2015 Mid-Atlantic Regional Programming Contest

Welcome to the 2015 ICPC Mid-Atlantic Regional. Before you start the contest, please take the time to review the following:

The Contest

1. There are eight (8) problems in the packet, labeled 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	Positive Con Sequences	Orange
B	Refract Facts	Green
C	Hounded by Indecision	Silver
D	Avoiding an Arrrgument	Pink
E	Kinfolk	Red
F	Of the Children	Yellow
G	Talking About Numbers	Black
H	The Scheming Gardener	Purple

2. The winning team is the one that successfully completes the most problems in the time allowed.

   If teams are tied with the same number of problems solved, the tie is broken in favor of the team with the fewest penalty points. For each problem *solved correctly*, penalty points are charged as the sum of

   - the number of minutes elapsed since the start of the contest to when the successful submission was made, and
   - 20 points for each incorrect submission prior to the successful one.

   No penalty points are added for problems that are never solved.

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

   If the judges believe that the problem statement is sufficiently clear, you will receive the response, "No response, read problem statement." 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 may have found.

Submitting

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

5. The judges will respond to your submission with one of the following responses.

Response	Explanation
Yes	Your submission has been judged correct.
Wrong Answer	Your submission generated output that is not correct.
Output Format Error	Your submission's output is not in the correct format or is misspelled.
Incomplete Output	Your submission did not produce all of the required output.
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.
Other-Contact Staff	Contact your local site judge for clarification.

6. In the event that more than one response is applicable, the judges may respond with any of the applicable responses. For example, a program that runs too long but produces incorrect output before it is killed might receive either a "Wrong Answer" or a "Time-Limit Exceeded" response. A program that crashes before completing the test data set might receive either an "Incomplete Output" or a "Run-Time Error" response.

7. **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 local site judge to determine the cause of the delay.**

   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.

8. The submission of code deliberately designed to delay, crash, or otherwise negatively affect the contest itself will be considered grounds for immediate disqualification.

Your Programs

9. Your program must be contained within a single source-code file. Java programs should be written in the default (unnamed) package, meaning that it should not contain a `package` statement at all.

   Use the filename extension "`.cpp`" for C++ program files. Use the extension "`.c`" for C program files. Use the extension "`.java`" for Java program files.

Note that all filename extensions are lower case.

10. Your code will be compiled for judging as follows:

    C: `gcc -O2 -std=gnu99 -static` *yourFileName* `-lm`

    C++: `g++ -O2 -std=c++11 -static` *yourFileName*

    Java: `javac -encoding UTF-8 -sourcepath .` *yourFileName*

    For Java, the compiled code will be executed using the command:

    `java -Xss8m -Xmx1024m` *yourClassName*

11. 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`.

12. Unless otherwise specified, all lines of program output

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

    You must 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 minus sign (−) if negative, followed immediately by 1 or more decimal digits. If the number being printed is a floating point 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 "*decimal digits of precision*").

    All floating point numbers printed to a given precision should be rounded to the nearest value. For example, if 2 decimal digits of precision is requested, then 0.0152 would be printed as "0.02" but 0.0149 would be printed as "0.01".

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

14. 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.

15. 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).

16. 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.

17. 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.

Good luck, and HAVE FUN!!!

Problem A: Positive Con Sequences

Your younger sister is studying for an upcoming standardized test in mathematics, and needs practice with the common style of problem in which the student is presented with a sequence of numbers with one number missing and asked to fill in the missing value.

You are aware that the vast majority of these problems feature either arithmetic sequences (where each number in the sequence is formed by adding an integer constant to the prior number) or geometric sequences (where each number in the sequence is formed by multiplying the prior number by an integer constant).

Write a program that will help your sister drill on this style of problem by allowing her to check her answers on sample problems.

Input

Input will consist of one or more datasets.

Each dataset will be a single line containing 4 integers defining a sequence. One of these will be -1, denoting the missing value. The remainder will be positive integers in the range $1 \ldots 10{,}000$, inclusive. Other than the -1 placeholder value, the values will be in strictly increasing order.

End of input will be signaled by a line containing four -1 values.

Output

For each dataset, print one line of output.

