Czech Technical University in Prague
Faculty of Mathematics and Physics, Charles University in Prague
Faculty of Electrical Engineering and Computer Science, Technical University of Ostrava
Faculty of Informatics and Information Technologies, Slovak University of Technology
Faculty of Informatics, Masaryk University
Faculty of Management Science and Informatics, University of Zilina

CTU Open Contest 2009

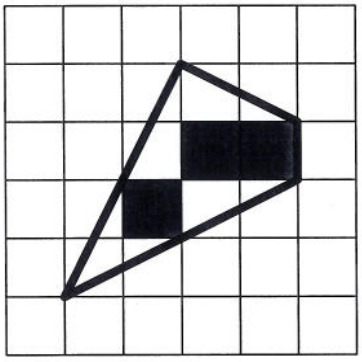
Arable Area

`aa.c`, `aa.C`, `aa.java`, `aa.p`

The prime minister has recently bought a piece of valuable agricultural land, which is situated in a valley forming a regular grid of unit square fields. ACM would like to verify the transaction, especially whether the price corresponds to the market value of the land, which is always determined as the number of unit square fields fully contained in it.

Your task is to write a program that computes the market value. The piece of land forms a closed polygon, whose vertices lie in the corners of unit fields.

For example, the polygon in the picture (it corresponds to the first scenario in Sample Input) contains three square fields.



Input Specification

The input consists of several test scenarios. Each scenario starts with a line containing one integer number N ($3 \leq N \leq 100$), the number of polygon vertices. Each of the following N lines contains a pair of integers X_i and Y_i, giving the coordinates of one vertex. The vertices are listed in the order they appear along the boundary of the polygon. You may assume that no coordinate will be less then -100 or more than 100 and that the boundary does not touch or cross itself.

The last scenario is followed by a line containing single zero.

Output Specification

For each scenario, output one line with an integer number — the number of unit squares that are completely inside the polygon.

Sample Input

```
4
1 1
5 3
5 4
3 5
5
3 3
2 5
3 4
5 2
1 1
5
0 0
0 -50
-50 -51
-51 -50
-50 0
0
```

Output for Sample Input

```
3
1
2500
```

Czech Technical University in Prague
Faculty of Mathematics and Physics, Charles University in Prague
Faculty of Electrical Engineering and Computer Science, Technical University of Ostrava
Faculty of Informatics and Information Technologies, Slovak University of Technology
Faculty of Informatics, Masaryk University
Faculty of Management Science and Informatics, University of Zilina

CTU Open Contest 2009

Clock Captcha

`cc.c`, `cc.C`, `cc.java`, `cc.p`

The digital display clock installed in problem `dd` is only one part of the solution to eliminate antedating documents. The second part is a camera which takes pictures of the office regularly and records at what times was each document signed and stamped. These records can later serve as an evidence.

Your task is to scan a picture take by the camera and decode numbers on the digital display.

Input Specification

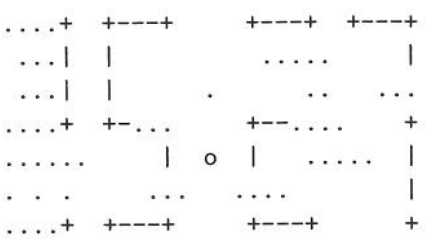
The input will exactly follow Output Specification for Problem `dd`, with only one important exception: Since image scanning is always an error-prone process, some of the characters representing the display may not be recognized correctly. Fortunately, we know which characters are incorrect, because a very sophisticated error-detecting mechanism is used. These characters are replaced by dots (".") in the input. Such spots can represent any of the allowed characters.

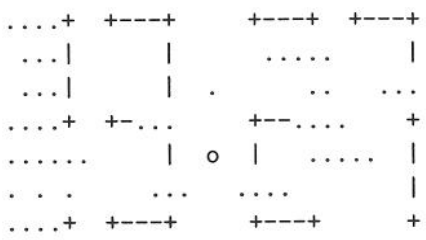
Output Specification

Your output must follow Input Specification for Problem `dd`. For each display, output the time shown on it. You may assume that at least one such time exists.

