ACM International Collegiate Programming Contest 2009

Latin American Regional Contests

October 23rd-24th, 2009

Contest Session

This problem set contains 11 problems; pages are numbered from 1 to 19.

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

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

Version 6

Problem A
Another Crisis

File code name: `another`

A couple of years ago, a new world wide crisis started, leaving many people with economical problems. Some workers of a particular company are trying to ask for an increase in their salaries.

The company has a strict hierarchy, in which each employee has exactly one direct boss, with the exception of the owner of the company that has no boss. Employees that are not bosses of any other employee are called *workers*. The rest of the employees and the owner are called *bosses*.

To ask for a salary increase, a worker should file a petition to his direct boss. Of course, each boss is encouraged to try to make their subordinates happy with their current income, making the company's profit as high as possible. However, when at least T percent of its direct subordinates have filed a petition, that boss will be pressured and have no choice but to file a petition himself to his own direct boss. Each boss files at most 1 petition to his own direct boss, regardless on how many of his subordinates filed him a petition. A boss only accounts his direct subordinates (the ones that filed him a petition and the ones that didn't) to calculate the pressure percentage.

Note that a boss can have both workers and bosses as direct subordinates at the same time. Such a boss may receive petitions from both kinds of employees, and each direct subordinate, regardless of its kind, will be accounted as 1 when checking the pressure percentage.

When a petition file gets all the way up to the owner of the company, all salaries are increased. The workers' union is desperately trying to make that happen, so they need to convince many workers to file a petition to their direct boss.

Given the company's hierarchy and the parameter T, you have to find out the minimum number of workers that have to file a petition in order to make the owner receive a petition.

Input

There are several test cases. The input for each test case is given in exactly two lines. The first line contains two integers N and T ($1 \leq N \leq 10^5, 1 \leq T \leq 100$), separated by a single space. N indicates the number of employees of the company (not counting the owner) and T is the parameter described above. Each of the employees is identified by an integer between 1 and N. The owner is identified by the number 0. The second line contains a list of integers separated by single spaces. The integer B_i, at position i on this list (starting from 1), indicates the identification of the direct boss of employee i ($0 \leq B_i \leq i - 1$).

The last test case is followed by a line containing two zeros separated by a single space.

Output

For each test case output a single line containing a single integer with the minimum number of workers that need to file a petition in order to get the owner of the company to receive a petition.

Sample input	**Output for the sample input**

`3 100`
`0 0 0`
`3 50`
`0 0 0`
`14 60`
`0 0 1 1 2 2 2 5 7 5 7 5 7 5`
`0 0`
	`3`
	`2`
	`5`

Problem B
Brothers

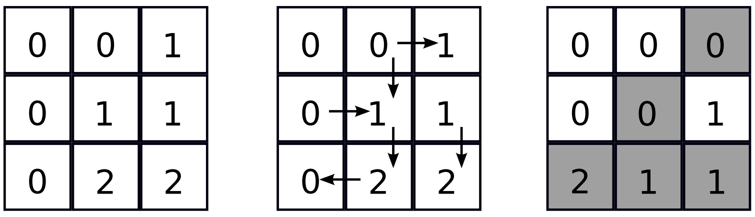
File code name: `brothers`

In the land of ACM ruled a great King who became obsessed with order. The kingdom had a rectangular form, and the King divided the territory into a grid of small rectangular *counties*. Before dying, the King distributed the counties among his sons.

However, he was unaware that his children had developed a strange rivalry: the first heir hated the second heir, but not the rest; the second heir hated the third heir, but not the rest, and so on . . . Finally, the last heir hated the first heir, but not the other heirs.

As soon as the King died, the strange rivalry among the King's sons sparked off a generalized war in the kingdom. Attacks only took place between pairs of adjacent counties (adjacent counties are those that share one vertical or horizontal border). A county X attacked an adjacent county Y whenever the owner of X hated the owner of Y. The attacked county was always conquered by the attacking brother. By a rule of honor all the attacks were carried out simultaneously, and a set of simultaneous attacks was called a *battle*. After a certain number of battles, the surviving sons made a truce and never battled again.

For example, if the King had three sons, named 0, 1 and 2, the figure below shows what happens in the first battle for a given initial land distribution:



Land before the first battle. Attacks during the first battle. Land after the first battle. Earned territory in gray.

You were hired to help an ACM historian determining, given the number of heirs, the initial land distribution and the number of battles, what was the land distribution after all battles.

