AdaByron 2016
Programming Contest
http://www.ada-byron.es

Problem set

Supported by

Facultad de Informática (UCM)
26-27 February 2016

In almost every computation a great variety of arrangements for the succession of the processes is possible, and various considerations must influence the selections amongst them for the purposes of a calculating engine. One essential object is to choose that arrangement which shall tend to reduce to a minimum the time necessary for completing the calculation.

Ada Byron

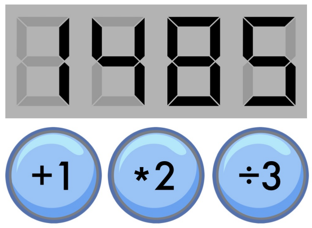
26-27 February 2016

Problem list

Problem setters:

- Marco Antonio Gómez Martín (Universidad Complutense de Madrid)
- Pedro Pablo Gómez Martín (Universidad Complutense de Madrid)
- Enrique Martín Martín (Universidad Complutense de Madrid)
- Isabel Pita Andreu (Universidad Complutense de Madrid)
- Antonio Sánchez Ruiz-Granados (Universidad Complutense de Madrid)
- Clara Segura Díaz (Universidad Complutense de Madrid)
- Alberto Verdejo López (Universidad Complutense de Madrid)

A
The calculator machine



Javier likes electronics and tinkering with machines. His son Luis is now learning to calculate with numbers, and Javier has built for him a machine with a display in which four digits are shown, and there are three buttons marked with the tags `+1`, $*$`2` and $\div$`3`. When the buttons are pressed the value in the display is updated by performing the corresponding operation (to add one, to multiply by two or to divide by three). Since the display has only four digits, the operations are performed modulo 10,000 and division is integer division.

Luis has perfectly understood how the machine works and uses it to verify that the calculations he mentally performs before pressing a button are correct. Now Javier has challenged him to a game: he sets the display to show a specific number and asks Luis to obtain a different number by pressing the buttons as few times as possible.

Can you help them by computing the smallest number of keystrokes to get the final number from the original one?

Input

The program will respond to a series of test cases. Each case consists of a single line with two numbers (from 0 to 9,999), the one which originally appears on the display and the one that Luis has to obtain by pressing the buttons on the calculator machine.

Output

For each test case, the fewest keystrokes required to get the final number from the original one will be written on a single line.

Sample input

```
0 1024
5000 0
9999 6666
```

Sample output

```
11
1
2
```

B

Palm trees in the snow

The mayor of Marbella, a beautiful coastal city of Spain, has received a worrying weather forecast. The 29th of February a great snowfall is expected at the sea level, which is something very unusual. The city hall gardeners have warned him that some palm trees in the promenade could overcome under the weight of the snow, and he is very worried about the deterioration that the promenade could suffer in case many palm trees fall.

The promenade is one of the most popular touristic places in the city where lots of people meet in the beach bars in the hot summer. The hundreds of palm trees that grow there provide for shadow and freshness. The gardening service has made a big effort so do not miss any palm tree in the promenade. When a palm tree falls down there isn't enough time for another one to grow before the summer, so in those places with a small amount of palm trees it will not be possible to install beach bars because of the heat.

The weather service has estimated the weight of the snow falling on the palm trees. And the gardeners have approximated the maximum weight that each palm tree will be able to resist standing. With this information, the mayor wants to know which part of the promenade will be the most affected one; more specifically, the longest strip in which, after the snowfall, a maximum of 5 palm trees will remain standing.

Input

The first line contains the number of proof cases appearing below. Each proof case consists in the following data: in the first line the weight in kilograms of the snow that the weather service has estimated will fall on each palm tree (a natural number); in the following line the number of palm trees in the promenade (bigger or equal than 1 and less or equal than 100,000) and the sequence of weights in kilograms (natural numbers) that each palm tree can resist, from left to right.

Output

For each proof case write in a different line the length of the most affected strip, that is, the longest which contains at most 5 palm trees standing.

Sample input

```
3
30
10
10 30 50 20 40 60 30 40 50 36
40
10
10 30 50 20 40 20 10 10 20 36
20
3
40 10 15
```

Sample output

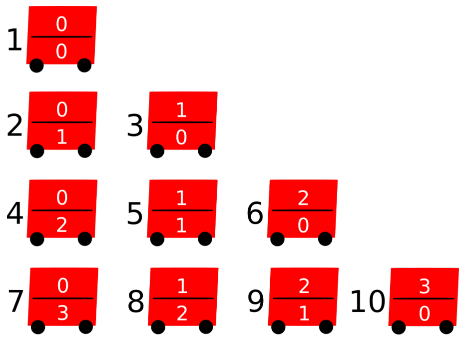
```
7
10
3
```

C

Double decker

John D. Controller works in Tourist-o-Matic, a company that organizes sightseeing double decker buses in Madrid. However, John's job is very boring: he must check all the buses that leave the initial stop and record how many tourists there are on each deck.

In order to make his job more enjoyable, he has created a little game. Instead of recording two numbers for each bus, he only records a single natural number (called *rank*) that perfectly determines the occupation of the bus. To calculate the rank, he distributes the buses by rows depending on the total number of tourists (the rank is the number next to each bus):



John is not very good at maths, so he usually computes a wrong rank. Could you help him with a program?