If a positive integer in the range $1 \ldots 1{,}000{,}000$ exists that can be filled in to the missing value position to create an arithmetic or geometric sequence, print that missing value.

If there is no such positive integer, print -1.

Example

Given the input:

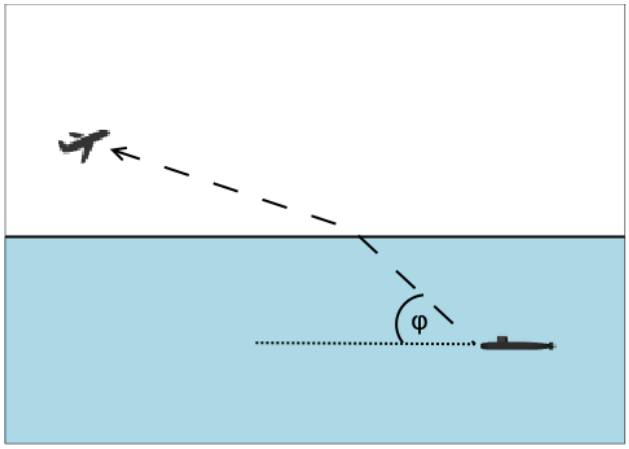
```
1 2 -1 4
2 4 8 -1
7 8 -1 21
5 -1 11 14
-1 2 4 6
-1 -1 -1 -1
```

the output should be:

```
3
16
-1
8
-1
```

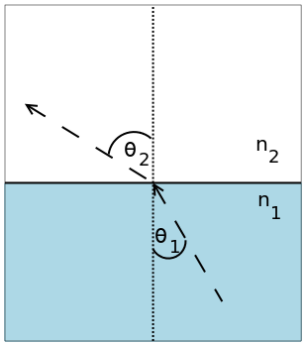
Problem B: Refract Facts

A submarine is using a communications laser to send a message to a jet cruising overhead. The sea surface is flat. The submarine is cruising at a depth d below the surface. The jet is at height h above the sea surface, and a horizontal distance x from the sub. The submarine turns toward the jet before starting communications, but needs to know the angle of elevation, ϕ, at which to aim the laser.



When the laser passes from the sea into the air, it is refracted (its path is bent). The refraction is described by Snell's law, which says that light approaching the horizontal surface at an angle θ_1, measured from the vertical, will leave at an angle θ_2, given by the formula

$$\frac{\sin\theta_1}{\sin\theta_2} = \frac{n_1}{n_2}$$



where n_1 and n_2 are the respective *refraction indices* of the water and air. (The refraction index of a material is inversely proportional to how fast light can travel through that material.)

Input

Input will consist of one or more datasets.

Each dataset will consist of one line with 5 floating point numbers. In order:

- d, the depth of the submarine (specifically, of the laser emitter) in feet, $1 \leq d \leq 800$
- h, the height of the plane in feet, $100 \leq h \leq 10{,}000$
- x, the horizontal distance from the sub to the plane in feet, $0 \leq x \leq 10{,}000$
- n_1, the refractive index of water, $1.0 < n_1 \leq 2.5$,
- n_2, the refractive index of air, $1.0 \leq n_2 < n_1$

A value of 0 for d will signal the end of input.

Output

For each dataset, print a single line containing the angle of elevation in degrees, to two decimals precision, at which the submarine should aim its laser to illuminate the jet.

Example

Given the input:

```
600 600 1000 1.333 1.01
600 1200 4000 1.5 1.01
0 0 0 0 0
```

the output should be:

```
44.37
11.51
```

Problem C: Hounded by Indecision

OK, maybe stealing the Duchess's favorite ruby necklace was not such a good idea. You were making your way toward the city gates when you heard the sound you had been dreading: a sharp whistle followed by an answering bark. You know that the constable has just fetched his favorite hound and is starting to search for you. They might head straight for a gate. They might try to pick up your trail on the way. You really can't guess. But if they reach the gate before you, you're caught. If they happen across your trail, the hound will pick up your scent. The dog knows your scent already - this isn't your first offense! The constable will loose the hound, who can run *fast* once he has the trail to follow.

