The 2006 ACM ASIA Programming Contest Dhaka Site

Sponsored by IBM
Hosted by North South University
Dhaka, Bangladesh

22nd and 23rd September, 2006
You get 17 Pages
10 Problems
&
300 Minutes

A Mobile Casanova

Input: a.in
Output: Standard Output

Mobile Phone industry is the fastest growing industry in Wonderland. The three prominent mobile service providers of Wonderland are Coktel, Anglolink and Grinding Phone. All three operators provide very cheap rate packages after 12:00 AM. The name of their packages are "The Thief Talk" ("Chorer Alap" in Bangla), "Insomnia" and "The Vampire Chat" respectively. These cheap packages have inspired many people to disturb others at Night.

One such gone to ashtray guy is Arif whose activities begin after 12:00 AM. He dials a number randomly and writes down it in his notebook if he likes it. Later he dials those numbers written in his notebook frequently. As names are not important to him so his note book contains only numbers, and no names. But in a few days his notebook becomes filled with many numbers. So now he wants to reorganize the numbers in a new notebook. The reorganizing process is described below:

Arif's notebook contains many numbers so there may exist some consecutive numbers. Therefore he wants to save space by writing the consecutive numbers as a range. For example if there are three numbers 01711322396, 01711322397, 01711322398 then he will write them as 01711322396-8. It means that the smallest number in the sequence is the first part of the range followed by a hyphen and then only the rightmost digits of the largest number that are different than the first number. A phone number can have any numerical value within 100 and 2000000000 (inclusive) and is always preceded by only a single zero. So 01123 is a valid phone number but 00123, 089, 1234, 02000000001 are invalid phone numbers.

You will be given up to 100000 sorted phone (As you can't sort numbers) numbers and your job is to write a program that does the job for Arif.

Input

The input file contains 110 sets of inputs. But only about 10% of them are large inputs. The description of each set is given below:

Each set starts with an integer N ($0 \leq N \leq 100000$) which denotes how many phone numbers are there in this set. Each of the next N lines contains a valid phone number. The phone numbers will be sorted in ascending order according to their numerical values. You can assume that all input phone numbers will be valid and distinct. You can also assume that all phone numbers in a single set will have equal length.

Input is terminated by a set where the value of N=0. This set need not be processed.

Output

For each set of inputs produce two or more lines of outputs. The description of output for each set is given below.

First line contains the serial of output. Next lines contain the phone numbers printed according to the rule mentioned in the problem statement. The numbers which are not part of a consecutive sequence are printed exactly as the input and the consecutive number sequences are printed as range. Print a blank line after the output for each set of input. Look at the output for sample input for details. Remember that it needs at least two consecutive numbers to form a consecutive sequence.

Sample Input	Output for Sample Input
3 01711322396 01711322397 01711322398 7 01187239192 01711322396 01711322397 01711322398 01711322399 01711322400 01711389821 0	Case 1: 01711322396-8 Case 2: 01187239192 01711322396-400 01711389821

B Potentiometers

Input: b.in
Output: Standard Output

A potentiometer, or potmeter for short, is an electronic device with a variable electric resistance. It has two terminals and some kind of control mechanism (often a dial, a wheel or a slide) with which the resistance between the terminals can be adjusted from zero (no resistance) to some maximum value. Resistance is measured in Ohms, and when two or more resistors are connected in series (one after the other, in a row), the total resistance of the array is the sum of the resistances of the individual resistors.

In this problem we will consider an array of **N** potmeters, numbered **1** to **N** from left to right. The left terminal of some potmeter numbered **x** is connected to the right terminal of potmeter **x-1**, and its right terminal to the left terminal of potmeter **x+1**. The left terminal of potmeter **1** and the right terminal of potmeter **N** are not connected.

Initially all the potmeters are set to some value between 0 and 1000 Ohms. Then we can do two things:

- Set one of the potmeters to another value.
- Measure the resistance between two terminals anywhere in the array.

Input

The input consists less than 5 cases. Each case starts with **N**, the number of potmeters in the array, on a line by itself. **N** can be as large as 200000. Each of next N lines contains one numbers between 0 and 1000, the initial resistances of the potmeters in the order **1** to **N**. Then follow a number of actions, each on a line by itself. The number of actions can be as many as 200000. There are three types of action:

- "S x r" - set potmeter **x** to **r** Ohms. **x** is a valid potmeter number and **r** is between 0 and 1000.
- "M x y" - measure the resistance between the left terminal of potmeter **x** and the right terminal of potmeter **y**. Both numbers will be valid and **x** is smaller than or equal to **y**.
- "END" - end of this case. Appears only once at the end of a list of actions.

A case with **N**=0 signals the end of the input and it should not be processed.

