2010 Mid-Atlantic Regional Programming Contest

Welcome to the 2010 Programming Contest. Before you start the contest, please be aware of the following notes:

The Contest

1. There are eight (8) problems in the packet, using letters A–H. These problems are NOT sorted by difficulty. As a team's solution is judged correct, the team will be awarded a balloon. The balloon colors are as follows:

Problem	Problem Name	Balloon Color
A	Palindometer	Yellow
B	Balloons	Green
C	Selling Cells	Silver
D	Not One Bit More	Black
E	Abstract Extract	Orange
F	Roller Coaster	Purple
G	Spy Cam	Pink
H	Underground Cables	Red

2. Solutions for problems submitted for judging are called runs. Each run will be judged.

   The judges will respond to your submission with one of the following responses. In the event that more than one response is applicable, the judges may respond with any of the applicable responses.

Response	Explanation
Correct	Your submission has been judged correct.
Wrong Answer	Your submission generated output that is not correct or is incomplete.
Output Format Error	Your submission's output is not in the correct format or is misspelled.
Excessive Output	Your submission generated output in addition to or instead of what is required.
Compilation Error	Your submission failed to compile.
Run-Time Error	Your submission experienced a run-time error.
Time-Limit Exceeded	Your submission did not solve the judges' test data within 30 seconds.

3. A team's score is based on the number of problems they solve and penalty points, which reflect the amount of time and number of incorrect submissions made before the problem is solved. For each problem solved correctly, penalty points are charged equal to the time at

which the problem was solved plus 20 minutes for each incorrect submission. No penalty points are added for problems that are never solved. Teams are ranked first by the number of problems solved and then by the fewest penalty points.

4. This problem set contains sample input and output for each problem. However, you may be assured that the judges will test your submission against several other more complex datasets, which will not be revealed until after the contest. Your major challenge is designing other input sets for yourself so that you may fully test your program before submitting your run. Should you receive a judgment stating that your submission was incorrect, you should consider what other datasets you could design to further evaluate your program.

5. In the event that you feel a problem statement is ambiguous or incorrect, you may request a clarification. Read the problem carefully before requesting a clarification. If the judges believe that the problem statement is sufficiently clear, you will receive the response, "The problem statement is sufficient; no clarification is necessary." If you receive this response, you should read the problem description more carefully. If you still feel there is an ambiguity, you will have to be more specific or descriptive of the ambiguity you have found. If the problem statement is ambiguous in specifying the correct output for a particular input, please include that input data in the clarification request.

   You may not submit clarification requests asking for the correct output for inputs that you provide. Sample inputs *may* be useful in explaining the nature of a perceived ambiguity, e.g., "There is no statement about the desired order of outputs. Given the input: ..., would both this: ... and this: ... be valid outputs?".

   If a clarification is issued during the contest, it will be broadcast to all teams.

6. Runs for each particular problem will be judged in the order they are received. However, it is possible that runs for different problems may be judged out of order. For example, you may submit a run for B followed by a run for C, but receive the response for C first.

   Do not request clarifications on when a response will be returned. If you have not received a response for a run within 30 minutes of submitting it, **you may have a runner ask the site judge to determine the cause of the delay. Under no circumstances should you ever submit a clarification request about a submission for which you have not received a judgment.**

   If, due to unforeseen circumstances, judging for one or more problems begins to lag more than 30 minutes behind submissions, a clarification announcement will be issued to all teams. This announcement will include a change to the 30 minute time period that teams are expected to wait before consulting the site judge.

7. The submission of abusive programs or clarification requests to the judges will be considered grounds for immediate disqualification.

Your Programs

8. All solutions must read from standard input and write to standard output. In C this is scanf/printf, in C++ this is cin/cout, and in Java this is System.in/System.out. The judges will ignore all output sent to standard error (cerr in C++ or System.err in Java). You may wish to use standard error to output debugging information. From your workstation you may test your program with an input file by redirecting input from a file:

   ```
   program < file.in
   ```

9. Every effort has been made to ensure that the compilers and run-time environments used by the judges are as similar as possible to those that you will use in developing your code. With that said, some differences may exist. It is, in general, your responsibility to write your code in a portable manner compliant with the rules and standards of the programming language. You should not rely upon undocumented and non-standard behaviors.