You have a dilemma. If you are absolutely sure that you can reach the gates before the guard and before being overtaken by the hound, you can keep the necklace. But if you aren't sure, you need to drop the necklace right now into the nearest pile of rubbish and saunter casually away. Even if they grab you, without the necklace in your hands they will eventually release you.

So, keep the necklace or drop the necklace?

The town is modeled as a rectangular maze of discrete squares. It is surrounded by a wall that contains one or more exits. You know, of course, your own position within the town. You also know the location of the kennel where the constable and the hound start out.

- In each turn (unit of time), you, the constable, and the hound move simultaneously.
- You can move zero or one square(s) horizontally or vertically per turn.
- Initially, the constable and hound move together, also zero or one square(s) vertically or horizontally.
- If the constable and the hound, moving together, reach a square that you have previously occupied, the hound catches your scent and the constable looses the hound. On each subsequent turn, the hound follows your trail at a speed of one or two squares per turn. (The hound moves two squares unless doing so would cause it to jump over the thief or the exit.)
- If the constable and/or the hound overtake you (occupy the same square as you), you are caught. To escape, you must reach an exit at least one turn before the constable and/or hound.

Input

Input consists of one or more mazes. Each maze begins with a line containing two integers, W and H, $3 \leq W, H \leq 40$, denoting the width and the height of the maze. End of input is indicated when either of these values is less than 3.

This is followed by H lines of input, each containing W characters.

The interpretation of the characters in these lines is as follows:

- ' ' denotes an open space

- 'K' is an open space denoting the kennel and hence the starting position of the constable and the hound. There will be exactly one of these in any maze.
- 'T' is an open space denoting the original position of the thief (you). There will be exactly one of these in any maze.
- 'X' denotes a wall.
- 'E' is an open space representing an exit (a city gate). There will be at least one of these.

  All exits will occur on the outer perimeter (as defined by the W and H values) of the maze.

All mazes will be completely enclosed by a combination of 'X' and 'E' characters. There will be a path from the thief's starting location to each exit and from the kennel to each exit.

Output

For each maze, print a single line of output. If there is a path that you can take that will guarantee that you can escape no matter what path the constable and hound take, then print "KEEP IT". If there is no path that offers such a guarantee, print "DROP IT".

Example

Given the input:

```
19 11
XXXXXXXXXXXXXXXXXX
X                 X
E                 X
X   XXX XXX  K    X
X   X     X       X
X   X     X       X
X   X  T  X       X
X   X     X       X
X   XXXXXXX       X
X                 X
XXXXXXXXXXXXXXEXXXX
19 11
XXXXXXXXXXXXXXXXXX
X                 X
E                 X
X   XXX XXX  K    X
X   X     X       X
X   X     X       X
X   X  T  X       X
X         X       X
X   XXXXXXX       X
X                 X
XXXXXXXXXXXXXXEXXXX
0 0
```

the output should be:

```
DROP IT
KEEP IT
```

In both cases, we can pretty much ignore the exit at the bottom of the maze. The constable can always get there before the thief.

In the first case, if the thief heads straight for the exit on the left, he emerges from the "door" on turn 4, and reaches the exit on turn 11. On turn 7, the constable can reach the space where the thief had been on turn 4, and would then loose the hound, which also reaches the exit on turn 11, catching the thief.

In the second case, the thief has the option of taking the "side door" out of the house, then heading for the wall before turning right and heading for the exit. It's actually a longer path, taking 13 turns to reach the exit. But the constable won't reach the exit before turn 14, and the earliest that the constable could pick up the thief's trail would be on turn 11 (at the thief's staring location), at which point the constable and hound are too far away to catch the thief.

Problem D: Avoiding an Arrrgument

The pirate captain and his crew gazed happily upon the chest brimming with gemstones. Diamonds, rubies, sapphires, opals, and many other kinds of gems added to the sparkling collection.

"We're rich, my lads!" the captain said, "Now, here's what we're going to do. Each of us will take two stones now, then we'll bury the rest because ... because that's what pirates do! Also, everyone must take two different kinds of gems. We don't know what the merchants in the next port will be buying, and I don't want to listen to any belly-aching from someone who only took rubies if our next port happens to have a flourishing ruby mine nearby"

"Who chooses first?" asked the cabin boy, fearing that he already knew the answer.