However, due to the scanning errors, sometimes there may be several possible valid times that could be represented by the same picture. In such cases, we may not risk giving an incorrect information — print the word "`ambiguous`" instead of the time.

Sample Input





```
end
```

Output for Sample Input

```
15:27
ambiguous
end
```

Czech Technical University in Prague
Faculty of Mathematics and Physics, Charles University in Prague
Faculty of Electrical Engineering and Computer Science, Technical University of Ostrava
Faculty of Informatics and Information Technologies, Slovak University of Technology
Faculty of Informatics, Masaryk University
Faculty of Management Science and Informatics, University of Zilina

CTU Open Contest 2009

Digital Display

`dd.c`, `dd.C`, `dd.java`, `dd.p`

There is a suspicion that some officers cheat when reporting exact time of various events. They antedate important documents and gain enormous profit by that. To eliminate such practices, ACM decided to install big digital clock on the wall of each office. Thank to this, everyone can easily see the current time and verify that it is correctly recorded on documents.

Your task is to write a firmware for this digital clock. Digits will be shown using the classical seven-segment display.

Input Specification

On each line of the input, there are exactly five characters: two digits, a colon (":") and another two digits. These characters specify a time between 00:00 and 23:59, inclusive.

The last time is followed by a line containing the word "`end`".

Output Specification

For each input line, print a display consisting of seven rows of exactly 29 characters each. The only characters allowed are dashes ("-") for horizontal segments, pipes ("|") for vertical segments, plus signs ("+") for corners, small latin letter "o" for the colon between hours and minutes, and spaces (" ").

The area for each digit is exactly 5 characters wide and there are two empty columns (spaces) between neighboring digit areas or between a digit area and the colon. Exact digit shapes and placements can be seen in Sample Output.

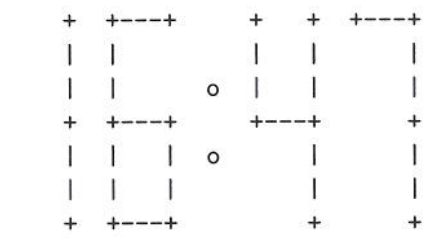
Always print two empty lines after each time displayed. Also, print the word "`end`" at the end of the output.

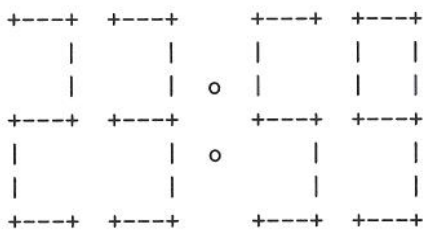
Notice *In this problem, the meaning of evaluation system responses is a little bit different than usual. If you get a Presentation Error, it most probably means that you did not follow the output format exactly. Note that it does not necessarily imply that your answer is close to the correct one. On the other hand, a Wrong Answer will be issued if the format seems basically all-right but the contents of the display are not correct. If your output contains both errors, the decision between Wrong Answer and Presentation Error is solely at the discretion of the system. Therefore, you should never try to derive any important conclusions from the response.*

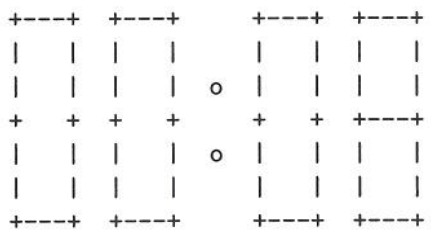
Sample Input

```
16:47
23:59
00:08
end
```

Output for Sample Input

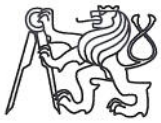






```
end
```

Czech Technical University in Prague
Faculty of Mathematics and Physics, Charles University in Prague
Faculty of Electrical Engineering and Computer Science, Technical University of Ostrava
Faculty of Informatics and Information Technologies, Slovak University of Technology
Faculty of Informatics, Masaryk University
Faculty of Management Science and Informatics, University of Zilina



CTU Open Contest 2009

Intriguing Identifiers

`ii.c`, `ii.C`, `ii.java`, `ii.p`

Corruptionists often use fictional persons to camouflage their illegal activities. They create fake database records containing non-existing names, addresses, or even personal identifiers. ACM would like to fight against such practices. They are going to examine several suspicious databases and search them for invalid records. Can you help them by checking validity of Personal IDs?

