acm International Collegiate Programming Contest 2008 IBM event sponsor

ACM International Collegiate Programming Contest 2008

South American Regional Contests

November 14th-15th, 2008

Contest Session

This problem set contains 11 problems; pages are numbered from 1 to 22.

This problem set is used in simultaneous contests hosted in the following countries:

- Argentina
- Bolivia
- Brazil
- Chile
- Colombia
- Peru
- Venezuela

Version 2

Problem A
Almost Shortest Path

Source file name: `almost.c`, `almost.cpp` *or* `almost.java`

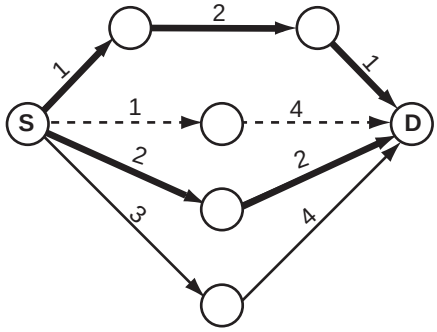
Finding the shortest path that goes from a starting point to a destination point given a set of points and route lengths connecting them is an already well known problem, and it's even part of our daily lives, as shortest path programs are widely available nowadays.

Most people usually like very much these applications as they make their lives easier. Well, maybe not that much easier.

Now that almost everyone can have access to GPS navigation devices able to calculate shortest paths, most routes that form the shortest path are getting slower because of heavy traffic. As most people try to follow the same path, it's not worth it anymore to follow these directions.

With this in his mind, your boss asks you to develop a new application that only he will have access to, thus saving him time whenever he has a meeting or any urgent event. He asks you that the program must answer not the shortest path, but the almost shortest path. He defines the almost shortest path as the shortest path that goes from a starting point to a destination point such that no route between two consecutive points belongs to any shortest path from the starting point to the destination.

For example, suppose the figure below represents the map given, with circles representing location points, and lines representing direct, one-way routes with lengths indicated. The starting point is marked as `S` and the destination point is marked as `D`. The bold lines belong to a shortest path (in this case there are two shortest paths, each with total length 4). Thus, the almost shortest path would be the one indicated by dashed lines (total length 5), as no route between two consecutive points belongs to any shortest path. Notice that there could exist more than one possible answer, for instance if the route with length 3 had length 1. There could exist no possible answer as well.



Input

The input contains several test cases. The first line of a test case contains two integers N ($2 \leq N \leq 500$) and M ($1 \leq M \leq 10^4$), separated by a single space, indicating respectively the number of points in the map and the number of existing one-way routes connecting two points directly. Each point is identified by an integer between 0 and $N-1$. The second line contains two integers S and D, separated by a single space, indicating respectively the starting and the destination points ($S \neq D$; $0 \leq S, D < N$). Each one of the following M lines contains three integers U, V and P ($U \neq V$; $0 \leq U, V < N$; $1 \leq P \leq 10^3$), separated by single spaces, indicating the existence of a one-way route from U to V with distance P. There is at most one

route from a given point U to a given point V, but notice that the existence of a route from U to V does not imply there is a route from V to U, and, if such road exists, it can have a different length. The end of input is indicated by a line containing only two zeros separated by a single space.

The input must be read from standard input.

Output

For each test case in the input, your program must print a single line, containing `-1` if it is not possible to match the requirements, or an integer representing the length of the almost shortest path found.

The output must be written to standard output.

Sample input	**Output for the sample input**
7 9 0 6 0 1 1 0 2 1 0 3 2 0 4 3 1 5 2 2 6 4 3 6 2 4 6 4 5 6 1 4 6 0 2 0 1 1 1 2 1 1 3 1 3 2 1 2 0 3 3 0 2 6 8 0 1 0 1 1 0 2 2 0 3 3 2 5 3 3 4 2 4 1 1 5 1 1 3 0 1 0 0	5 -1 6

Problem B
Bases

Source file name: `bases.c`, `bases.cpp` *or* `bases.java`

What do you get if you multiply 6 by 9? The answer, of course, is 42, but only if you do the calculations in base 13.

Given an integer $B \geq 2$, the *base B numbering system* is a manner of writing integers using only digits between 0 and $B-1$, inclusive. In a number written in base B, the rightmost digit has its value multiplied by 1, the second rightmost digit has its value multiplied by B, the third rightmost digit has its value multiplied by B^2, and so on.