"I'll choose one first," said the captain, "and then each man from there in order of rank, starting with the first mate down to the cabin boy". The lower-ranked crew members began to grumble. "Then," continued the captain, "the cabin boy immediately chooses his second gem, and we proceed in reversed order until we get to me, and I will be the last to choose my second stone."

The crew quickly agreed and, almost as quickly, began trying to figure how to claim the best stones under this arrangement.

Write a program to figure out what stone a crewman should pick first in order to maximize the total value they can be guaranteed to acquire no matter how the rest of the crew chooses.

Input

The input will consist of one or more data sets.

Each data set starts with a line containing 2 integers: M, the number of kinds of gemstone available, and N, the number of crewmen who select their first stone after you and select their second before you. Each data set will have $2 \leq M \leq 26$, $0 \leq N \leq 10$. A value of 0 for M indicates the end of input.

The remainder of the dataset consists of M lines, each describing one kind of available gem. The line begins with an uppercase alphabetic character indicating the type of gem (e.g., 'D' might denote diamonds, 'Z' might stand for zircons). No two kinds of gem will have the same code, but the codes may occur in any order. The alphabetic code is followed by $N + 1$ integers, each demoting the value in doubloons of one of the $N + 1$ most valuable remaining gems of that kind. These will be presented on the line in non-increasing order of value. Values will be in the range $0 \ldots 10,000$.

Output

For each dataset, print a single line containing the letter code for the type of gem that you should pick first to maximize the total value that you can be guaranteed to receive on both picks, no matter what choices the rest of the crew makes. If there is more than one type of gem tied for the maximum total value, print the one the comes earliest alphabetically.

Example

Given the input:

```
4 3
D 10000 9000 8000 6000
R 5000 500 50 5
S 500 500 50 50
Z 1 0 0 0
2 1
S 6000 5000
D 5000 4000
0 0
```

the output should be

```
R
D
```

Problem E: Kinfolk

The English language abounds with terms for describing family (genetic) relationships.
The basic relationships are:

- "Parent" and "child" are well understood.
- Children of the same parent are "siblings".
- The child of your sibling is your niece (if she is female) or nephew (if he is male). You would be their aunt (if you are female) or uncle (if you are male).
- Two people who share a common grandparent but not a common parent are "1st cousins". If they share a common great-grandparent but not a common grandparent, they are "2nd cousins". This can be extended to "3rd cousins", "4th cousins", and so on.

These relationships are extended to later generations as follows:

- The "child", "niece", and "nephew" relationships can be extended to later generations by pre-pending "grand", "great-grand", or "great-great-grand". Thus the child of one's child is a grandchild. The male child of your niece or nephew is your grandnephew. You grand-nephew's female child would be your great-grandniece, and so on. (In theory, we could extend this to any number of additional "great-" prefixes, but we will stop with "great-great-grand" in this problem.)
- The "parent", "aunt", and "uncle" relationships are extended symmetrically by the same prefix. Thus you would be the grandparent of your grandchild, the great-granduncle or great-grandaunt of your great-grandnieces and great-grandnephews, etc.
- The "cousin" relationship is extended to your cousin's descendants by degrees of removal. The children of your 1st cousin are your first cousins once removed (and, symmetrically, you are their 1st cousin once removed). The grand-children of your 3rd cousin are your 3rd cousins twice removed.The great-grandchildren of your 2nd cousin are your 2nd cousins thrice removed. All of the cousin-based relationships are symmetric, so if someone is your K^{th} cousin *something* removed, you are theirs as well.

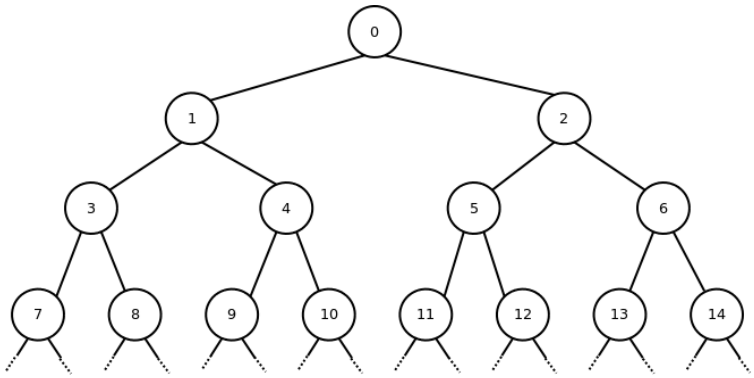
Write a program to determine the relationship of one person to another.