(Please note: In reality, some of the facts stated below may not be true and some identifiers may violate them in exceptional cases. However, for the purpose of this problem, we will assume an ideal world without any such exceptions.)

Personal ID (a.k.a. "rodné číslo" or simply RČ) is a unique identifier of a Czech (or Slovak) citizen. It is assigned once a child is born and remains unchanged for the whole life.

The identifier has two numerical parts separated with a single slash character ("/"). The first part always consists of exactly six digits and it represents the birth date of the person in the form of "`YYMMDD`": the first two digits specify a year ("modulo" 100), other two digits a month (01–12), and the last two digits a day in the month (01–31, but some months have less days, of course). Beside the birth date, this part stores also information about the gender: all girls get the month number increased by 50. Thus, for instance, 61 means November for them.

The second part may have either 3 or 4 digits (all of them may be zero). For all identifiers assigned until December 31, 1953, there were only three digits. Starting from January 1, 1954, all children are assigned IDs with exactly four digits after the slash. The additional digit allows for a simple automated check — if we remove the slash, the whole ten-digits-long number must be divisible by 11.

Input Specification

Each line of the input contains one string consisting of at least 1 and at most 20 characters, each of them being either a digit or a slash ("/"). The word "`end`" follows the last string.

Output Specification

For each string, output the word "`invalid`" if the given string does not represent a valid Personal ID of a person born between January 1, 1920 and December 31, 2009, inclusive.

For valid identifiers, print either "`girl`" or "`boy`", depending on the gender of the child who could be assigned that ID.

Sample Input

```
531019/534
541019/123
535318/6663
545318/6662
545318/6666
006231/5000
000229/0002
010229/0001
/
/3/
200101/000
191231/999
end
```

Output for Sample Input

```
boy
invalid
invalid
girl
invalid
girl
boy
invalid
invalid
invalid
boy
invalid
```

Czech Technical University in Prague
Faculty of Mathematics and Physics, Charles University in Prague
Faculty of Electrical Engineering and Computer Science, Technical University of Ostrava
Faculty of Informatics and Information Technologies, Slovak University of Technology
Faculty of Informatics, Masaryk University
Faculty of Management Science and Informatics, University of Zilina

CTU Open Contest 2009

Letter Lies

`ll.c`, `ll.C`, `ll.java`, `ll.p`

Some government projects are not done by government offices themselves, but are instead contracted to private companies. The selection of the best candidate is a task for a specially selected experts. Before the selection process starts, all the candidates send letters with their offer. The committee then picks the best offer based on its price and quality.

That was the theory... In reality, the selection process is often just a formality and the winner was chosen long before, based on the bribe given to the committee. To masquerade these frauds, fake candidate companies are included in the process. And, of course, it is necessary to pretend that the offer letters by these companies have been received. Some people are involved in such selection procedures so often that they implemented an automated letter generator to make their work easier. The generator can assemble letters from a list of pre-defined sentences and the following rules:

- Some sentences are greetings and can appear only at the beginning of the letter. Each letter has to start with a greeting.
- Some sentences are closings and can appear only at the end. Each letter has to end with a closing.
- No sentence can appear twice in the same letter.
- Successor rules: Each sentence restricts the list of other sentences that can follow. For example, a sentence saying "`Hello`" cannot be followed by "`This concludes our offer`".
- The resulting letter must have an appropriate length. Therefore, the exact number of sentences is specified.