Input

The input contains several test cases. The first line of a test case contains four integers N, R, C and K, separated by single spaces. N is the number of heirs ($2 \leq N \leq 100$), R and C are the dimensions of the kingdom ($2 \leq R, C \leq 100$), and K is the number of battles ($1 \leq K \leq 100$). Heirs are identified by sequential integers starting from zero (0 is the first heir, 1 is the second heir, . . ., $N - 1$ is the last heir). Each of the next R lines contains C integers $H_{r,c}$ separated by single spaces, representing initial land distribution: $H_{r,c}$ is the initial owner of the county in row r and column c ($0 \leq H_{r,c} \leq N - 1$).

The last test case is followed by a line containing four zeroes separated by single spaces.

Output

For each test case, your program must print R lines with C integers each, separated by single spaces in the same format as the input, representing the land distribution after all battles.

Sample input

```
3 4 4 3
0 1 2 0
1 0 2 0
0 1 2 0
0 1 2 2
4 2 3 4
1 0 3
2 1 2
8 4 2 1
0 7
1 6
2 5
3 4
0 0 0 0
```

Output for the sample input

```
2 2 2 0
2 1 0 1
2 2 2 0
0 2 0 0
1 0 3
2 1 2
7 6
0 5
1 4
2 3
```

Problem C
Code Lock

File code name: `code`

A lock you use has a code system to be opened instead of a key. The lock contains a sequence of wheels. Each wheel has the 26 letters of the English alphabet '*a*' through '*z*', in order. If you move a wheel up, the letter it shows changes to the next letter in the English alphabet (if it was showing the last letter '*z*', then it changes to '*a*'). If you move the wheel down, it changes to show the previous letter in the English alphabet (if it was showing '*a*', then it changes to '*z*').

It is also possible to move any subsequence of contiguous wheels in the same direction with only one movement. This has the same effect of moving each of the wheels within the subsequence on that direction, but saves the effort of doing that one wheel at a time.

The lock opens when the wheels show a secret sequence of letters. Currently all wheels are showing the letter '*a*'. You want to know the minimum number of movements you need to open the lock.

Input

The input has several test cases. Each of them is given in exactly one line containing a non-empty string of at most 1000 lowercase letters. The string represents the secret sequence of letters that opens the lock.

The last test case is followed by a line containing a single asterisk.

Output

For each test case, output a line containing a single integer with the minimum number of movements to open the lock.

Sample input	**Output for the sample input**
`abcxyz`	`5`
`abcdefghijklmnopqrstuvwxyz`	`25`
`aaaaaaaaa`	`0`
`zzzzzzzzz`	`1`
`zzzzbzzzz`	`3`
`*`	

Problem D
Dinner Hall

File code name: `dinner`

The University administration plans to build a new dinner hall, to replace the several small (and rather inadequate) dinner halls spread over the campus. To estimate the number of places needed in the new dinner hall, they performed an experiment to measure the maximum total number of clients inside the existing dinner halls at any time. They hired several students as *pollers*, and positioned one poller at each entrance and each exit of the existing dinner halls. The pollers' task was to note in small cards the time each client entered or exited the hall (one card for each event). In each card they wrote the time, in the format HH:MM:SS, and the associated event (letter 'E' for an entry, letter 'X' for and exit).

The experiment started in the morning, before breakfast, and ended in the evening, after dinner. The pollers had their watches synchronized, and the halls were empty both before and after the experiment (that is, no client was inside a hall before the experiment began, and no client remained in a hall after the experiment ended). The pollers wrote exactly one card for every client who entered a hall and for every client who exited a hall.

After the experiment, the cards were collected and sent to the administration for processing. The task, however, was not as easy as planned, because two problems were detected. Firstly, the cards were bunched together in no particular order and therefore needed sorting; that is fairly easy, but time-consuming to do by hand. But what is worse is that, although all cards had a valid time, some pollers forgot to write the letter specifying the event. The University administration decided they needed help from an expert!

Given a set of cards with times and the indication of the event (the indication of the event may be missing), write a program to determine the maximum number of clients that could possibly had been inside the dinner halls in a given instant of time.