Output

For each case in the input produce a line "Case n:", where n is the case number, starting from 1. For each measurement in the input, output a line containing one number: the measured resistance in Ohms. The actions should be applied to the array of potmeters in the order given in the input. Print a blank line between cases.

Sample Input

```
3
100
100
100
M 1 1
M 1 3
S 2 200
M 1 2
S 3 0
M 2 3
END
10
1
2
3
4
5
6
7
8
9
10
M 1 10
END
0
```

Output for Sample Input

```
Case 1:
100
300
300
200

Case 2:
55
```

C Collecting Marbles

Input: c.in
Output: Standard Output

There is a grid of r rows and c columns. The rows are numbered from 1 to r and the columns are numbered from 1 to c. The upper left cell is in row 1 and column 1. The lower right cell is in row r and column c. A cell(p,q) denotes the cell in row p and column q in the grid. A subgrid (r_1, c_1, r_2, c_2) is a part of the grid that contains all the cells from rows r_1 to r_2 and columns c_1 to c_2 (inclusive). In one unit of time you can move one marble from the cell(p,q) to any of the following 4 cells: cell(p-1,q), cell(p+1,q), cell(p,q-1), cell(p,q+1). You will be given the information of a grid. Then you will be given some subgrids. For each subgrid your task is to calculate the minimum amount of time needed to move all the marbles to any of the cells in that sub grid.

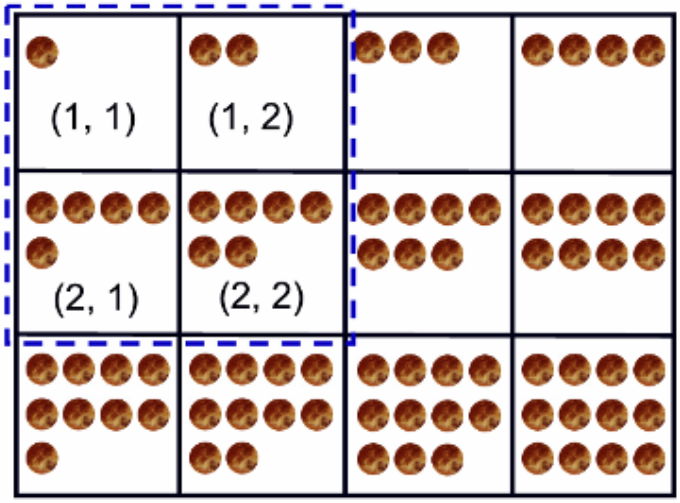


Figure: The figure above corresponds to the first sample input. The dashed rectangle shows the subgrid (1, 1, 2, 2). If we move all the marbles of this subgrid to cell (2, 2) the total cost will be (1*2+2*1+5*1+6*0)=9. And this is the minimum possible cost, because the cost of moving all the marbles to (1, 1), (1, 2) and (2, 1) is 19, 17 and 11 respectively. This example corresponds to the second query of first sample input.

Input

First line of the input contains an integer T ($T \leq 5$) the number of test cases. Each of the test cases begins with three integers r, c ($1 \leq r, c \leq 500$) and q ($0 \leq q \leq 10000$) in one line. Here r is the number of rows, c is the number of columns and q is the number of queries. Each of the next r lines contains c integers. The j'th integer in the i'th line contains the number of marbles in the cell(i,j). All these numbers are non- negative and less than 1001. Each of the next q lines contains 4 integers: r_1, c_1, r_2, c_2. These 4 integers denote the sub grid (r_1, c_1, r_2, c_2). You can obviously assume that ($1 \leq r_1, r_2 \leq r$ and $1 \leq c_1, c_2 \leq c$)

Output

For each test case you have to produce q+1 lines of output. The description of output for each test case is given below:

First line of each test case contains the serial of that test case. Each of the next q line contains output for one query of that test case. Output for each query contains two integers separated by a single space. The first integer denotes the serial of the query and the second integer denotes the minimum time required to move the marbles within the query subgrid to one of the cells within the subgrid. Print a blank line after each test case.

Sample Input	Output for Sample Input
2 3 4 3 1 2 3 4 5 6 7 8 9 10 11 12 1 1 3 4 1 1 2 2 1 1 3 3 3 3 3 2 1 3 4 6 1 11 2 3 1 1 3 3 1 1 2 3 2 1 3 3	Test Case 1: 1 118 2 9 3 66 Test Case 2: 1 45 2 16 3 27

D Expression

Input: d.in
Output: Standard Output

Professor Conrad loves to work with expressions and we all know that there are many well known techniques to evaluate expressions. After teaching his class about evaluating expression he gives his class a bit tricky assignment. As usual being the most brilliant programmer of the class, your classmates have grasped you to do the assignment for them. The assignment is stated below:

You are given an expression consisting only brackets, blank spaces and +/- operators and also some digits. You have to place the given digits in the blank spaces in such a way so that the value of the expression is maximum.