Some equations are true or false depending on the base they are considered in. The equation $2+2=4$, for instance, is true for any $B \geq 5$ — it does not hold in base 4, for instance, since there is no digit '4' in base 4. On the other hand, an equation like $2+2=5$ is never true.

Write a program that given an equation determines for which bases it holds.

Input

Each line of the input contains a test case; each test case is an equation of the form "`EXPR=EXPR`", where both "`EXPR`" are arithmetic expressions with at most 17 characters.

All expressions are valid, and contain only the characters '`+`', '`*`' and the digits from '`0`' to '`9`'. No expressions contain leading plus signs, and no numbers in it have leading zeros.

The end of input is indicated by a line containing only "`=`".

The input must be read from standard input.

Output

For each test case in the input your program should produce a single line in the output, indicating for which bases the given equation holds.

If the expression is true for infinitely many bases, print "`B+`", where B is the first base for which the equation holds.

If the expression is valid only for a finite set of bases, print them in ascending order, separated by single spaces.

If the expression is not true in any base, print the character '`*`'.

The output must be written to standard output.

Sample input	**Output for the sample input**
6*9=42 10000+3*5*334=3*5000+10+0 2+2=3 2+2=4 0*0=0 =	13 6 10 * 5+ 2+

Problem C
Candy

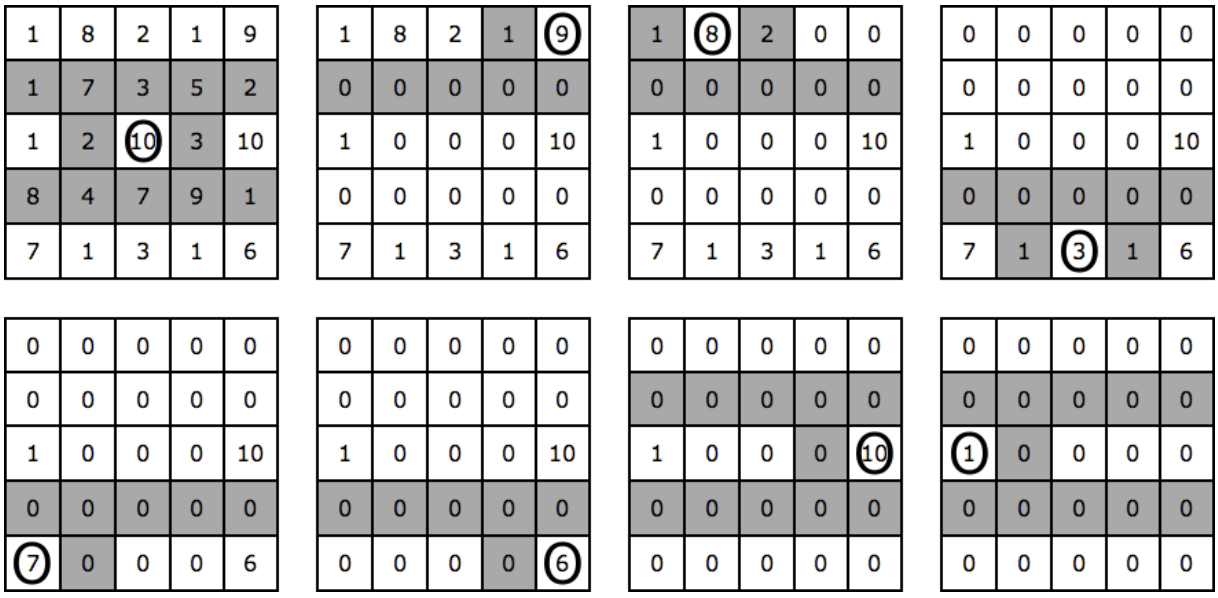
Source file name: `candy.c`, `candy.cpp` *or* `candy.java`

Little Charlie is a nice boy addicted to candies. He is even a subscriber to All Candies Magazine and was selected to participate in the International Candy Picking Contest.

In this contest a random number of boxes containing candies are disposed in M rows with N columns each (so, there are a total of $M \times N$ boxes). Each box has a number indicating how many candies it contains.

The contestant can pick a box (any one) and get all the candies it contains. But there is a catch (there is always a catch): when choosing a box, all the boxes from the rows immediately above and immediately below are emptied, as well as the box to the left and the box to the right of the chosen box. The contestant continues to pick a box until there are no candies left.