Input

The input contains several test cases. The first line of a test case contains one integer N indicating the number of cards collected in the experiment ($2 \leq N \leq 64800$). Each of the next N lines contains the information written in a card, consisting of a *time specification*, followed by a single space, followed by an *event specification.* A time specification has the format `HH:MM:SS`, where `HH` represents hours ($06 \leq$ `HH` ≤ 23), `MM` represents minutes ($00 \leq$ `MM` ≤ 59) and `SS` represents seconds ($00 \leq$ `SS` ≤ 59). Within a test case, no two cards have the same time. An event specification is a single character: uppercase '`E`' for entry, uppercase '`X`' for exit and '`?`' for unknown. Information may be missing, but the information given is always correct. That is, the times noted in all cards are valid. Also, if a card describes an entry, then a client did enter a hall at the informed time; if a card describes an exit, then a client did leave a hall at the informed time; and if a card describes an unknown event, then a client did enter or leave a hall at the informed time.

The last test case is followed by a line containing a single zero.

Output

For each test case in the input, your program must print a single line, containing one single integer, the maximum total number of clients that could have been inside the dinner halls in a given instant of time.

Sample input

```
4
07:22:03 X
07:13:22 E
08:30:51 E
21:59:02 X
4
09:00:00 E
20:00:01 X
09:05:00 ?
20:00:00 ?
8
10:21:00 E
10:25:00 X
10:23:00 E
10:24:00 X
10:26:00 X
10:27:00 ?
10:22:00 ?
10:20:00 ?
0
```

Output for the sample input

```
1
2
4
```

Problem E
Electric Bill

File code name: `electric`

It's year 2100. Electricity has become very expensive. Recently, your electricity company raised the power rates once more. The table below shows the new rates (consumption is always a positive integer):

Range (Crazy-Watt-hour)	**Price (Americus)**
$1 \sim 100$	2
$101 \sim 10000$	3
$10001 \sim 1000000$	5
> 1000000	7

This means that, when calculating the amount to pay, the first 100 CWh have a price of 2 Americus each; the next 9900 CWh (between 101 and 10000) have a price of 3 Americus each and so on.

For instance, if you consume 10123 CWh you will have to pay $2\times100+3\times9900+5\times123 = 30515$ Americus.

The evil mathematicians from the company have found a way to gain even more money. Instead of telling you how much energy you have consumed and how much you have to pay, they will show you two numbers related to yourself and to a random neighbor:

A: the total amount to pay if your consumptions were billed together; and

B: the absolute value of the difference between the amounts of your bills.

If you can't figure out how much you have to pay, you must pay another 100 Americus for such a "service". You are very economical, and therefore you are sure you cannot possibly consume more than any of your neighbors. So, being smart, you know you can compute how much you have to pay. For example, suppose the company informed you the following two numbers: $A = 1100$ and $B = 300$. Then you and your neighbor's consumptions had to be 150 CWh and 250 CWh respectively. The total consumption is 400 CWh and then A is $2\times100+3\times300 = 1100$. You have to pay $2\times100+3\times50 = 350$ Americus, while your neighbor must pay $2 \times 100 + 3 \times 150 = 650$ Americus, so B is $|350 - 650| = 300$.

Not willing to pay the additional fee, you decided to write a computer program to find out how much you have to pay.

Input

The input contains several test cases. Each test case is composed of a single line, containing two integers A and B, separated by a single space, representing the numbers shown to you

($1 \leq A, B \leq 10^9$). You may assume there is always a unique solution, that is, there exists exactly one pair of consumptions that produces those numbers.

The last test case is followed by a line containing two zeros separated by a single space.

Output

For each test case in the input, your program must print a single line containing one integer, representing the amount you have to pay.

Sample input	**Output for the sample input**

`1100 300`
`35515 27615`
`0 0`
	`350`
	`2900`

Problem F
File Recover

File code name: `file`

Your school has a computer that is used as a web server for hosting its institutional web site, personal pages of the staff, sites for research groups, subjects, and many others.

Recently, the hard disk table was corrupted, so the organization of all the files was lost. Sadly enough, there are no backups of that information. The only hope is to look through the entire disk data and try to find out which parts correspond to each file. Fortunately, the disk was using a file system that kept each individual file contiguous, so only contiguous pieces of data need to be inspected.

The disk data is a sequence of bytes. Each byte in this particular disk can store an English alphabet letter (distinguishing lowercase and uppercase), a decimal digit, a point or a comma, making a total of 64 different characters.