Would you help ACM to detect fake letters generated by this program? You are given a set of letter-generator rules and your task is to compute the total number of different valid letters that can be assembled.

Input Specification

The input will consist of several test scenarios. Each scenario starts by a line with four positive integers N, L, B, F. N is the number of all sentences ($1 \leq N \leq 1000$), B and F are the number of greetings and closings ($1 \leq B, F \leq 1000$), and L is the desired length ($1 \leq L \leq 1000000$).

Then there are N lines with successor rules; i-th of them is composed of the number i, integer D_i ($0 \leq D_i \leq 1000$), and a list of D_i integers determining the sentences that can follow sentence number i.

Next B lines contain numbers of all possible starting sentences and the last F lines numbers of all possible final sentences.

The last scenario is followed by a line containing four zeros.

The author of the generator did not implement any check that some sentence occurs more than once. Instead, this is always guaranteed by the set of successor rules. If these rules are followed, you may safely assume that no letter can ever contain the same sentence twice, that a greeting can appear only at the beginning and a closing only at the end of the letter.

Output Specification

For each scenario, output a line with one integer number — the total number of valid letters with exactly L sentences. Please note that this number may exceed 2^{32}. If there is no valid letter of the given length, output the string "`impossible`".

Sample Input

```
4 2 2 2
1 1 3
2 1 4
3 0
4 0
1
2
4
3
2 1000000 1 1
1 1 2
2 0
1
2
0 0 0 0
```

Output for Sample Input

```
2
impossible
```

Czech Technical University in Prague
Faculty of Mathematics and Physics, Charles University in Prague
Faculty of Electrical Engineering and Computer Science, Technical University of Ostrava
Faculty of Informatics and Information Technologies, Slovak University of Technology
Faculty of Informatics, Masaryk University
Faculty of Management Science and Informatics, University of Zilina

CTU Open Contest 2009

Odd Opportunities

`oo.c`, `oo.C`, `oo.java`, `oo.p`

ACM discovered there is a secret worldwide organization (Mafia) that operates in complete secrecy. Each member of Mafia has contacts to some other members, but not necessarily to all of them.

The new (recently elected) head of the Mafia came with a brilliant plan to make their operation even more secret: Some members will have to eliminate some of their contacts and to stop communicating with them. It has been announced that it is very important whether the number of contacts is odd or even. Each member has been instructed whether the number of their contacts must remain an odd or an even number.

A little bit of chaos developed after this announcement. Nobody understands why this is so important, but nobody dares to protest. Your task is to write a program that will suggest a suitable solution and select contacts to be dropped.

Input Specification

The input will consist of several test scenarios. Each scenario starts by a line with two positive integers V and E. V is the number of Mafia members ($1 \leq V \leq 1000$) and E is the total number of contacts ($0 \leq E \leq V.(V-1)/2$).

Then there are E lines describing individual contacts. Each line contains two integers v_1 and v_2, which means that there is a contact between members v_1 and v_2 ($1 \leq v_1, v_2 \leq N$, $v_1 \neq v_2$).

Then there is one more line in each scenario containing exactly V lowercase characters. Each character corresponds to one member and is either "`o`" (the number of contacts must remain odd) or "`e`" (the number of contacts must remain even). Obviously, the first character corresponds to member 1, second character to member 2, etc.

The last scenario is followed by a line containing two zeros.

Output Specification

For each scenario, output a subset of original contacts such that will satisfy the even/odd conditions for all members. On the first line, output one integer number R — the number of contacts that remain ($0 \leq R \leq E$). Then, output R lines, each of them specifying two members that remain in contact. Make sure no pair appears more than once.

If it is not possible to find a suitable subset of contacts, output the word "`impossible`". If there are more correct solutions, you may choose any of them.