   One place where differences are likely to arise is in the size of the various numeric types. Many problems will specify minimum and maximum values for numeric inputs and outputs. You should write your code with the understanding that, on the *judges'* machines:

   - A C++ `int`, a C++ `long`, and a Java `int` are all 32-bits wide.
   - A C++ `long long` and a Java `long` are 64-bits wide.
   - A `float` in both languages is a 32-bit value capable of holding 6-7 decimal digits, though many library functions will be less accurate.
   - A `double` in both languages is a 64-bit value capable of holding 15-16 decimal digits, though many library functions will be less accurate.

   The data types on your own machines may differ in size from these, but if you follow the guidelines above in choosing the types to hold your numbers, you can be assured that they will suffice to hold those values on the judges' machines.

10. All lines of program input will end with the appropriate line terminator (e.g., a linefeed on Unix/Linux systems, a carriage return-linefeed pair on Windows systems).

11. All input sets used by the judges will follow the input format specification found in the problem description. You do not need to test for input that violates the input format specified in the problem.

12. Unless otherwise specified, all lines of program output

    - should be left justified, with no leading blank spaces prior to the first non-blank character on that line,
    - should end with the appropriate line terminator (`\n`, `endl`, or `println()`), and
    - should not contain any blank characters at the end of the line, between the final specified output and the line terminator.

You should not print extra lines of output, even if empty, that are not specifically required by the problem statement.

13. Unless otherwise specified, all numbers in your output should begin with the − if negative, followed immediately by 1 or more decimal digits. If it is a real number, then the decimal point should appear, followed by the appropriate number of decimal digits. For output of real numbers, the number of digits after the decimal point will be specified in the problem description (as the "precision").

    All real numbers printed to a given precision should be rounded to the nearest value. Rounding should be carried out so that trailing digits of 5 of higher are rounded up, tr4ailing digits of 4 or less are rounded down. For example, if a precision of 2 decimal digits is requested, then 0.0152 would round to 0.02, but 0.0149 would round to 0.01.

    In simpler terms, neither scientific notation nor commas will be used for numbers, and you should ensure you use a printing technique that rounds to the appropriate precision.

14. If a problem specifies that an input is a floating point number, the input will be presented according to the rules stipulated above for output of real numbers, except that decimal points and the following digits may be omitted for numbers with no non-zero decimal portion. Scientific notation will not be used in input sets unless a problem explicitly allows it.

Good luck, and HAVE FUN!!!

Problem A: Palindrometer

While driving the other day, John looked down at his odometer, and it read 100000. John was pretty excited about that. But, just one mile further, the odometer read 100001, and John was REALLY excited! You see, John loves palindromes — things that read the same way forwards and backwards. So, given any odometer reading, what is the least number of miles John must drive before the odometer reading is a palindrome? For John, every odometer digit counts. If the odometer reading was 000121, he wouldn't consider that a palindrome.

Input

There will be several test cases in the data file, each consisting of an odometer reading on its own line. Each odometer reading will be from 2 to 9 digits long. The odometer in question has the number of digits given in the input - so, if the input is 00456, the odometer has 5 digits. There will be no spaces in the input, and no blank lines between input sets.

The data file will end with a line with a single 0.

Output

For each test case, output the minimum number of miles John must drive before the odometer reading is a palindrome. This may be 0 if the number is already a palindrome. Output each integer on its own line.

Example

Input:

Given the input

```
100000
100001
000121
00456
0
```

the output would be

Output:

```
1
0
979
44
```

Problem B: Balloons

As you know, balloons are handed out during this contest to teams as they solve problems. However, in the past this has sometimes presented challenging logistical problems.