While you were thinking about how to solve the problem, you suddenly remembered that the file system also maintained multiple copies of each file, so only the pieces of contiguous bytes that are repeated had a chance of being a file. Moreover, for each repetition of the same contiguous bytes, only one copy needs to be checked. For instance, if the data is '*ababcabb*', the contiguous subsequences '*a*', '*b*' and '*ab*' are repeated, but nothing containing '*c*', nor '*ba*' or '*bb*' is. Therefore, we have 3 pieces of contiguous bytes that need checking in this case.

You decide to write a program that computes exactly how many sequences need checking, that is, the number of different sequences of contiguous bytes that appear at least twice in the data.

Input

There are several test cases. The input of each test case is given in exactly one line, containing a non-empty string of at most 10^5 characters that represents the disk data. Each character in the string is either a lowercase letter, an uppercase letter, a digit, a point or a comma.

The last test case is followed by a line containing a single asterisk.

Output

For each test case output a single line with an integer, representing the number of different contiguous subsequences that appear at least twice in the input string.

Sample input	Output for the sample input
`ababcabb`	`3`
`mississippi`	`9`
`aaaaaaaaaaaaaaaaaaaaaaaaa`	`25`
`012345678,abcdefg.STUVWXYZ`	`0`
`say.twice,say.twice`	`45`
`*`	

Problem G
Grapevine

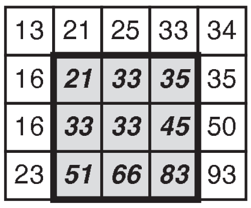
File code name: `grape`

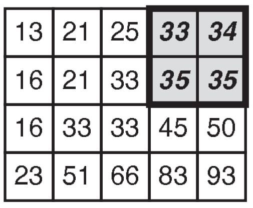
In Quadradonia, all rural properties are square, all have the same area, all are perfectly flat and all have the sides aligned to the North-South and West-East axes.

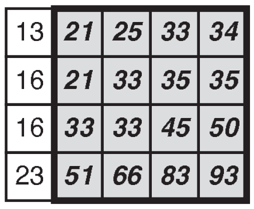
Since properties are flat, the hills in Quadradonia look like a series of huge stairs' steps, with different heights. In a certain mountain, an interesting situation occurs in a rectangular area of $N \times M$ properties. Starting from anywhere within the region, traversing it in the West to East direction, the properties have non-descending heights. Similarly, traversing that region in the North to South direction, starting from anywhere, the properties have also non-descending heights.

A large wine company in Quadradonia wants to rent some properties from that region to grow wine grapes. The company is interested in some special varieties of wine grapes, which are productive only if grown in properties whose heights are within a certain interval. That is, the company is interested in renting properties whose heights are equal to or higher than a given altitude L, and equal to or lower than a given altitude U. To make it easier for harvesting, the rented properties must form a contiguous area. And since everyone in Quadradonia likes squares, the area to be rented must have the shape of a square.

The company has not yet decided which variety of grapes it will grow, and therefore it has a list of queries involving intervals, one for each grape variety. The figure below shows an area of interest of dimensions 4×5 (in number of properties) with examples of areas the company could rent to grow grapes in heights within the intervals given in the picture.


interval=[20,90]


interval=[33,35]


interval=[20,100]

You must write a program that, given the description of the rectangular area of interest in the mountain, and a list of queries containing height intervals, determines, for each query, the largest side, in number of properties, of a contiguous square area with heights within the specified interval.

Input

The input contains several test cases. The first line of a test case contains two integers N and M, separated by a single space, representing respectively the number of properties in the North-South direction ($1 \leq N \leq 500$) and the number of properties in the West-East direction

($1 \leq M \leq 500$) of the region of interest. Each of the next N lines contains M integers $H_{i,j}$, separated by single spaces, indicating the heights of the properties in the region of interest ($0 \leq H_{i,j} \leq 10^5$, for $1 \leq i \leq N$ and $1 \leq j \leq M$; also, $H_{i-1,j} \leq H_{i,j}$ and $H_{i,j-1} \leq H_{i,j}$). The next line contains an integer Q indicating the number of queries ($1 \leq Q \leq 10^4$). Each of the next Q lines describes a query, and contains two integers L and U, separated by a single space, indicating one interval of heights ($0 \leq L \leq U \leq 10^5$). The heights of properties to be rented must be greater than or equal to L and less than or equal to U.

The last test case is followed by a line containing two zeros separated by a single space.