Input

First line of the input file contains an integer N (N<=505) which denotes how many sets of inputs will be there. The description of each set is given below.

First line of each set contains a correct expression and the second line contains a sequence of digits. You can put any of this one of these digits in the blank spaces of the expression. In the input the blank spaces are denoted with '#' characters. You can assume that (a) The number of '#' signs in the expression and the number of given digits will be equal (b) There will be maximum eight consecutive '#' signs in the given expression (c) The length of the expression will be less than 200

Output

For each set of input produce three lines of output. First line contains the serial of output, the second line contains the expression which produces maximum value and the third line contains this maximum value. If more than one expression produces maximum value then output the lexicographically smallest one. You can assume that (a) the maximum value of the expression will fit in a 32-bit signed integer (b) Numbers in the expression can have leading zeroes.

Sample Input	Output for Sample Input
3 ##-(##+###) 3333333 #+# 31 #-# 31	Case 1: 33-(33+333) -333 Case 2: 1+3 4 Case 3: 3-1 2

E The Luncheon

Input: e.in
Output: Standard Output

Karim and Rahim go to the same restaurant once every day to have lunch. They both have their own list of favorite dishes but they don't necessarily eat their favorite dishes at lunch. The restaurant they visit have **N** dishes in total. Karim and Rahim choose a dish randomly (They both eat the same dish) from the **N** dishes and have it for lunch. They go to that restaurant regularly for **D** days. You have to find the probability that the number of days for which the dish chosen is a favorite of Rahim's is exactly twice the number of days for which the dish chosen is a favorite of Karim's.

Input

The input file contains at most **600** sets of inputs. The description of each set is given below:

Each set starts with an integer N (0<N<37) which denotes the total number of dishes. Dishes are identified by integers from 1..N. Next line starts with an integer S (0≤S<N) which denotes the total number of Karim's favorite dishes. Next S distinct integers are favorite dishes of Karim. All these integers are within the range (1..N). The Next line starts with an integer T (0≤T<N) which denotes the total number of Rahim's favorite dishes. Next T distinct integers are favorite dishes of Rahim. All these integers are also within the range (1..N). The next line contains an integer D (0<D<101) which indicates how many days Karim and Rahim goes to the restaurant.

Input is terminated by a line containing a single zero.

Output

For each set of input produce one line of output. This line contains the serial of output followed by the desired probability rounded to five digits after the decimal point. Look at the output for sample input for details. There is no special judge for this problem. But the judge data is such that errors less than $2*10^{-7}$ will be ignored. But please make sure when the answer very near to zero you print it as "0.00000" and not as "-0.00000".

Sample Input

```
6
3 1 2 3
3 4 5 6
18
7
3 1 2 3
3 4 5 6
20
8
3 1 2 3
3 2 3 4
20
0
```

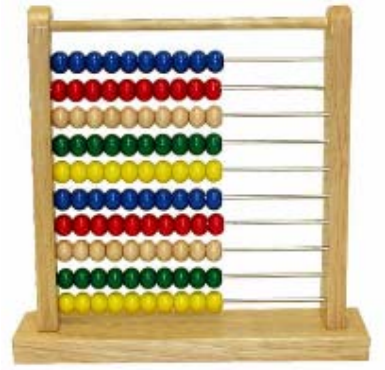
Output for Sample Input

```
Case 1: 0.07082
Case 2: 0.02592
Case 3: 0.01501
```

F Counting Zeroes

Input: f.in
Output: Standard Output

In our daily life we often meet some stupid persons. They tend to do things by hand which can only be done with computers. Our friend Hashmat is such a person and now he is counting with abacus, pencil and paper the summation of trailing zeroes of a particular number in all possible number systems (Of course, I am talking about the common number system like decimal, binary, hexadecimal or n-based number system).



We the average intelligent peoples know what a number system is. Our daily life number system is the decimal (10-based) number system, our corporate life number system is the binary (2-based) number system. A number may or may not have trailing zeroes in certain number system. For example in decimal number system $102\mathbf{0}_{10}$ (It means 1020 is a number whose base is 10) has 1 trailing zero but in hexadecimal number system $102\mathbf{0}_{10}$ has no trailing zeroes. Let us create a function *fzero(n,b)*, which actually denotes the number of trailing zeroes of n in b-based number system. So fzero($102\mathbf{0}_{10}$,10)=1 and fzero($102\mathbf{0}_{10}$,16)=0. Although for a certain number and certain base the task is trivial but it is very time consuming to do it for all possible number systems with the help of only pencil and paper. So your job is now to help Hashmat, so that he can complete this task and do something meaningful in life. In other words given the value of n your job is to find out the value of:

$$\sum_{b=2}^{\infty} fzero(n,b)$$

Input

The input file contains at most 800 lines of inputs. Each line contains a decimal integer n ($0<n<=10^{14}$).