The figure bellow illustrates this, step by step. Each cell represents one box and the number of candies it contains. At each step, the chosen box is circled and the shaded cells represent the boxes that will be emptied. After eight steps the game is over and Charlie picked $10 + 9 + 8 + 3 + 7 + 6 + 10 + 1 = 54$ candies.



For small values of M and N, Charlie can easily find the maximum number of candies he can pick, but when the numbers are really large he gets completely lost. Can you help Charlie maximize the number of candies he can pick?

Input

The input contains several test cases. The first line of a test case contains two positive integers M and N ($1 \leq M \times N \leq 10^5$), separated by a single space, indicating the number of rows and columns respectively. Each of the following M lines contains N integers separated by single

spaces, each representing the initial number of candies in the corresponding box. Each box will have initially at least 1 and at most 10^3 candies.

The end of input is indicated by a line containing two zeroes separated by a single space.

The input must be read from standard input.

Output

For each test case in the input, your program must print a single line, containing a single value, the integer indicating the maximum number of candies that Charlie can pick.

The output must be written to standard output.

Sample input	**Output for the sample input**
5 5 1 8 2 1 9 1 7 3 5 2 1 2 10 3 10 8 4 7 9 1 7 1 3 1 6 4 4 10 1 1 10 1 1 1 1 1 1 1 1 10 1 1 10 2 4 9 10 2 7 5 1 1 5 0 0	54 40 17

Problem D
DNA Subsequences

Source file name: `sequence.c`, `sequence.cpp` *or* `sequence.java`

Thomas, a computer scientist that works with DNA sequences, needs to compute longest common subsequences of given pairs of strings. Consider an alphabet Σ of letters and a word $w = a_1 a_2 \cdots a_r$, where $a_i \in \Sigma$, for $i = 1, 2, \ldots, r$. A *subsequence* of w is a word $x = a_{i_1} a_{i_2} \cdots a_{i_s}$ such that $1 \leq i_1 < i_2 < \ldots < i_s \leq r$. Subsequence x is a *segment* of w if $i_{j+1} = i_j + 1$, for $j = 1, 2, \ldots, s-1$. For example the word `ove` is a segment of the word `lovely`, whereas the word `loly` is a subsequence of `lovely`, but not a segment.

A word is a *common subsequence* of two words w_1 and w_2 if it is a subsequence of each of the two words. A *longest common subsequence* of w_1 and w_2 is a common subsequence of w_1 and w_2 having the largest possible length. For example, consider the words w_1 = `lovxxelyxxxxx` and w_2 = `xxxxxxxlovely`. The words w_3 =`lovely` and w_4 = `xxxxxxx`, the latter of length 7, are both common subsequences of w_1 and w_2. In fact, w_4 is their longest common subsequence. Notice that the empty word, of length zero, is always a common subsequence, although not necessarily the longest.

In the case of Thomas, there is an extra requirement: the subsequence must be formed from common segments having length K or more. For example, if Thomas decides that $K = 3$, then he considers `lovely` to be an acceptable common subsequence of `lovxxelyxxxxx` and `xxxxxxxlovely`, whereas `xxxxxxx`, which has length 7 and is also a common subsequence, is not acceptable. Can you help Thomas?

Input

The input contains several test cases. The first line of a test case contains an integer K representing the minimum length of common segments, where $1 \leq K \leq 100$. The next two lines contain each a string on lowercase letters from the regular alphabet of 26 letters. The length ℓ of each string satisfies the inequality $1 \leq \ell \leq 10^3$. There are no spaces on any line in the input. The end of the input is indicated by a line containing a zero.

The input must be read from standard input.

Output

For each test case in the input, your program must print a single line, containing the length of the longest subsequence formed by consecutive segments of length at least K from both strings. If no such common subsequence of length greater than zero exists, then `0` must be printed.

The output must be written to standard output.

Sample input	**Output for the sample input**
3 lovxxelyxxxxx xxxxxxxlovely 1 lovxxelyxxxxx xxxxxxxlovely 3 lovxxxelxyxxxx xxxlovelyxxxxxxx 4 lovxxxelyxxx xxxxxxlovely 0	6 7 10 0

Problem E
Electricity

Source file name: `electricity.c`, `electricity.cpp` *or* `electricity.java`

Martin and Isa stopped playing crazy games and finally got married. It's good news! They're pursuing a new life of happiness for both and, moreover, they're moving to a new house in a remote place, bought with most of their savings.