Output

For each test case in the input your program must print $Q + 1$ lines. Each of the first Q lines must contain a single integer, indicating the largest side, in number of properties, of a contiguous square area with heights within the interval specified in the respective input query. The last line to be printed for each test case is used as a separator and must contain a single character '`-`' (known as hyphen or minus sign).

Sample input	**Output for the sample input**

```
4 5
13 21 25 33 34
16 21 33 35 35
16 33 33 45 50
23 51 66 83 93
3
22 90
33 35
20 100
4 4
1 7 9 11
5 8 10 12
7 10 15 17
11 19 30 41
4
6 20
7 9
10 10
13 14
0 0
```
	```
	3
	2
	4
	-
	3
	1
	1
	0
	-
	```

Problem H
Hooligan

File code name: `hooligan`

Soccer is the American English word used to describe Football, the British English word applied to the most popular sport in Latin America (and in the world). Hooligan is sometimes used to describe an aggressive, troublemaking, soccer fan.

In Linearonia, a soccer tournament is in progress. There, the ranking process is as follows: for each game, the winner gets two points and the loser gets no points; in case of a tie, both competitors get one point each. The champion is the team with the highest number of points. Every pair of distinct teams play against each other exactly the same number of times, called the *matching number.*

You have your favorite team, your *dream team*, and you wonder whether it is possible for your dream team to be champion. You know the number of teams, the matching number and the results of some games that have already been played. Write a program to decide whether at the end of the tournament your dream team can still be the *only* champion, with strictly more points than any other team.

Input

The input contains several test cases, each case consists of one or more lines. The first line contains three integers, N, M and G, separated by single spaces, representing respectively the number of teams playing in the tournament ($2 \leq N \leq 40$), the matching number ($1 \leq M \leq 4$) and the number of games already played ($1 \leq G$). Your dream team is identified by the number 0, the other teams are identified by the integers $1, 2, \ldots, N-1$.

Each of the next G lines describes a game already played. The line contains an integer I, a character C and an integer J, separated by single spaces. Integers I and J are the teams that played that game ($I \neq J$ and $0 \leq I, J \leq N-1$). Character C is '`<`' if team I lost to team J, or '`=`' if the game ended in a tie.

The last test case is followed by a line containing three zeros separated by single spaces.

Output

For each test case in the input, your program must print a single line, containing one single character, uppercase '`Y`' if your dream team can be the champion, or uppercase '`N`' otherwise.

Sample input	Output for the sample input

```
4 2 6
0 < 3
3 = 2
2 < 0
1 < 0
2 = 0
3 < 0
4 1 5
2 = 0
0 < 1
1 = 3
2 < 1
0 < 3
4 2 5
2 = 0
0 < 1
1 = 3
2 < 1
0 < 3
2 1 1
1 < 0
4 1 1
0 < 1
4 1 2
0 < 1
0 < 2
0 0 0
```
	```
	Y
	N
	Y
	Y
	Y
	N
	```

Problem I
Isosceles Triangles

File code name: `isosceles`

A given triangle can be either equilateral (three sides of the same length), scalene (three sides of different lengths), or isosceles (two sides of the same length and a third side of a different length). It is a known fact that points with all integer coordinates cannot be the vertices of an equilateral triangle.

You are given a set of different points with integer coordinates on the XY plane, such that no three points in the set lay on the same line. Your job is to calculate how many of the possible choices of three points are the vertices of an isosceles triangle.

Input

There are several test cases. Each test case is given in several lines. The first line of each test case contains an integer N indicating the number of points in the set ($3 \leq N \leq 1000$). Each of the next N lines describes a different point of the set using two integers X and Y separated by a single space ($1 \leq X, Y \leq 10^6$); these values represent the coordinates of the point on the XY plane. You may assume that within each test case no two points have the same location and no three points are collinear.

The last test case is followed by a line containing a single zero.

Output

For each test case output a single line with a single integer indicating the number of subsets of three points that are the vertices of an isosceles triangle.

Sample input	**Output for the sample input**

```
5
1 2
2 1
2 2
1 1
1000 1000000
6
1000 1000
996 1003
996 997
1003 996
1003 1004
992 1000
0
```
	```
	4
	10
	```

Problem J
Jingle Composing

File code name: `jingle`

A. C. Marcos is taking his first steps in the direction of jingle composition. He is having some troubles, but at least he is achieving pleasant melodies and attractive rhythms.

In music, a *note* has a pitch (its frequency, resulting in how high or low is the sound) and a duration (for how long the note should sound). In this problem we are interested only in the duration of the notes.