One contest hosting site maintained two rooms, A and B, each containing a supply of balloons. There were N teams attending the contest, each sitting in different locations, some being closer to room A, and others to room B. Given the number of balloons needed by each team and each teams distance from room A and room B, what is the minimum total possible distance that must be traveled by all balloons as they are delivered to their respective teams, assuming they are allocated in an optimal fashion from rooms A and B? For the purposes of this problem, assume that the contest staff were cheap and only bought one color of balloon.

Input

There will be several test cases in the data file. Each test case will begin with a line with three integers:

$$N\ A\ B$$

where N is the number of teams ($1 \leq N \leq 1,000$), and A and B are the number of balloons in rooms A and B, respectively ($0 \leq A, B \leq 10,000$).

On each of the next N lines there will be three integers, representing information for each team:

$$K\ D_A\ D_B$$

where K is the total number of balloons that this team will need, D_A is the distance of this team from room A, and D_B is this teams distance from room B ($0 \leq D_A, D_B \leq 1,000$). You may assume that there are enough balloons - that is, $\sum_i K_i \leq A + B$. The data file will end with a line with three 0s.

Output

For each test case, output a single integer, representing the minimum total distance that must be traveled to deliver all of the balloons. Count only the outbound trip, from A or B to the team. Dont count the distance that a runner must travel to return to room A or room B. Print each integer on its own line with no spaces. Do not print any blank lines between answers.

Example

Input:

Given the input

```
3 15 35
10 20 10
10 10 30
10 40 10
0 0 0
```

the output would be

Output:

```
300
```

Problem C: Selling Cells

Competition has been fierce among the cellphone providers in the town of Eastern WestField. Several companies have been running ads in which they each claim to provide more coverage to the town than do any of their competitors.

AetherTech Telecommunications suspects that they, in fact, really do cover a larger portion of the town than do their competitors. They would like to compute their actual coverage so thay they can advertise the true amount.

The metropolitan area of Eastern WestField will be modeled as a circle around the town center. The area covered by each cell tower can also be modeled as a circle, centered on the location of the tower, indicating the area within which a the tower provides sufficient signal strength for a phone to connect (e.g., one bar on the average phone's signal strength meter).

All cell towers to be considered will have their centers within the metropolitan area, though their area of coverage may extend beyond the border of the metropolitan area. The areas covered by different towers can overlap, though for economic reasons no two towers have been constructed so close to one another that the center of one would lie within the coverage area of the other.

Compute the fraction of the metropolitan area within which a cell phone would be covered by at least one tower.

Input

Input consists of multiple data sets. Each data set begins with a line containing a single integer N, $1 \leq N \leq 25$. This line is followed by N lines, each containing three floating point numbers x, y, r. These give the (x,y) coordinates of the circle's center and the radius of that circle, respectively.

The first circle in the list denotes the metropolitan area of Eastern Westfield. Each of the remaining N-1 circles describes a cell tower.

The final data set is followed by a line containing a zero.

Output

For each data set, compute the fraction of the metropolitan area covered by the cell towers (as a number in the range $0.00 \ldots 1.00$ and print a line containing that fraction as a real number presented to 2 decimal points precision.

Example

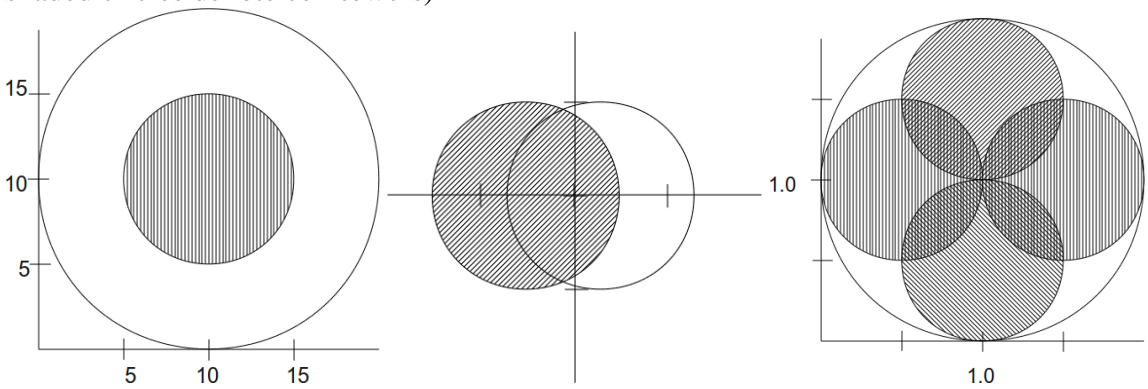
Input:

Given the input

```
2
10 10 10
10 10 5
2
0.404 0.0 1.0
```

```
-0.404 0.0 1.0
5
0 0 2
1 0 1
0 1 1
-1 0 1
0 -1 1
0
```

(corresponding to the following three pictures, where the empty circle denotes the city and the shaded circles denote cell towers)



the output would be

Output:

```
0.25
0.50
0.82
```

Problem D: Not One Bit More

Start with an integer, N_0, which is greater than 0. Let N_1 be the number of ones in the binary representation of N_0. So, if $N_0 = 27$, $N_1 = 4$.

In general, let N_i be the number of ones in the binary representation of N_{i-1}. This sequence will always converge to one.

For any starting number, N_0, let $K(N_0)$ be the minimum i such that N_i is one. For example, if $N_0 = 31$, then $N_1 = 5$, $N_2 = 2$, $N_3 = 1$, so $K(31) = 3$.

Given a range of consecutive numbers, and a value X, how many numbers in the range have a $K(\ldots)$ value equal to X?

Input

There will be several test cases in the data file. Each test case will consist of three integers on a single line:

$$\text{LO HI } X$$

where LO and HI ($1 \leq \text{LO} \leq \text{HI} \leq 10^{18}$) are the lower and upper limits of a range of integers, and X ($0 \leq X \leq 10$) is the target value for $K(\ldots)$.

The data file will end with a line with three 0s.

Output

For each test case, output a line with a single integer, representing the number of integers in the range from LO to HI (inclusive) which have a $K(\ldots)$ value equal to X in the input.

Example

Input:

Given the input

```
31 31 3
31 31 1
27 31 1
27 31 2
1 4 1
1023 1025 1
1023 1025 2
0 0 0
```

the output would be

Output:

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

Problem E: Abstract Extract

When writing articles, there is usually an abstract section which summarizes the entire article. We are experimenting with an automatic abstract generation algorithm. The algorithm reads in an article and prepares an abstract that summarizes the entire article. The abstract is formed by combining the "topic sentences" extracted from consecutive paragraphs.

For the purposes of this problem,

- An *article* consists of one or more paragraphs.
- A *paragraph* is a maximal sequence of non-empty lines.
- A *sentence* is a maximal sequence of characters within a paragraph that begins with a non-whitespace character, that ends with a '.' (period), '?', or '!', and that does not contain any other occurrences of '.', '?', or '!'.
- A *word* is a maximal sequence of alphabetic characters within a sentence.

The term *maximal* in the definitions above is intended to convey the idea that we are only interested in the longest sequences that match the definition, not in any of their subsequences. For example, the sentence "How now, brown cow?" contains four words. "now" is a word in this sentence, but "no", "ow", etc., are not words because they are not maximal – they are subsequences of a larger sequence of alphabetic characters.

A *topic sentence* for a paragraph is the single sentence in the paragraph that best describes the paragraph's content. For our purposes, we will select the earliest sentence in the paragraph that maximizes the number of distinct words in S that also occur in any following sentence within the same paragraph.

Paragraphs with fewer than three sentences are ignored and will not contribute to the abstract.

When comparing words for distinctness, changes in upper/lower case are ignored. For example, the sentence "See what I see." contains three distinct words, not four.

Input

Input will consist of one or more articles. Each article is terminated by a line containing only "***" or "******". The latter string ("******") indicates the end of the entire input set.

- Each article will contain one or more paragraphs. Each paragraph will consist of one or more non-empty lines, and is terminated by an empty line or by the "***" or "******" markers described above.
- No article will be longer than 500 lines; No line will contain more than 150 characters. No word will contain more than 50 characters.
- The only whitespace characters in the input will be blanks (ASCII 32) and line terminators.