Life is different in this new place. In particular, electricity is very expensive, and they want to keep everything under control. That's why Martin proposed to keep a daily record of how much electricity has been consumed in the house. They have an electricity meter, which displays a number with the amount of KWh (kilowatt-hour) that has been consumed since their arrival.

At the beginning of each day they consult the electricity meter, and write down the consumption. Some days Martin does it, and some days Isa does. That way, they will be able to look at the differences of consumption between consecutive days and know how much has been consumed.

But some days they simply forget to do it, so, after a long time, their register is now incomplete. They have a list of dates and consumptions, but not all of the dates are consecutive. They want to take into account only the days for which the consumption can be precisely determined, and they need help.

Input

The input contains several test cases. The first line of each test case contains one integer N indicating the number of measures that have been taken ($2 \leq N \leq 10^3$). Each of the N following lines contains four integers D, M, Y and C, separated by single spaces, indicating respectively the day ($1 \leq D \leq 31$), month ($1 \leq M \leq 12$), year ($1900 \leq Y \leq 2100$), and consumption ($0 \leq C \leq 10^6$) read at the beginning of that day. These N lines are increasingly ordered by date, and may include leap years. The sequence of consumptions is strictly increasing (this is, no two different readings have the same number). You may assume that D, M and Y represent a valid date.

Remember that a year is a leap year if it is divisible by 4 and not by 100, or well, if the year is divisible by 400.

The end of input is indicated by a line containing only one zero.

The input must be read from standard input.

Output

For each test case in the input, your program must print a single line containing two integers separated by a single space: the number of days for which a consumption can be precisely determined, and the sum of the consumptions for those days.

The output must be written to standard output.

Sample input	**Output for the sample input**
5 9 9 1979 440 29 10 1979 458 30 10 1979 470 1 11 1979 480 2 11 1979 483 3 5 5 2000 6780 6 5 2001 7795 7 5 2002 8201 8 28 2 1978 112 1 3 1978 113 28 2 1980 220 1 3 1980 221 5 11 1980 500 14 11 2008 600 15 11 2008 790 16 12 2008 810 0	2 15 0 0 2 191

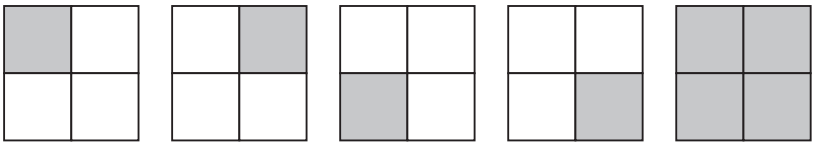
Problem F
Feynman

Source file name: `feynman.c`, `feynman.cpp` *or* `feynman.java`

Richard Phillips Feynman was a well known American physicist and a recipient of the Nobel Prize in Physics. He worked in theoretical physics and also pioneered the field of quantum computing. He visited South America for ten months, giving lectures and enjoying life in the tropics. He is also known for his books "Surely You're Joking, Mr. Feynman!" and "What Do You Care What Other People Think?", which include some of his adventures below the equator.

His life-long addiction was solving and making puzzles, locks, and cyphers. Recently, an old farmer in South America, who was a host to the young physicist in 1949, found some papers and notes that is believed to have belonged to Feynman. Among notes about mesons and electromagnetism, there was a napkin where he wrote a simple puzzle: "how many different squares are there in a grid of $N \times N$ squares?".

In the same napkin there was a drawing which is reproduced below, showing that, for $N = 2$, the answer is 5.



Input

The input contains several test cases. Each test case is composed of a single line, containing only one integer N, representing the number of squares in each side of the grid ($1 \leq N \leq 100$).

The end of input is indicated by a line containing only one zero.

The input must be read from standard input.

Output

For each test case in the input, your program must print a single line, containing the number of different squares for the corresponding input.

The output must be written to standard output.