Sample Input

```
5 6
1 2
2 3
3 4
4 5
1 3
1 4
oeooo
3 1
1 2
oeo
5 0
eeeee
5 0
eeoee
5 0
eeoeo
4 2
1 2
4 3
eeee
0 0
```

Output for Sample Input

```
3
4 5
4 3
1 4
impossible
0
impossible
impossible
0
```

Czech Technical University in Prague
Faculty of Mathematics and Physics, Charles University in Prague
Faculty of Electrical Engineering and Computer Science, Technical University of Ostrava
Faculty of Informatics and Information Technologies, Slovak University of Technology
Faculty of Informatics, Masaryk University
Faculty of Management Science and Informatics, University of Zilina

CTU Open Contest 2009

Peculiar Primes

`pp.c`, `pp.C`, `pp.java`, `pp.p`

The level of corruption in some countries is really high. It is hard to imagine that these unethical manners have already hit the academic field. Some rumors are spreading that some students tried to bribe their lecturers to get better grades. Would you believe it?

But the real situation may be even much worse. ACM has a very strong suspicion that somebody has bribed mathematicians in the Academy of Sciences in order to forge some of their results. In particular, it is suspected that a very influential person wants to prefer some prime numbers over others.

It is said that many mathematicians have already completely stopped using some primes and they create only those numbers that can be "assembled" without those primes. Your task is to verify this hypothesis.

Given a set of prime numbers, your program should output all integer numbers that can be created solely by multiplying these primes, without using any other primes.

Input Specification

The input will consist of several test scenarios. Each scenario starts by a line with a single positive integer N ($1 \leq N \leq 10$) — the number of primes in the set.

On the second line of a scenario, there are N integer numbers $2 \leq P_1 < P_2 < P_3 < \ldots < P_N < 10000$, separated by a space. You are guaranteed that all these numbers are prime.

On the third line of each scenario, there are 2 integers X and Y ($1 \leq X \leq Y < 2^{31}$), separated by a space.

The last scenario is followed by a line containing single zero.

Output Specification

Your task is to print all positive integer numbers in the closed interval $[X, Y]$ that have no other prime factors than those given in the input (P_i). Print all such numbers in the increasing order, with no duplicates and separated by a single comma character (","). If there are no such numbers, print the word "`none`" instead.

Note that the number 1 does not need any primes to be constructed and is therefore always allowed.

Sample Input

```
1
3
1 12
2
2 3
10 20
3
2 3 5
20 30
1
17
20 30
0
```

Output for Sample Input

```
1,3,9
12,16,18
20,24,25,27,30
none
```

Czech Technical University in Prague
Faculty of Mathematics and Physics, Charles University in Prague
Faculty of Electrical Engineering and Computer Science, Technical University of Ostrava
Faculty of Informatics and Information Technologies, Slovak University of Technology
Faculty of Informatics, Masaryk University
Faculty of Management Science and Informatics, University of Zilina

CTU Open Contest 2009

Robotic Rails

`rr.c`, `rr.C`, `rr.java`, `rr.p`

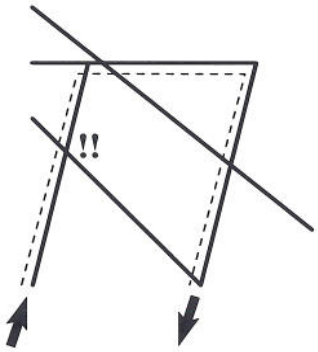
ACM recently presented their new programme to fight against corruption by employing robots in various activities that have been performed by humans before, especially in the state machinery.

Robots have one big advantage — it is practically impossible to bribe them, whatever amount you offer.

Robots also have one big disadvantage — someone must program them to do anything useful. A robot without a program would only stand in the same place and never move.

Your task is to program a mechanical robot named Karel to allow him to move. Karel is situated in a very large hall and it can move using a special system of rails, which are built in the floor. The robot always moves across a single rail, which is so narrow that we can consider its width being zero.

All rails are formed by straight segments, which may intersect with each other (or even overlap). The robot can switch from one segment to another at any point they have in common. However, due to some technical limitations, at any single point, the robot may never turn by more than 90 degrees. In other words, all angles on Karel's path must be obtuse.[1]



If we look at the picture above, it is impossible to make a turn at the crossing marked by exclamation marks and Karel must use a longer path instead (dashed line). The picture corresponds to the first scenario in Sample Input.