Input is terminated by a line containing a single zero.

Output

For each line of input produce one line of output. This line contains the input number followed by the value of $\sum_{b=2}^{\infty} fzero(n,b)$.

Sample Input	Output for Sample Input
10 20 0	10 3 20 6

G Multiplication

Input: g.in
Output: Standard Output

Multiplying factors is one of the most tiresome jobs of algebra, so now you might want to take the help of computers to do that. In this problem you will have to deal with expression of the form $(x+a_1)(x+a_2)(x+a_3)...(x+a_{n-1})(x+a_n)$, given the value of n. To be more specific for n=2 you have to find the length of the output $x^2+x(a_1+a_2)+a_1a_2$, for n=3 you have to find the length of the output $x^3+x^2(a_1+a_2+a_3)+x(a_1a_2+a_1a_3+a_2a_3)+a_1a_2a_3$ and so on. In plain text mode this type of strings can be outputted in three lines as:

```
1234567890123456789012345678901234567890 //This is not output
 3  2
x +x (a +a +a )+x(a a +a a +a a )+a a a
       1  2  3     1 2  1 3  2 3   1 2 3
```

So the length of the result is 40 when n=3. As with growth of n the length grows very fast so you don't need to output the length but output modulo 10000 value of the length only. For your convenience you might want to notice that for n=10 the initial part of the multiplication answer is:

```
 10  9                                    8
x  +x (a +a +a +a +a +a +a +a +a +a  )+x (a a +a a +a a +a a +a a +a a + a a +a a +a a  )+ …
       1  2  3  4  5  6  7  8  9  10     1 2  1 3  1 4  1 5  1 6  1 7   1 8  1 9  1 10
```

Input

The input file contains at most 6005 lines of inputs. Each line consists of a single integer n ($0<n\leq1000000000$). Here n means the number of factors that are to be multiplied.

Input is terminated by a line where the value of n is zero. This line should not be processed.

Output

For each line of input produce one line of output. If the length of the multiplied string is t then for each input you should output the serial of output followed by t % 10000. Please note that you don't need to put any unnecessary brackets and also note that you don't need to output terms like x^1. Show the power of x only when it is greater than 1.

Sample Input	Output for Sample Input
1 2 3 4 5 8 12 0	Case 1: 4 Case 2: 16 Case 3: 40 Case 4: 92 Case 5: 208 Case 6: 2332 Case 7: 9439

H Paint the Roads

Input: h.in
Output: Standard Output

In a country there are n cities connected by m one way roads. You can paint any of these roads. To paint a road it costs d unit of money where d is the length of that road. Your task is to paint some of the roads so that the painted roads can be partitioned into some disjoint cycles such that every vertex appears in exactly k of these disjoint cycles. But you have to minimize the costs of painting these roads.

Input

First line of the input contains T the number of test case. Then following lines contains T Test cases.

Each case starts with a line containing 3 integers n ($1 \le n \le 40$), m ($1 \le m \le 2000$) and k ($1 \le k$ and $1 \le k*n \le 100$). Next m lines contain description of m roads. Each line contains three integers f, t ($0 \le f, t < n$ and $f \ne t$) and d ($0 \le d < 100$). That means there is a road of d length from city f to city t. You can assume that there will be at most one road in one direction between two cities.

Output

For each test case output contains 1 integer denoting the minimum unit of money needed to paint roads. In the case it is impossible to paint the roads maintaining the constraints output -1.

Sample Input	Output for Sample Input
4 4 8 1 0 1 1 1 0 2 2 3 1 3 2 2 0 2 5 2 0 6 1 3 5 3 1 6	6 4 28 -1

4 8 1 0 1 1 1 0 10 2 3 10 3 2 1 0 2 10 2 0 1 1 3 1 3 1 10 4 8 2 0 1 1 1 0 2 2 3 1 3 2 2 0 2 5 2 0 6 1 3 5 3 1 6 3 4 1 0 1 5 1 0 6 0 2 7 2 0 8	

I Protecting Zonq

Input: i.in
Output: Standard Output

In a star system far, far away, there are two civilised planets: Zonq and Clunq. Zonq is a lovely planet with plenty of natural resources and its inhabitants, the zonq-ians, developed into a peace-loving people, with high moral standards and a rich culture of arts, sciences and all things nice. Clunq, on the other hand, is a cold, dark planet where the resources are scarce, and its inhabitants, the clunq-ons, struggle for their existence from the day they are born. It is no wonder they developed into strong fighters, and martial arts are their main form of culture.