Input

Input will consist of one or more datasets. Each dataset consists of a single line containing two non-equal integers in the range $0 \ldots 32{,}767$ and a character. A negative number for the first integer indicates end of input.

The integers identify persons A and B. The character will be either 'M' or 'F', designating the gender of person B as male or female.

The integers identify the positions of person A and person B in a family tree envisioned as follows: consider a full binary tree in which the root is numbered 0, its children are numbered 1 and 2, and numbering proceeds in that manner, level by level, left to right. This numbering scheme is shown in the diagram to the right. A parent-child relationship in this tree represents a parent-child relationship in the family.



Output

For each dataset, print a single line indicating the relationship of B to A. This relationship must be constructed from the phrases "child", "parent", "niece", "nephew", "aunt", "uncle", "cousin", "grand", "great-", "1st", "2nd", "3rd", "once removed", "twice removed", and "thrice removed".

- No more than two "great-" prefixes may be applied.
- If "1st", "2nd", or "3rd" is used, it should be separated from the following part of the line by a single blank.
- If "once removed", "twice removed", or "thrice removed" is used, it must be separated from the preceding part of the line by a single blank.

If it is not possible to describe the relationship of B to A under the above limitations, then print "kin".

Example

Given the input:

```
1 5 M
1 11 F
0 8 F
5 7 M
0 32767 F
-1 -1 M
```

the output should be:

```
nephew
grandniece
great-grandchild
1st cousin once removed
kin
```

Problem F: Of the Children

This problem was withdrawn from the contest by a minute-1 clarification message sent to all teams.

The reason for the withdrawal was that, as of 24 hours before the contest, the authoring team had only a single sample solution, and we strongly prefer to move forward with a problem only after independent confirmation by two or more authors.

A nation-wide charity, *Won't Someone Think of the Children*, is sending one of its celebrity spokespeople out on a coast-to-coast publicity tour. Like many charities, they are bit strapped for funding and need to plan carefully to make sure they can pay the celebrity's travel expenses to get from their office on the west coast to the final press event on the east coast.

Local offices of the charity have been taking pledges and collecting donations to pay for this celebrity to visit their cities as part of the tour. The national office has made no promises that the celebrity will visit any of these intermediate locations, but hopes that these locally collected funds can actually help fund the coast-to-coast trip. In fact, without the help of the local offices, they aren't sure they can afford to get their spokesperson to the final destination.

The trip will be paid for on an incremental basis. The celebrity can only travel from one city to another if he or she has enough money to pay for the transportation to that next city. Upon arrival at the city, he or she can collect any money held there and use it to help pay for the later legs of the journey. The celebrity can only visit a given city once, lest multiple visits be considered an abuse of their hospitality.

Given a list of cities that have invited the celebrity, the amount of money raised by each city, and the travel costs between various pairs of cities, what is the smallest amount of money that the home office needs to provide the celebrity at the beginning of the journey to make sure that he or she can make it all the way to the end without being unable to pay for any leg of the trip?

Input

Input will consist of one or more datasets. Each dataset begins with a line containing a single integer, N, $2 \leq N < 24$, indicating the number of cities involved. A value of zero for N signals the end of input.

Cities are identified by integers in the range $0 \ldots N-1$, with city 0 being the home office and starting point of the journey, and city $N-1$ being the final destination city for the journey.

The first line of the dataset is followed by $N-2$ lines, each containing an integer in the range $0 \ldots 10{,}000$ indicating the amount of money collected by the cities numbered $1 \ldots N-2$. (The amount of money at city $N-1$ is irrelevant and the amount of money to be provided at city 0 is what you need to compute).