Sample input	**Output for the sample input**
2 1 8 0	5 1 204

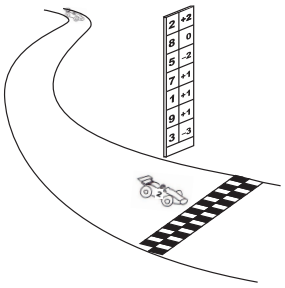
Problem G
Pole Position

Source file name: `pole.c`, `pole.cpp` *or* `pole.java`

In car races, there is always a high pole next to the finish line of the track. Before the race starts, the pole is used to display the starting grid. The number of the first car in the grid is displayed at the top of the pole, the number of the car in second place is shown below that, and so on.

During the race, the pole is used to display the current position of each car: the car that is winning the race has its number displayed at the top of the pole, followed by the car that is in second place, and so on.

Besides showing the current position of a car, the pole is also used to display the number of positions the cars have won or lost, relative to the starting grid. This is done by showing, side by side to the car number, an integer number. A positive value v beside a car number in the pole means that car has won v positions relative to the starting grid. A negative value v means that car has lost v positions relative to the starting grid. A zero beside a car number in the pole means the car has neither won nor lost any positions relative to the starting grid (the car is in the same position it started).



We are in the middle of the Swedish Grand Prix, the last race of the World Championship. The race director, Dr. Shoo Makra, is getting worried: there have been some complaints that the software that controls the pole position system is defective, showing information that does not reflect the true race order.

Dr. Shoo Makra devised a way to check whether the pole system is working properly. Given the information currently displayed in the pole, he wants to reconstruct the starting grid of the race. If it is possible to reconstruct a valid starting grid, he plans to check it against the real starting grid. On the other hand, if it is not possible to reconstruct a valid starting grid, the pole system is indeed defective.

Can you help Dr. Shoo Makra?

Input

The input contains several test cases. The first line of a test case contains one integer N indicating the number of cars in the race ($2 \leq N \leq 10^3$). Each of the next N lines contains two integers C and P, separated by one space, representing respectively a car number ($1 \leq$

$C \leq 10^4$) and the number of positions that car has won or lost relative to the starting grid ($-10^6 \leq P \leq 10^6$), according to the pole system. All cars in a race have different numbers.

The end of input is indicated by a line containing only one zero.

The input must be read from standard input.

Output

For each test case in the input, your program must print a single line, containing the reconstructed starting grid, with car numbers separated by single spaces. If it is not possible to reconstruct a valid starting grid, the line must contain only the value `-1`.

The output must be written to standard output.

Sample input	**Output for the sample input**
4 1 0 3 1 2 -1 4 0 4 22 1 9 1 13 0 21 -2 3 19 1 9 -345 17 0 7 2 2 8 0 5 -2 7 1 1 1 9 1 3 -3 0	1 2 3 4 -1 -1 5 8 2 3 7 1 9

Problem H
Higgs Boson

Source file name: `higgs.c`, `higgs.cpp` *or* `higgs.java`

It's been 100 years since the detection of the first Higgs boson and now particle physics is a mainstream subject in all high schools. Obviously, kids love the fact that they can create tiny black holes using only their portable particle accelerators and show off to their friends and colleagues. Although the creation of big black holes that could swallow the whole planet is possible even with these portable particle accelerators, the devices are programmed to only thrown particles when this undesirable side effect is impossible.

Your granddaughter is trying to create her own black holes with a portable accelerator kit, which is composed of two small particle accelerators that throw, each one, a boson-sized particle. Both particles are thrown at the same time, and a black hole appears when the particles collide. However, your granddaughter doesn't know how much time she'll have to wait before this happens. Fortunately, each accelerator can predict the particle's trajectory, showing four integer values into its display, called A, B, C and D. Each value can be replaced into the following equations:

$$r = At + B$$
$$\theta = Ct + D$$

in order to determine the trajectory of the particle, in polar coordinates. The radius (r) is represented in distance units and the angle (θ) in degrees. The time (t) is given in time units and it is always a rational value which can be represented by an irreducible fraction. Your granddaughter knows that in polar coordinates a point has infinite representations. In general, the point (r, θ) can be represented as $(r, \theta \pm k \times 360^o)$ or $(-r, \theta \pm (2k + 1) \times 180^o)$, where k is any integer. Besides, the origin ($r = 0$) can be represented as $(0, \theta)$ for any θ.

Using these parameters informed by each particle accelerator, your granddaughter wants to determine whether the particles will eventually collide and, if they do, the time when they will collide. After the first collision it is impossible to predict the particle's trajectory, therefore, only the first possible collision should be considered.