As always happens in the universe, at some point in their development the clunq-ons invented space exploration, and now they are on the verge of loading their space-ships with warriors to conquer Zonq. Although the zonq-ians developed mathematics to a very high level (they factorize million digit numbers for breakfast and the proof of Fermat's Last Theorem is taught in elementary school), they never bothered to tire themselves with such mundane things as computers and informatics. So now they need your help to defend the planet.

Zonq consists of a number of villages connected by roads. Since the zonq-ians hate physical labour, they built just enough roads to insure that all villages are mutually connected, directly or indirectly, but not one road more. When the clunq-ons invade, they will always land on a road, halfway between two villages, and spread out from there. To defend the roads, the zonq-ians can place guard robots in their villages. There are two types: soldier-bots and sergeant-bots. A soldier-bot, when placed in a village, protects all roads connected to that village. A sergeant-bot is more powerful: it protects the roads connected to the village it is placed in, but also all roads connected to the villages that are direct neighbors to it. Both types of robot come at a price, and it's your task to assign robots to villages such that: a) all roads are protected, b) the total cost of the robots is as low as possible.

Input

There are several scenarios. Each scenario starts with three numbers on a line by it self: N ($1 \leq N \leq 10000$), the number of villages, C1 ($0 \leq C1 \leq 1000$), the cost of a soldier-bot, and C2 ($0 \leq C2 \leq 1000$), the cost of a sergeant-bot. For the sake of abstraction, the villages are numbered from 1 to N. Then follow N-1 lines containing two numbers V1 ($1 \leq V1 \leq N$) and V2 ($1 \leq V2 \leq N$)., each defining a road between two villages, numbered V1 and V2, respectively. No road is mentioned twice, and all roads together span all villages.

A line with three zeros marks the end of the input and should not be processed.

Output

For each scenario in the input, print just one number on a line by it self: the minimal cost of protecting Zonq.

Sample Input	Output for Sample Input
5 30 50 1 2 2 3 3 4 4 5 9 20 30 1 2 2 3 3 4 4 5 4 8 5 6 5 7 8 9 6 100 500 1 3 2 3 3 4 4 5 4 6 0 0 0	50 50 200

PS. How were the zonq-ians able to utilize robots, if they never bothered to build computers? Well, that's a stupid question: they called 0800-RENT-A-BOT, of course! How else do you think they hired you, in the first place?

J Battle of the Triangle

Input: j.in
Output: Standard Output

It is now year 2020, people don't install games in their hard disk now as they only play online games. "Battle of the triangle" has evolved as the most popular online game now. The game is played between two armies: One army is controlled by 100 players from one city and another army is controlled by another 100 player from another city.

At the start of the game there is a toss. The winner of the toss is marked Team A and the loser is marked team B. The battle field is already generated for the two teams. Battle field means a two dimensional plane where positions of the soldiers are given by two dimensional Cartesian Coordinate System and position of the tanks are also given by two dimensional Cartesian Coordinate System. There are three boundary lines which create a triangle and everything inside the triangle belongs to Team A and everything in the reciprocal regions of the triangle belongs to team B, as shown in the picture below. The other three regions are neutral regions. Team A can move the boundary lines to select his own triangle. But of course he cannot rotate the boundary lines.

There is a central server where the players have to connect to play this game and it is already mentioned that each team consists of 100 players. Each team has a Captain. The captain of Team A initially selects the three boundary lines. When the boundary lines are selected the server has to give the Captain of Team A two information: The difference between soldier number of Team A and Team B and the difference between the tank number of Team A and Team B so that the captain can rethink his strategy. But as many teams all around the world play at the same time using the same battle field so the server has to answer many queries simultaneously. Your job is to write a program that does this job very efficiently.

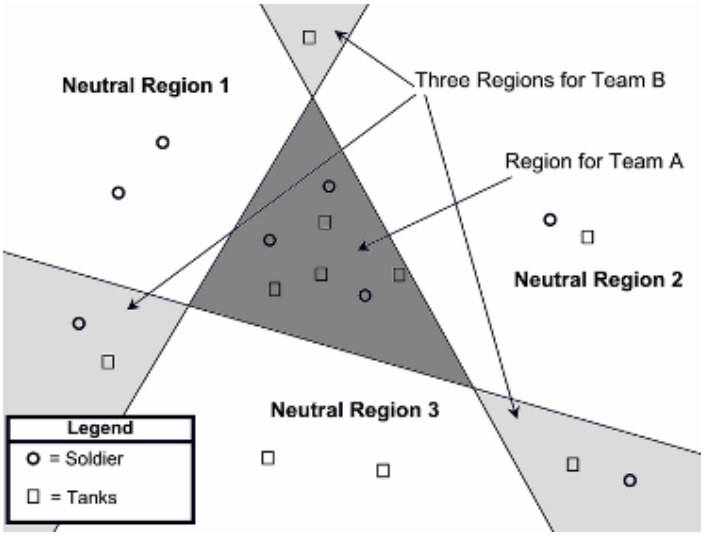


Figure: Dark Gray is the region of A, Light Gray is the region of B and white is the neutral region.