A jingle is divided into a sequence of *measures*, and a measure is formed by a series of notes.

The duration of a note is indicated by its shape. In this problem, we will use uppercase letters to indicate a note's duration. The following table lists all the available notes:

Notes	𝅝	𝅗𝅥	♩	♪	𝅘𝅥𝅯	𝅘𝅥𝅰	𝅘𝅥𝅱
Identifier	W	H	Q	E	S	T	X
Duration	1	1/2	1/4	1/8	1/16	1/32	1/64

The duration of a measure is the sum of the durations of its notes. In Marcos' jingles, each measure has the same duration. As Marcos is just a beginner, his famous teacher Johann Sebastian III taught him that the duration of a measure must always be 1.

For example, Marcos wrote a composition containing five measures, of which the first four have the correct duration and the last one is wrong. In the example below, each measure is surrounded with slashes and each note is represented as in the table above.

```
/HH/QQQQ/XXXTXTEQH/W/HW/
```

Marcos likes computers as much as music. He wants you to write a program that determines, for each one of his compositions, how many measures have the right duration.

Input

The input contains several test cases. Each test case is described in a single line containing a string whose length is between 3 and 200 characters, inclusive, representing a composition. A composition begins and ends with a slash '`/`'. Measures in a composition are separated by a slash '`/`'. Each note in a measure is represented by the corresponding uppercase letter, as described above. You may assume that each composition contains at least one measure and that each measure contains at least one note. All characters in the input will be either slashes or one of the seven uppercase letters used to represent notes, as described above.

The last test case is followed by a line containing a single asterisk.

Output

For each test case your program must output a single line, containing a single integer, the number of measures that have the right duration.

Sample input	**Output for the sample input**
`/HH/QQQQ/XXXTXTEQH/W/HW/`	`4`
`/W/W/SQHES/`	`3`
`/WE/TEX/THES/`	`0`
`*`	

Problem K
Klingon Levels

File code name: `klingon`

In a Latin American high school, the klingon language has become so popular that many of the students have begun learning this artificial language on their own. After becoming aware of the situation, the directors have decided to implement formal klingon courses. The problem is that kids have different starting levels of knowledge of the language. Therefore, the directors decided to offer two course levels: basic and advanced.

The school has several divisions, with each student belonging to exactly one division. Because of bureaucracy and schedule conflicts, students of different divisions cannot be in the same klingon course. Also, to be fair, the basic and advanced klingon levels should be offered to all divisions, and have the same level of difficulty among the divisions.

Therefore, each division will be partitioned into two groups: one group will be assigned a basic level course, and the other group an advanced level course. It is possible, also, that a division does not contain any students in one of the levels.

To assign the levels, a klingon test has been previously taken by all students of the school, each getting an integer grade between 0 and 1000, inclusive. To accomplish the aforementioned goals, the school directors have decided that all students with a score greater than or equal to some T will be assigned the advanced level, and all students with a score less than T will be assigned the basic level.

However, they cannot decide on the best value of T. They would like a value that evenly splits all divisions. For this, they came up with a metric: They want the value of T that minimizes the accumulated difference, that is, the sum of the difference between the number of students in the two groups (basic and advanced) within each division.

For example, if the school has two divisions, where one division has 10 students in the basic level and 20 in the advanced level, while the other one has 17 and 15, respectively, the accumulated difference would be $|10 - 20| + |17 - 15| = 12$.

Input

There are several test cases. Each test case is given in several lines. The first line of each test case contains a single integer N ($1 \leq N \leq 10^4$), the number of divisions in the school. $2 \times N$ lines follow, with each division being described in two consecutive lines. The first line of each group of two contains a single integer K_i ($1 \leq K_i \leq 10^4$) the number of students in division i. The second line contains K_i integers between 0 and 1000, inclusive, separated by single spaces, representing the scores of each of the students in division i. You may assume that the total number of students within each test case (that is, the sum of the values of all K_i) is not greater than 10^5.

The last test case is followed by a line containing a single zero.

Output

For each test case, output a single line with a single integer representing the minimum value for the accumulated difference if T is chosen optimally.

Sample input

```
2
2
1 2
2
3 4
2
2
1 4
2
2 3
3
4
1 10 100 1000
3
5 55 555
5
4 16 64 256 1000
1
4
500 500 500 500
0
```

Output for the sample input

```
2
0
2
4
```