This is followed by at least 1 and up to $N(N-1)/2$ lines, each containing three integers i, j, & c. i and j are distinct integers in the range $0 \ldots N-1$ identifying two cities and c is the cost to travel from one of those cities to the other, $0 < c < 10{,}000$. The cost is the same when traveling from i to j as it is when traveling from j to i. The end of this list of potential travel expenses is indicated by a line containing negative values for i, j, and c.

Output

For each dataset print a single line of output.

If it is possible to reach city $N-1$ from city 0, print the smallest amount of money that the celebrity needs to be given at city 0 in order to guarantee reaching city $N-1$.

If it is not possible to reach city $N-1$ when starting from city 0, print -1.

Example

Given the input:

```
4
4
12
0 1 5
0 2 10
2 3 8
2 1 6
-1 -1 -1
4
4
12
0 1 5
0 2 10
2 3 8
2 1 6
0 3 6
-1 -1 -1
0
```

the output should be:

```
7
6
```

Problem G: Talking About Numbers

The programmers of a TTS (text-to-speech) program have discovered that their product does a poor job of reading numbers written in conventional numeric form. Currently, it just announces each digit in the number, one after the other, which gets confusing to the listener after a few digits have gone by.

They would prefer to have a more natural reading of numbers, and one of them has suggested that, if they convert numeric strings into the appropriate text equivalent, e.g., convert "1023" into "one thousand and twenty three", then their speech engine will be able to handle the reading with no further modification.

Write a program to carry out the transformation of non-negative integers into conventional English wording:

- English has unique names for the numbers 0-19: "zero", "one", "two", "three", "four", "five", "six", "seven", "eight", "nine", "ten", "eleven", "twelve", "thirteen", "fourteen", "fifteen", "sixteen", "seventeen", "eighteen", "nineteen".
- The subsequent multiples of 10 are named "twenty", "thirty", "forty", "fifty", "sixty", "seventy", "eighty", "ninety".
- The combination of one of those multiples of ten with a digit 1-9 is always hyphenated: e.g., $31 \Rightarrow$ "thirty-one", $77 \Rightarrow$ "seventy-seven".
- Multiples of 100 are counted 1-9 and set off from any following non-zero digits by 'and': e.g., $200 \Rightarrow$ "two hundred", $412 \Rightarrow$ "four hundred and twelve", $777 \Rightarrow$ "seven hundred and seventy-seven".
- Thousands and millions are counted off using the above rules to form numbers 1-999, and are set off from any non-zero remainder by a comma: e.g., $1{,}253{,}101 \Rightarrow$ "one million, two hundred and fifty-three thousand, one hundred and one".
- If a number with a non-empty thousands or millions component is followed by a remainder of 1-99, then instead of a comma the parts are separated by "and": e.g., $1{,}000{,}011 \Rightarrow$ "one million and eleven", $20{,}222{,}043 \Rightarrow$ "twenty million, two hundred and twenty-two thousand and forty-three".

Input

Input will consist of one or more datasets. Each dataset will consist of a single line containing a non-negative integer in the range $0 \ldots 999{,}999{,}999$. Although we have used commas within digit strings for clarity in this problem description, there will be no commas in the input. There will be no leading zeros on positive input numbers.

A line with a negative value signals the end of input.

Output

For each dataset, print a single line containing the spelled-out equivalent of the number, according to the rules above.

Formatting requirements:

- The output must be left-justified.
- All alphabetic characters must be in lower-case.
- Exactly one blank must separate adjacent words, except when a hyphen or comma is called for.
- When a comma is used, it must be followed by exactly one blank.
- When a hyphen is used, no blank space appears to either side of the hyphen.

Example

Given the input:

```
1
222143
1000001
-1
```

the output should be:

```
one
two hundred and twenty-two thousand, one hundred and forty-three
one million and one
```

Problem H: The Scheming Gardener

The Jackstraw Gardening Club is a cooperative of gardeners whose share a common lot of land, parceling it out every year among their members, each of whom gets their own plot of land within the larger lot.