Input

The input file contains several sets of inputs. The description of each set is given below:

Each set starts with three integers S, T and Q. Here S ($0<S\leq50000$) is the total number of soldiers in the map, T ($0<T\leq50000$) is the total number of Tanks in the map and Q ($0<Q\leq10000$) is the total number of queries the server has to handle. Each of the next S lines contain two integers x_i, y_i ($|x_i|$, $|y_i|\leq10000000$) which denotes the Cartesian coordinates of the soldiers. Each of the next T lines

contains two integers x_j, y_j ($|x_j|$, $|y_j| \le 10000000$) which denotes the coordinates of the tanks. These lines are followed by 3Q lines as each query consists of three lines of inputs. First line contains three integers A_1, B_1, C_1, second line contains three integers A_2, B_2, C_2 and third line contains three integers A_3, B_3, C_3. These denotes that captain of team A has placed the first boundary line along the straight line $A_1x + B_1y + C_1 = 0$, the second boundary line along the straight line along the straight line $A_2x + B_2y + C_2 = 0$ and the third boundary line along the straight line $A_3x + B_3y + C_3 = 0$. So the region for team A is the triangle formed by these three straight lines and the region for team B is three reciprocal regions of this triangle. For each query you have to find the soldier number difference between team A and B and tank number difference between team A and team B. The figure below corresponds to the sample input. You can of course assume that in a set of input the first line of each query are parallel to one another, and this is true for second lines of each query and third line of each query as well. You can also assume that ($|A_1|, |B_1|, |A_2|, |B_2| \le 1000$), ($|C_1|, |C_2| \le 10000000$), no two lines in a query are parallel each other, the three lines in a query are not concurrent and no soldier and no tank will be on any of the three boundary lines.

Input is terminated by a case where S=T=Q=0. This case should not be processed. The input file size is around 10 MB. This will give you some idea on how efficient your program needs to be.

Output

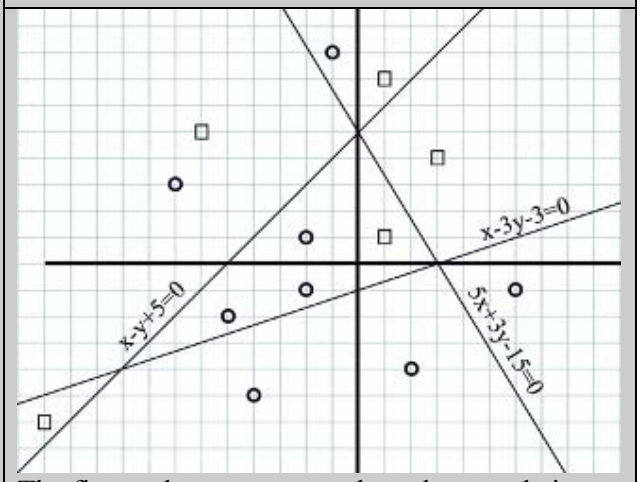
For each set of input produce (Q+1) lines of outputs. First line contains the battle field serial. Each of the next Q lines contains output for one query. For each query output in a single line the serial of the query followed by two integers DS and DT separated by a single space. Here DS is the difference of number of soldiers between team A and team B, and DT is the difference of number of tanks between team A and team B. Look at output for sample input for details.

Sample Input

```
8 5 2
-1 8
-7 3
-2 1
-2 -1
-5 -2
6 -1
2 -4
-4 -5
1 7
1 1
3 4
-6 5
-12 -6
2 -2 10
-2 6 6
-5 -3 15
1 -1 5
1 -3 -3
5 3 -15
0 0 0
```

Output for Sample Input

```
Battle Field 1:
Query 1: 1 -1
Query 2: 1 -1
```



The figure above corresponds to the sample input:

Problemset Analysis:

Problem A – Mobile Casanova

This problem was the easiest problem of the contest. It does not have any trap. It can be done in two ways (a) Treating the phone number as strings (b) Treating the phone numbers as integers. Approach (a) is a bit easier. One judge solution treated the numbers as string and the other judge solution treated the numbers as integers. Because after all a mistake in this problem meant disaster to the contest. All the phone numbers fitted in long integer data type.

Problem B – Potentiometers

The sample solution uses a heap as balanced binary tree to be able to answer queries in O(ln N), at the cost of updates also being O(ln N), as opposed to a brute force solution that answers queries in O(N), but updates in only O(1). Other tree-like structures are possible to implement in this problem, possibly a little slower.