Although your granddaughter is really intelligent and has a deep knowledge of particle physics, she does not know how to program computers and is looking for some notes in her grandfather's (or grandmother's) ICPC notebook (don't forget, she is *your* granddaughter!). Fortunately for you, there is a note on your notebook which says that you wrote that code during the 2008 ICPC South America Regional Contest (or, to be more specific, *this* contest).

Input

The input consists of several test cases, one per line. Each test case contains eight integer numbers separated by single spaces, $A_1, B_1, C_1, D_1, A_2, B_2, C_2, D_2$ ($-10^4 \leq A_1, B_1, C_1, D_1, A_2, B_2, C_2, D_2 \leq 10^4$). The first four input values (A_1, B_1, C_1, D_1) correspond to the four parameters displayed by the first portable particle accelerator and the following input values (A_2, B_2, C_2, D_2) correspond to the four parameters displayed by the second portable particle accelerator when both particles are thrown. The end of the input is represented by $A_1 = B_1 = C_1 = D_1 = A_2 =$

$B_2 = C_2 = D_2 = 0$, which should not be processed as a test case, since these are the values displayed by the particle accelerators when a big black hole would be created if the particles were trown. Although the end of input is represented by a line with eight zeroes, note that the number zero is a possible input value.

The input must be read from standard input.

Output

For each test case, your program must output a line containing two non-negative integers t_a and t_b separated by a single space. If there is no possibility of collision, $t_a = t_b = 0$, otherwise, t_a/t_b must be an irreducible fraction representing the earliest collision time. Even if the fraction results in an integer value, you still must output the number 1 as the denominator (see samples below).

The output must be written to standard output.

Sample input	**Output for the sample input**

`1 1 180 0 2 0 180 360`
`10 10 360 0 -24 18 180 72`
`5 5 180 0 -12 9 10 40`
`-9 5 5 180 2 5 5 180`
`0 0 0 0 0 0 0 0`
	`1 1`
	`0 0`
	`4 17`
	`0 1`

Problem I
Traveling Shoemaker Problem

Source file name: `tsp.c`, `tsp.cpp` *or* `tsp.java`

Once upon a time there was a very peaceful country named Nlogonia. Back then, Poly the Shoemaker could come to the country and travel freely from city to city doing his job without any harassment. This task was very easy, as every city in Nlogonia had a direct road to every other city in the country. He could then easily travel the whole country visiting each city exactly once and fixing everybody's shoes.

But not anymore. The times have changed and war has come to Nlogonia. The age when people could travel freely is over.

Confederations identified by colors were formed among the cities all over the country, and now each city belongs to at least one and at most two confederations. When trying to enter a city, you must give to the border officer a ticket from one of the confederations this city belongs to. When leaving the city, you receive a ticket from the other confederation the city belongs to (i.e. different from the one you gave when entering) or from the same confederation if the city only belongs to one.

As Poly the Shoemaker is a long time friend of Nlogonia, he is allowed to choose a ticket and a city he wants to enter as the first city in the country, but after that he must obey the confederations rules. He wants to do the same routine he did before, visiting each city exactly once in Nlogonia, but now it's not easy for him to do this, even though he can choose where to start his journey.

For example, suppose there are four cities, labeled from 0 to 3. City 0 belongs to confederations *red* and *green*; city 1 belongs only to *red*; city 2 belongs to *green* and *yellow*; and city 3 belongs to *blue* and *red*. If Poly the Shoemaker chooses to start at city 0, he can enter it carrying either the *red* or the *green* ticket and leave receiving the other. Should he choose the *red* ticket, he will leave with a *green* ticket, and then there is only city 2 he can travel to. When leaving city 2 he receives the *yellow* ticket and now can't go anywhere else. If he had chosen the *green* ticket as the first he would receive the *red* one when leaving, and then he could travel to cities 1 or 3. If he chooses city 3, when leaving he will receive the *blue* ticket and again can't go anywhere else. If he chooses city 1, he receives the *red* ticket again when leaving (city 1 belongs only to the *red* confederation) and can only travel to city 3 and will never get to city 2. Thus, it is not possible to visit each city exactly once starting at city 0. It is possible, however, starting at city 2 with the *yellow* ticket, leaving the city with the *green* ticket, then visiting city 0, leaving with *red* ticket, then visiting city 1, leaving with *red* ticket again and, at last, visiting city 3.

