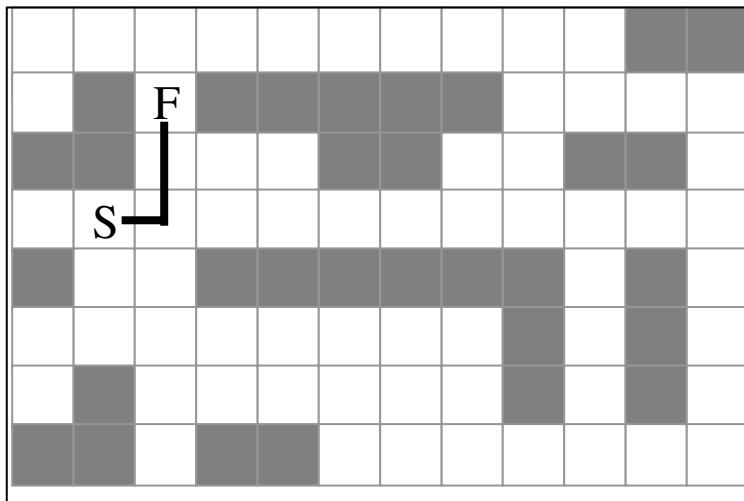
The shortest path between the Right Head (denoted by the S, for start) and the knight (denoted by the F, for finish) is of length 3, as illustrated above. But! Right Head can't turn left or make U-Turns. He can only move forward and turn right. That means the shortest path that Right Head can find is significantly longer: at 29!
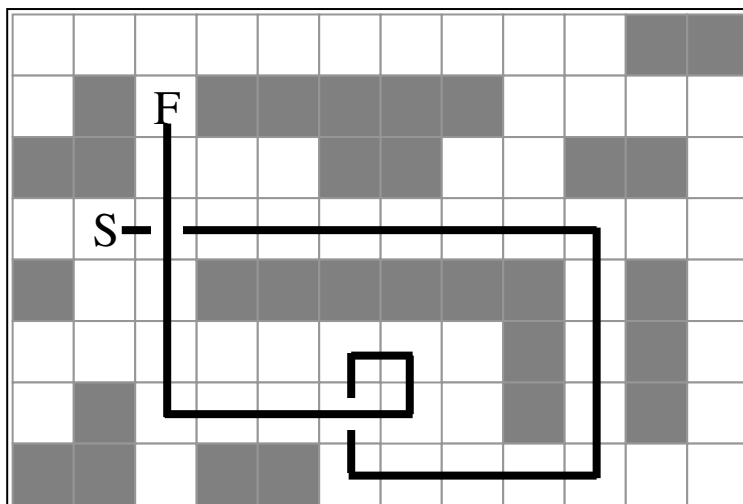
Input

The input file will consist of a single integer **N** (**N** > 0) specifying the number of mazes in the file. Following this, on a maze by maze basis will be the number of rows, **r** (3 < **r** ≤ 20), a space, then the number of columns, **c** (3 < **c** ≤ 20). After this will follow **r** lines of **c** characters, representing a map of the maze:

```
XXXXXXXXXXXXXX
X            XXX
X XFXXXXX      X
XXX    XX   XX X
X S             X
XX   XXXXXX X X
X          X X X
X X        X X X
XXX XX         X
XXXXXXXXXXXXXX
```

X's mark those locations that are walls and can't be occupied. **S** marks the start location, and **F** marks the Knight. Blanks are locations that can be freely traveled.

Output

The output is the length of the shortest path between the start and finish locations. Based on the above maze, your program would output the minimum no-left-turns path length of 29.

Additional Constraints/Information:
- Right Head is capable of moving from the start position in any of the four primary compass directions. After that, he's constrained to either step forward or right.
- The start and end locations will never be the same.
- The maze is always surrounded by four walls.
- You can assume that a path between the start and final locations always exists.

Sample Input	Sample Output
1 10 14 XXXXXXXXXXXXXX X XXX X XFXXXXX X XXX XX XX X X S X XX XXXXXX X X X X X X X X X X X XXX XX X XXXXXXXXXXXXXX	29

Coconuts

A group of **n** castle guards are voting to determine whether African swallows can carry coconuts. While each guard has his own personal opinion on the matter, a guard will often vote contrary to his beliefs in order to avoid disagreeing with the votes of his friends.

You are given a list of guards who either do or do not believe in the coconut-carrying capacity of African swallows, and a list of all pairs of guards who are friends. Your task is to determine how each guard must vote in order to minimize the sum of the total number of disagreements between friends and the total number of guards who must vote against their own beliefs.

Input

The input to this problem will contain multiple test cases. Each test case begins with a single line containing an integer **n** (where $2 \leq n \leq 300$), the number of guards, and an integer **m** (where $1 \leq m \leq n(n\text{-}1)/2$), the number of pairs of guards who are friends. The second line of the test case contains **n** integers, where the **i**th integer is 1 if the **i**th guard believes in the ability of African swallows to carry coconuts, and 0 otherwise. Finally, the next **m** lines of the test case each contain two distinct integers **i** and **j** (where $1 \leq i, j \leq n$), indicating that guards **i** and **j** are friends. Guards within each pair of friends may be listed in any order, but no pair of guards will be repeated. The input is terminated by an invalid test case with **n** = **m** = 0, which should not be processed.

Output

For each input test case, print a single line containing the minimum possible sum of the total number of disagreements between all friends plus the total number of guards who must vote against their own beliefs.

Sample Input	Sample Output
3 3 1 0 0 1 2 1 3 3 2 6 6 1 1 1 0 0 0 1 2 2 3 4 2 3 5 4 5 5 6 0 0	1 2

Notes:
In the first test case, the best result is achieved when all guards vote that African swallows cannot carry coconuts. Here, there is only a penalty of 1 for the first guard voting against his beliefs.

In the second test case, the best result is achieved when each guard votes for his beliefs. The penalty of 2 arises from the disagreements between guards 2 and 4, and guards 3 and 5.