There are also some easier ways of solving this problem like making a group of consecutive 1000 elements. Then each can be answered in O(sqrt(N)). Actually this method is very useful during contest time.

Problem C – Collecting Marbles

This problem is a dynamic programming problem. Each query has to be answered in linear time. The key observation is to realize that the minimum cost row and the minimum cost column has to be found independently. This problem can also be solved based on the fact "For a set of numbers if we find deviation (absolute) with respect to a certain value then the deviation with respect to the median of the numbers will be minimum."

Problem D – Expressions

This is a tricky easy problem. And basically some intelligent sorting is required. In other words it is a greedy problem. The main thing to note is that as the expression contains only + and – operators and the precedence of these operators are same and these operators are associative. So we can actually remove all the brackets just by changing some + to minus and some minus to plus. In the final expression some numbers will be preceded by plus and some by minus. So now we have to make the positive numbers maximum and the negative numbers minimum to maximize the expression. To achieve this goal we sort the given digits. Now to understand the next step we may look at the following example:

+ # #

Suppose we have 987654 digits in hand and we want to put these digits in the above six places. The important thing to realize is that as the sign of both the numbers are positive so we must put the larger digits in the position of higher weights. So a partially filled expression will be:

9 # # + 8 # #, but as we are looking for the lexicographically smallest solution so actual solution will be like:

8 # # + 9 # #.

Following similar logics the final solution will be 8 6 4 + 9 7 5. Following inverse logic we can make the negative signed numbers minimum.

Problem E – The Luncheon

The problem at first sight seems like conventional binomial theorem mathematics. For example consider the math problem below:

"A fair coin is thrown 12 times. What is the probability that it shows head twice the times it shows tail"

Solution: It is obvious that the coin shows heads 8 times and tail 4 times. So the desired probability is:

$$C_8^{12}\left(\frac{1}{2}\right)^8\left(\frac{1}{2}\right)^4$$

Now consider another problem:

"A fair dice is thrown 10 times. Find the probability that 1 is shown twice the time of 2."

Solution:
This problem is different than the previous because other than showing 1 and showing 2 there is a third event of showing $\{3,4,5,6\}$. This third event has probability $\frac{4}{6}$ and the first two events have probability $\frac{1}{6}$. So the solution is: (Which requires multinomial theorem)

$$\frac{10!}{0!0!10!}\left(\frac{1}{6}\right)^0\left(\frac{1}{6}\right)^0\left(\frac{4}{6}\right)^{10}+\frac{10!}{2!1!7!}\left(\frac{1}{6}\right)^2\left(\frac{1}{6}\right)^1\left(\frac{4}{6}\right)^7+\frac{10!}{4!2!4!}\left(\frac{1}{6}\right)^4\left(\frac{1}{6}\right)^2\left(\frac{4}{6}\right)^4+\frac{10!}{6!3!1!}\left(\frac{1}{6}\right)^6\left(\frac{1}{6}\right)^3\left(\frac{4}{6}\right)^1$$

But the problem we are considering is a bit different. All the events in previous two examples were mutually exclusive. In this problem Rahim's list and Karim's list can have common elements. So they are not mutually exclusive. Now what we need to do is to split these two events into mutually exclusive parts and then solve the problem.
We define the following events:

A = Karim eats his favourite dish
B = Rahim eats his favourite dish
C = AB //C is an event consisting the items that are common in A and B
AP = A-C //Eating items that are in Karim's list but not in Rahim's list
BP= B-C // Eating items that are in Rarim's list but not in Kahim's list
Z = S – A-B //S is the total menu event and Z denotes the event that Rahim and Karim eats
//something that is in none of their favorite menu

Now AP, BP, C and Z are mutually exclusive.

Let FAP, FBP and FC denote the number of occurrence of AP, BP and CP respectively. So when A occurs then AP and C both occurs and when B occurs then BP and C both occurs. So when B occurs twice the time of A then the relationship is:

FBP+FC=2(FAP+FC)
Or, FBP+FC=2FAP+2FC
Or, FBP=2FAP+FC.

Based on this relationship We loop over FAP and FC and get the value of FBP from this relationship. Using multinomial theorem we find the values for all possible triple of (FAP, FBP,FC), and we add these probabilities to get the desired probability.

During actual contest the time limit was a bit relaxed and so teams were able to solve in with other slower methods.

Problem F – Counting Zeroes

The problem is an interesting problem of number theory. It is not as easy as it seems but also not difficult. Yes the brute force solution is very easy but that solution will easily time out.

Problem G – Multiplication

This is a problem of combinatorics. To solve this problem one has to separately find the length occupied by x, the length occupied by brackets, the length occupied by powers, the length occupied by operator '+', the length occupied by 'a's and the length occupied by the suffixes of a. Different parts requires four different methods to solve.