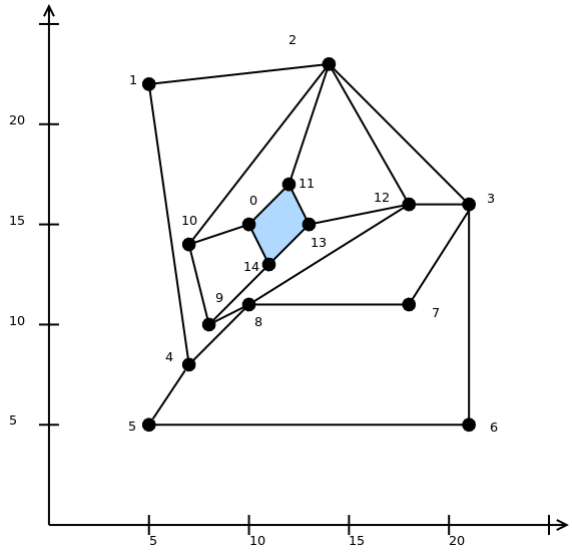
Each winter, before the planting season, the club gathers together on the lot to select their plots. Each member has a handful of wooden stakes and a ball of string. Taking turns in rotation, each member may choose to drive a new stake into the ground, and then may tie a length of string between one or more pairs of stakes, forming a straight-line connection between the two. Strings may not cross (except at the ends where they are tied to same stake) nor may they lie directly atop one another co-linearly. A string may not have both ends tied to the same stake. Stakes may not be driven so close to an existing string or stake that one could not easily step between them.

Each portion of the lot entirely surrounded by strings defines one garden plot. The process continues until a majority of the club members feel that enough enclosed plots have been formed, are willing to stop, subject to the limitations:

- Upon stopping, any useless strings will be removed. A useless string is one where the land just to each side of the string lies in the same plot or in the unenclosed portion of the lot.
- At least one enclosed area must remain after removing the useless strings.
- There will be no enclosed plots of zero area.
- All the remaining strings (after the useless ones are removed) will be connected — it will be possible to trace a path from one string to any other string in the lot.

The gardeners then choose their plots from among the enclosed areas.

You are lucky enough to have the first choice. You know that most of your fellow gardeners will probably try for the largest area plots they can get, but you have a different goal entirely. You are hoping to raise a single, beautiful pumpkin that will win first prize at the next county fair. You don't need much space, but you worry that your pumpkins may be chewed upon by deer, rabbits, mice, and other four-footed wildlife. You want to choose a plot that would force such vermin to cross as many other plots as possible before reaching yours. You hope that the sheer variety of crops presented by the other gardeners will distract the vermin before they ever get to your plot. The vermin will not step directly on or over a stake, but will always pass it to one side or the other.



Input

Input will consist of 1 or more datasets. Each dataset will begin with a line containing two integers, P, and E. $2 < P \leq 750$, $3 \leq E \leq 1000$. A value of zero for P indicates end of the input.

The first line of the dataset is followed by P lines, each containing x, y coordinates of one point. These will be integers in the range $0 \ldots 10{,}000$. These points will be distinct.

Those lines are followed by E lines, each containing a pair of point numbers, indicating a connection between those two points. These numbers will be in the range $0 \ldots P - 1$ and refer to the order of occurrence of the points in the earlier input, with point 0 being the first such point.

Output

For each dataset, find a plot that maximizes the smallest number of other plots that an animal approaching from outside would need to cross before reaching your chosen one. Print the minimum number of other plots that would need to be crossed by such an animal for your chosen plot.

Example

Here is a possible input:

```
15 23
10 15
5 22
14 23
21 16
7 8
5 5
21 5
18 11
10 11
8 10
7 14
12 17
18 16
13 15
11 13
1 2
2 3
2 12
2 11
2 10
3 6
3 7
3 12
```

```
4 1
4 5
4 8
5 6
7 8
8 9
8 12
9 10
9 14
10 0
11 13
11 0
12 13
13 14
0 14
5 6
0 0
0 10
10 0
10 10
5 6
0 1
0 4
1 4
2 3
2 4
3 4
0 0
```

The output should be:

```
2
0
```

There are two datasets in the above input. The first, which ends with the line containing "0 14", corresponds to the picture shown above. The shaded plot in that picture is the one selected by the scheming gardener.