Output

For each document, print the abstract followed by a line containing "======" (six equal signs).

Each abstract will be formed from the sequence of topic sentences, selected as described above, in the order that they occur in the input document. Each sentence shall be printed exactly as it appears in the input and should be followed by a line break.

Example

Input:

Given the input

```
Hello world!  Everyone should greet the entire world in a friendly
manner. Wouldn't that make the world a more friendly place?

This is not a sentence
***
From: The Wright Brothers' Aeroplane.

By Orville and Wilbur Wright

In the field of aviation there were two schools. The first,
represented by such men as Professor Langley and Sir Hiram Maxim,
gave chief attention to power flight; the second, represented by
Lilienthal, Mouillard, and Chanute, to soaring flight. Our sympathies
were with the latter school, partly from impatience at the wasteful
extravagance of mounting delicate and costly machinery on wings
which no one knew how to manage, and partly, no doubt, from the
extraordinary charm and enthusiasm with which the apostles of soaring
flight set forth the beauties of sailing through the air on fixed
wings, deriving the motive power from the wind itself.

We began our active experiments at the close of this period, in October,
1900, at Kitty Hawk, North Carolina. Our machine was designed to be
flown as a kite, with a man on board, in winds from 15 to 20 miles an
hour. But, upon trial, it was found that much stronger winds were
required to lift it. Suitable winds not being plentiful, we found it
necessary, in order to test the new balancing system, to fly the machine
as a kite without a man on board, operating the levers through cords
from the ground. This did not give the practice anticipated, but it
inspired confidence in the new system of balance.

In the summer of 1901 we became personally acquainted with Mr. Chanute.
When he learned that we were interested in flying as a sport, and not
with any expectation of recovering the money we were expending on it, he
gave us much encouragement. At our invitation, he spent several weeks
```

```
with us at our camp at Kill Devil Hill, four miles south of Kitty Hawk,
during our experiments of that and the two succeeding years. He also
witnessed one flight of the power machine near Dayton, Ohio, in October,
1904.
******
```

the output would be

Output:

```
Everyone should greet the entire world in a friendly
manner.
======
The first,
represented by such men as Professor Langley and Sir Hiram Maxim,
gave chief attention to power flight; the second, represented by
Lilienthal, Mouillard, and Chanute, to soaring flight.
Our machine was designed to be
flown as a kite, with a man on board, in winds from 15 to 20 miles an
hour.
When he learned that we were interested in flying as a sport, and not
with any expectation of recovering the money we were expending on it, he
gave us much encouragement.
======
```

Problem F: Roller Coaster

Bessie has gone on a trip, and shes riding a roller coaster! Bessie really likes riding the roller coaster, but unfortunately she often gets dizzy.

The roller coaster has a number of distinct sections that Bessie rides in order. At the beginning of the ride, Bessie's dizziness and fun levels are both at 0. For each section of the roller coaster, Bessie can either keep her eyes open or keep them closed (and must keep them that way for the whole section). If she keeps her eyes open for a section, her total fun increases by a Fun factor for that section, and her dizziness increases by a Dizziness factor for that section. However, if she keeps her eyes closed for the section, her total fun will not change, but her dizziness will decrease by a value thats constant for the entire roller coaster. (Note that her dizziness can never go below 0.)

If at any point, Bessie's dizziness is above a certain limit, Bessie will get sick. Write a program to find the maximum amount of fun Bessie can have without getting sick.

Input

There will be several test cases in the data file. Each test case will begin with a line with three integers:

$$N\ K\ L$$

where N ($1 \leq N \leq 1,000$) is the number of sections in this particular the roller coaster; K ($1 \leq K \leq 500$) is the amount that Bessies dizziness level will go down if she keeps her eyes closed on any section of the ride; and L ($1 \leq L \leq 300,000$) is the limit of dizziness that Bessie can tolerate – if her dizziness ever becomes larger than L, Bessie will get sick, and thats not fun!