Problem H – Paint the Roads

It is obvious that every vertex should have k indegree and k outdegree. So just create a weighted bipartite graph. From source to every vertex in the left side set cap=k and cost=0. From every vertex in the right side to sink set cap = k and cost =0. If in the input there is an edge from i to j of length d then from i'th vertex in the left side to j'th vertex in the right side set cap=1 and cost=d. Then find the minimum cost of n*k flow from source to sink. You can use any standard min cost flow algorithm to solve this one.

Problem I – Protecting Zonk

The problem boils down to finding a minimum cost edge cover in a tree, using special 'guards' at the vertices.

The problem is solved by defining a rooted tree over the graph (the root can be at any vertex), and then recursively calculate the minimum cost of 4 levels of 'protection' for every vertex i:

level 0: the whole subtree rooted at i is covered, possibly with the exception of some of the edges to its direct children;
level 1: the whole subtree rooted at i is covered;
level 2: like level 1, but also the edge to the parent of i is covered;
level 3: like level 2, but also the edge from parent to grand parent is covered.

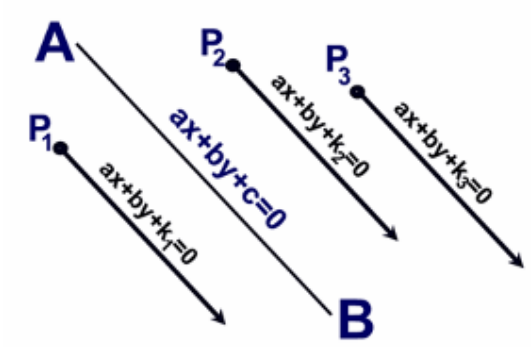
Level 3 protection can only be achieved if there is a sergeant at vertex i, and all children of i have at least level 0 protection.
Level 2 protection can be achieved if there is either a soldier at vertex i and all children have at least level 1 protection, or there is a sergeant at one of its children and all other children have at least level 1 protection.
Level 1 protection can be achieved if there is no guard at vertex i and all children have at least level 2 protection.
Level 0 protection can be achieved if there is no guard at vertex i and all children have at least level 1 protection.

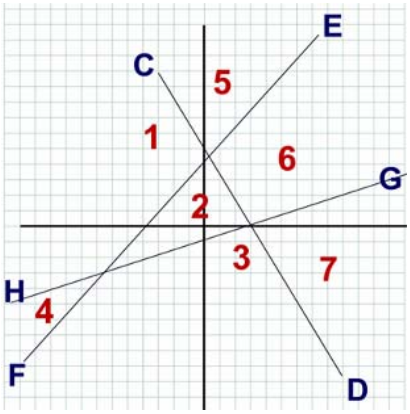
The minimum cost of level 1 protection at the root is the answer to this problem.

The sample solution calculates the costs bottom up, from leaves to root. An other approach is to start at the root and recursively go down to the leaves, memoizing the values of the already visited vertices. Probably the second approach is a little slower.

Problem J – Battle of the Triangle

At first it may seem that this problem for each query the cost is O(N), where N is number of points. But surprisingly after some preprocessing each query can be solved in O(lg (N)). Why? Because for example we have a line AB whose equation is ax+by+c=0 and many other points. Through each of the point we can consider a line parallel to AB whose equation is of the form ax+by+k_i=0, so they are actually different only in the constant term. So with respect to AB we can assign each point P_i a value k_i (the constant term in the equation of the line that passes through P_i) and sort the points based on that value. So now if another straight line is given which is parallel to AB then by doing some binary search we can find how many points are on one side and how many points are on the other side: the cost is O(lg(N)). But how does it help us find how many points are in the triangle and how many are in it's reciprocal region? The answer is - It does not help us find the numbers but it helps us find only the difference!!!





The three lines (No two of them are parallel) divides the plane in seven region, I have marked them from 1 to 7 in the figure on the right. From this moment onwards we denote total points in region 1 as R_1, total points in region 2 as R_2 and so on. Now after the pre processing we find the points that are on the left of CD doing binary search. So now we know $R_1+R_2+R_3+R_4$=AA. Again by doing another binary search we find the total points below GH. So now we know $R_7+R_3+R_4$=BB. Similarly we find total points above EF so we now know: $R_1+R_4+R_5$=BB. And if we find AA-BB-CC then we find $R_1+R_2+R_3+R_4 - R_7-R_3-R_4 - R_1-R_4-R_5 = R_2- R_7-R_4-R_5$, this is actually what we are looking for, the difference ☺. We apply this process twice, once for soldiers and then for tanks. Teams that will look to find R_2 and $R_7+R_4+R_5$ separately will probably fail in their sweet approach.