Your task is to find the shortest path for Karel to move from one position to another.

[1] obtuse angle = tupý úhel / tupý uhol

Input Specification

The input will consist of several test scenarios. Each scenario starts by a line with a single positive integer R ($1 \leq R \leq 100$) — the number of rail segments. Then there are R lines, each containing four integers x_1, y_1, x_2, y_2 giving the coordinates of the endpoints of one segment. You may assume that no coordinate will be less then -10000 or more than 10000 and that all segments will have non-zero length.

The last scenario is followed by a line containing single zero.

Karel always starts at the first point given in the scenario (at the beginning of the first segment) and is facing the direction of that first segment - therefore, its first movement must start either in that direction or within an angle of 90 degrees.

The target point is the last point given in the input of that scenario (the end of the last segment). You may assume these this point is always different from the start position. It is required that Karel not only reaches the target point, but it must also remain standing in a correct direction — the direction of the last segment. Karel may either arrive there via that segment, or via a different one but with a deviation no more than 90 degrees.

Output Specification

For each scenario, output one single number, rounded to exactly three digits after the decimal point (there may be unnecessary zeros). The number must state the length of the shortest path that satisfies all criteria described above. If it is not possible to reach the target point at all, output the word "`unreachable`" instead.

Sample Input

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

Output for Sample Input

```
9.541
unreachable
```

Czech Technical University in Prague
Faculty of Mathematics and Physics, Charles University in Prague
Faculty of Electrical Engineering and Computer Science, Technical University of Ostrava
Faculty of Informatics and Information Technologies, Slovak University of Technology
Faculty of Informatics, Masaryk University
Faculty of Management Science and Informatics, University of Zilina

CTU Open Contest 2009

Suspicious Stocks

`ss.c`, `ss.C`, `ss.java`, `ss.p`

The corruption is always extremely difficult to prove. Even if we know that some people accept bribes, it is almost impossible to punish them, because the only witnesses are them and the people who offered the bribe. Neither of them is likely to testify.

One way to prove that something illegal has happened is to count the money a person has and then give evidence that it was impossible for him/her to earn that money legally. However, this is not so easy. For example, the person may easily claim that the money have been donated by their uncle or earned by trading stocks.

ACM would like to be able to verify these explanations. Since it is impossible to verify the existence of "uncles", we will concentrate on the stocks. You are given the history of stock prices, and your task is to compute the maximal profit that could possibly be earned by buying and selling the stock.

Input Specification

The input will consist of several test scenarios, each of them consisting of two lines of text. The first line of a scenario contains two positive integers D and M ($1 \leq D \leq 70000$, $1 \leq M \leq 40000$) separated by a space. D is the number of days when the trading took place, M is the amount of money available at the beginning.

The second line of each scenario contains D positive integers $p_1, p_2, p_3, \ldots, p_D$ ($1 \leq p_i \leq 40000$) separated by a space. The number p_i denotes the price of one piece of stock at day i, meaning it is possible to buy or sell one piece for that price on that day.

The last scenario is followed by a line containing single zero.

Output Specification

For each scenario, output one line containing one number: The maximum profit that could be achieved by at most one Buy operation followed by one Sell operation. Assume that there is no stock at the beginning and that it is possible to do only one Buy during the whole trading.

For simplicity, assume that it is possible to buy as many pieces of stock as there are money for, but we can only buy and sell whole pieces (integer number) of stock, not fractions. For example, if the stock price is \$3 and we have \$11, we can only buy 3 pieces.

The Buy operation is followed by exactly one Sell operation on some of the following days. Of course, it is possible (and recommended) to sell all the pieces of stock to maximize profit.

Sample Input

```
3 1
1 2 3
3 1000
1200 40 10
3 10
3 4 5
5 10
2 3 7 1 4
0
```

Output for Sample Input

```
2
0
6
30
```