As you can see, it got really difficult for Poly the Shoemaker to accomplish the task, so he asks you to help him. He wants to know if it's possible to choose a city to start such that he can travel all cities from Nlogonia exactly once.

Can you help Poly the Shoemaker?

Input

The input contains several test cases. The first line of a test case contains two integers N and C, separated by one space, indicating respectively the number of cities ($1 \leq N \leq 500$) and confederations ($1 \leq C \leq 100$) in the country. Each of the next C lines describes a confederation. It starts with one integer K ($0 \leq K \leq N$) and then K integers representing the cities which belong to this confederation. All integers are separated by single spaces and cities are numbered from 0 to $N - 1$. Each city will appear at least once and at most twice and no city will be repeated on the same confederation.

The end of input is indicated by a line containing two zeroes separated by a single space.

The input must be read from standard input.

Output

For each test case in the input, your program must print a single line, containing the integer `-1` if it's not possible to match the requirements or one integer representing the city where Poly the Shoemaker can start his journey. If there are multiple correct answers, print the smallest one.

The output must be written to standard output.

Sample input	**Output for the sample input**
4 4 1 3 3 0 1 3 2 0 2 1 2 3 4 1 0 3 0 1 2 1 1 1 2 3 4 1 1 2 1 0 2 0 2 1 2 0 0	2 -1 1

Problem J
Bora Bora

Source file name: `bora.c`, `bora.cpp` *or* `bora.java`

Bora Bora is a simple card game for children, invented in the South Pacific Island of the same name. Two or more players can play, using a deck of standard cards. Cards have the usual ranks: Ace, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen and King. Each card has also one of four suits: Clubs, Diamonds, Hearts and Spades.

Players sit on a circle around the table and play by turns. The next player to play may be the one to the left (clockwise direction) or to the right (counter-clockwise direction) of the current player, depending on the cards played, as we will see. At the start, the direction of play is clockwise.

The deck is shuffled and each player is dealt a hand of cards. The remaining of the deck is placed, face down, on the table; this is called the *stock* pile. Then the first (topmost) card is removed from the stock and placed on the table, face up, starting another pile, called the *discard* pile.

The objective of the game is for a player to discard all his cards. At each turn, a player discards at most one card. A card can be discarded only if it has the same rank or the same suit as the topmost card on the discard pile. A player discards a card by placing it, face up, in the discard pile (this card becomes the topmost). If a player does not have a suitable card to discard on his turn, he must draw one card from the stock and add it to his hand; if he can discard that card, he does so, otherwise he does nothing else and his turn ends. A player always discards the highest valued card he possibly can. The *value* of a card is determined first by the card rank and then by the card suit. The rank order is the rank itself (Ace is the lowest, King is the highest), and the suit order is, from lowest to highest, Clubs, Diamonds, Hearts and Spades. Therefore, the highest valued card is the King of Spades and the lowest valued card is the Ace of Clubs. As an example, a Queen of Diamonds has a higher value than a Jack (any suit) but has a lower value than a Queen of Hearts.

Some of the discarded cards affect the play, as follows:

- when a Queen is discarded, the direction of play is reversed: if the direction is clockwise, it changes to counter-clockwise, and vice-versa;
- when a Seven is discarded, the next player to play must draw two cards from the stock (the number of cards in his hand increases by two), and misses his turn (does not discard any card);
- when an Ace is discarded, the next player to play must draw one card from the stock (the number of cards in his hand increases by one), and misses his turn (does not discard any card);
- when a Jack is discarded, the next player to play misses his turn (does not discard any card).

Notice that the penalty for the first card in the discard pile (the card draw from the stock at the beginning) is applied to the first player to play. For example, if the first player to play is p and the first card on the discard pile is an Ace, player p draws a card from the stock and does not discard any card on his first turn. Also notice that if the first card is a Queen, the direction of play is reversed to counter-clockwise, but the first player to play remains the same.

The winner is the player who first discards all his cards (the game ends after the winner discards his last card).

Given the description of the shuffled deck and the number of players, write a program to determine who will win the game.