Each of the next N lines will have two integers:

$$F\ D$$

where F ($1 \leq F \leq 20$) is the increase to Bessies total fun that shell get if she keeps her eyes open on that section, and D ($1 \leq D \leq 500$) is the increase to her dizziness level if she keeps her eyes open on that section. The sections will be listed in order.

The data file will end with a line with three 0s.

Output

For each test case, output line containing a single integer, representing the maximum amount of fun Bessie can have on that roller coaster without exceeding her dizziness limit.

Example

Input:

Given the input

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

the output would be

Output:

```
7
3
```

Problem G: Spy Cam

An overhead camera has snapped a photo of the papers on a bureaucrat's desk. Many of the papers overlap, but, because the bureaucrat in questions is a bit of a neatness freak, all papers are aligned with their edges parallel to the edges of the desk. As a preliminary step in analyzing these papers, other programs have used cues of paper color, edges, etc., to label each pixel in the image according to which sheet of paper is visible in that particular location. The result will look something like this

```
...aaaaaaaa.dd
...aaaaaaaa.ee
.ccaaaaaaa...
.ccaaaabbaa...
.ccaaaabbaa...
.ccaaaabbaa...
.ccaaaaaaa...
.............
```

A '.' denotes the desktop, and alphabetic labels 'a', 'b', 'c', etc., denote distinct pieces of paper. These labels are assigned in an arbitrary order but are dense (no letters are "skipped" during the labeling).

Assume that each paper is rectangular, and that no important information is being lost "between" the pixels. Assume also that the camera has recorded the entire desktop and that there are no papers that are completely hidden from view.

For each piece of paper (ordered by label), determine if the visual evidence proves that the entire sheet of paper is visible or if it is at least possible that a portion of the paper is hidden beneath another sheet.

Input

Input will consist of several snapshots.

Each snapshot begins with a line containing two integers, R and C, denoting the number of rows and columns of pixels in the snapshot. For each snapshot, $1 \leq R, C \leq 40$.

The R and C values are followed by R lines, each containing C characters. Those characters will be lower-case alphabetic characters or a period ('.'). As noted earlier, the alphabetic characters denote visible portions of distinct sheets of paper and the periods denote portions of the underlying desktop.

All snapshots in the input will correspond to a valid arrangement of rectangular sheets of paper. There will be no impossible arrangements.

The final snapshot will be followed by a line containing two zeros separated by a space.

Output

For each snapshot, print one line of output. That line will begin with the phrase "Uncovered:" followed by a single space. Then, on the same line, list the papers that are definitely completely visible, in order of character code, with no separating spaces.

Example

Input:

Given the input

```
3 3
..b
.cc
dca
8 14
...aaaaaaaa.dd
...aaaaaaaa..e
.ccaaaaaaaa...
.ccaagabbaa...
.ccafgffffa...
.ccaagabbaa...
.ccaaaaaaaa...
.cc..........
0 0
```

the output would be

Output:

```
Uncovered: a
Uncovered: cdg
```

Problem H: Underground Cables

A city wants to get rid of their unsightly power poles by moving their power cables underground. They have a list of points that all need to be connected, but they have some limitations. Their tunneling equipment can only move in straight lines between points, and they only have room for one underground cable at any location except at the given points so no two cables can cross.

Given a list of points, what is the least amount of cable necessary to make sure that every pair or points is connected, either directly, or indirectly through other points?

Input

There will be several test cases in the data file. Each test case will begin with an integer N ($2 \leq N \leq 1,000$), which is the number of points in the city. On each of the next N lines will be two integers, X and Y ($-1,000 \leq X, Y \leq 1,000$), which are the (X, Y) locations of the N points.

The data file will end with a line with a single 0.

Output

For each test case, output a line containing single real number, representing the least amount of cable the city will need to connect all of its points. Print this number with to two decimal places precision.

Example

Input:

Given the input

```
4
0 0
0 10
10 0
10 10
2
0 0
10 10
0
```

the output would be

Output:

```
30.00
14.14
```