Input

The input contains several test cases. The first line of a test case contains three integers P, M and N, separated by single spaces, indicating respectively the number of players ($2 \leq P \leq 10$), the number of cards distributed to each of the players at the beginning of the game ($1 \leq M \leq 11$) and the total number of cards in the shuffled deck ($3 \leq N \leq 300$). Each of the next N lines contains the description of one card. A card is described by one integer X and one character S, separated by one space, representing respectively the card rank and the card suite. Card ranks are mapped to integers from 1 to 13 (Ace is 1, Jack is 11, Queen is 12 and King is 13). Card suits are designated by the suit's first letter: '`C`' (Clubs), '`D`' (Diamonds), '`H`' (Hearts) or '`S`' (Spades).

Players are identified by numbers from 1 to P, and sit on a circle, in clockwise direction, $1, 2 \ldots P, 1$. The first $P \times M$ cards of the deck are dealt to the players: the first M cards to the first player (player 1), the next M to the second player (player 2), and so on. After dealing the cards to the players, the next card on the deck — the $(P \times M + 1)$-th card — is used to start the discard pile, and the remaining cards form the stock. The $(P \times M + 2)$-th card to appear on the input is the topmost card on the stock, and the last card to appear on the input (the N-th card) is the bottommost card of the stock (the last card that can be drawn). Player 1 is always the first to play (even when the card used to start the discard pile is a Queen). All test cases have one winner, and in all test cases the number of cards in the deck is sufficient for playing to the end of the game.

The end of input is indicated by a line containing only three zeros, separated by single spaces.

The input must be read from standard input.

Output

For each test case in the input, your program must print a single line, containing the number of the player who wins the game.

The output must be written to standard output.

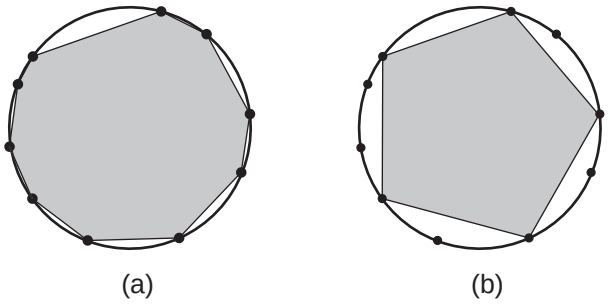
Sample input	**Output for the sample input**
2 2 10 1 D 7 D 1 S 3 C 13 D 1 S 5 H 12 D 7 S 2 C 3 2 11 1 S 7 D 11 D 3 D 7 D 3 S 11 C 8 C 9 H 6 H 9 S 3 3 16 1 H 10 C 13 D 7 C 10 H 2 S 2 C 10 S 8 S 12 H 11 C 1 C 1 C 4 S 5 D 6 S 0 0 0	1 3 2

Problem K
Shrinking Polygons

Source file name: `polygons.c`, `polygons.cpp` *or* `polygons.java`

A polygon is said to be *inscribed* in a circle when all its vertices lie on that circle. In this problem you will be given a polygon inscribed in a circle, and you must determine the minimum number of vertices that should be removed to transform the given polygon into a *regular polygon*, i.e., a polygon that is equiangular (all angles are congruent) and equilateral (all edges have the same length).

When you remove a vertex v from a polygon you first remove the vertex and the edges connecting it to its adjacent vertices w_1 and w_2, and then create a new edge connecting w_1 and w_2. Figure (a) below illustrates a polygon inscribed in a circle, with ten vertices, and figure (b) shows a pentagon (regular polygon with five edges) formed by removing five vertices from the polygon in (a).



In this problem, we consider that any polygon must have at least three edges.

Input

The input contains several test cases. The first line of a test case contains one integer N indicating the number of vertices of the inscribed polygon ($3 \leq N \leq 10^4$). The second line contains N integers X_i separated by single spaces ($1 \leq X_i \leq 10^3$, for $0 \leq i \leq N-1$). Each X_i represents the length of the arc defined in the inscribing circle, clockwise, by vertex i and vertex $(i+1)$ mod N. Remember that an *arc* is a segment of the circumference of a circle; do not mistake it for a *chord*, which is a line segment whose endpoints both lie on a circle.

The end of input is indicated by a line containing only one zero.

The input must be read from standard input.

Output

For each test case in the input, your program must print a single line, containing the minimum number of vertices that must be removed from the given polygon to form a regular polygon. If it is not possible to form a regular polygon, the line must contain only the value `-1`.

The output must be written to standard output.

Sample input	**Output for the sample input**
3 1000 1000 1000 6 1 2 3 1 2 3 3 1 1 2 10 10 40 20 30 30 10 10 50 24 26 0	0 2 -1 5