Input

The first line contains the number A of buses that leave the initial stop. Then there are A lines with the format N M, where N is the number of tourists on the upper deck and M the number of tourists on the lower deck. The maximum occupation of each deck is 1,000.

Output

For each bus, the program will write a line with its rank.

Sample input

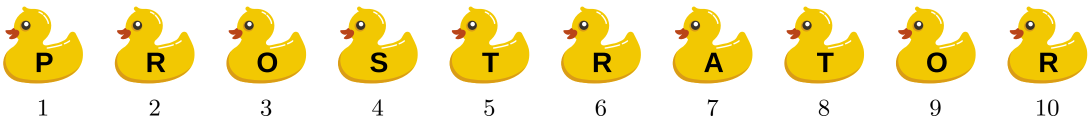
```
3
0 0
1 2
2 0
```

Sample output

```
1
8
6
```

D
Duckindromes

It is a tradition that at the festivities in the village of Juan Filloy[1], he spends one afternoon in the shooting galleries. When the owners know that he will visit them, instead of prizes in the ducks that have to be shooted, they place letters so that Juan can have fun shooting some ducks to form a palindrome with the letters in the surviving ducks (that is, the letters in the remaining ducks read the same both from left to right and from right to left). For example, if Juan faces the series of ducks in the figure below, he would shoot the ducks in positions 1, 4 and 6, thus forming the palindrome `ROTATOR`.



In addition, Juan likes to brag in front of the gals who look at him expectantly, so he is not satisfied with just finding a palindrome but tries to do it by shooting as few ducks as possible. In case of a tie, Juan prefers to shoot ducks on the left, because on the right the sun blinds him.

Would you know how to do the same?

Input

Input will consist of multiple test cases, each on one line. Each case consists of a series of a minimum of 1 and a maximum of 1,000 capital letters of the English alphabet, without special symbols or spaces.

Output

For each test case a line containing the longest palindrome that can be formed by removing (if necessary) some of the letters from the input will be written. In case of a tie, you must imitate John, eliminating letters on the left.

Sample input

```
PROSTRATOR
DELEVELED
ARADAROSOSOMI
OSORASODAR
```

Sample output

```
ROTATOR
DELEVELED
OSOSO
RADAR
```

[1] Argentine writer born in 1894, self-proclaimed world recordman of palindromicity, thanks to which today we know over 8,000 Spanish palindromes.

E

The Count of the Rose

Having reached the end of my poor sinner's life, waiting for my ailing and heavy body to free my soul to face me against the Lord, old and misty memories that I still preserve come to my mind, even though, at the same time, I'm not even able, as a spell of the Antichrist, to remember what I had for breakfast after Matins.

Being just a callow novice, ignorant of the saint and iron discipline in the monastery, I had a confrontation with the abbot, whose name it is only right and pious now to omit, due to what my young discernment considered an affront with the Brother Librarian. He ordered Adso, another novice who would eventually become a wise Franciscan, and me to number the 200 pages of a new edition of the Aristotle's *Poetics* that several brothers had been transcribing some months ago. The librarian, God rest his soul, assigned Adso the numbering of the very first 100 pages of the manuscript, 1 til 100, while I was commanded to number the last 100, from 101 to 200. I had heard disturbing rumours about a curse that killed anyone that became interested in that precise book, and realised that I would be forced to transcribe many more *digits*, but not pages, than my cell-mate, what led me to my confrontation with the abbot.

He, who considered a mere lust for knowledge that I had even considered such a thing in the first place, punished me by praying at Lauds, Terce and Vespers for a complete year the Psalm 30 to beg for protection against injustices. Even so, he must have seen something in my shiny little young eyes, because he conceded to give me a bull from his penalty if I told him what was the last page that Adso should be numbering, and where should I start numbering so that the distribution was fair to both of us. This way, if Adso numbered an extra page, he would be transcribing more digits than me.

I reaped what I sowed because my astuteness vanished and not even today has the Lord granted me the grace to know the answer.

Input

The input consists of an undetermined number of pairs of integers, (p_1, p_2). The first one indicates the starting page that must be numbered and the second one shows the last page. It can be safely assumed that $1 \leq p_1 < p_2 \leq 1{,}000{,}000$.

The input ends with a pair of zeros.

Output

For each testcase, the program will write the number of the last page that should be numbered by Adso, the first novice, so that both of them write the same number of digits. If this turns out to be impossible, Adso will have to number as many pages as possible without exceeding the number of *digits* transcribed by the second novice.

Sample input

```
1 200
99 100
99 101
97 103
0 0
```

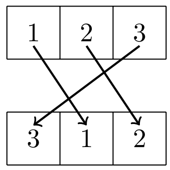
Sample output

```
118
99
99
100
```

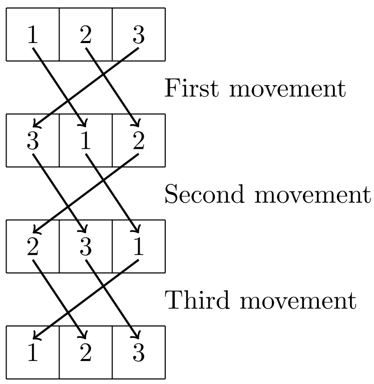
F
Fixed-point theorem

In the family of mathematical theorems called *fixed-point*, Brouwer studied one that, apparently, he explained with the analogy of stirring to dissolve sugar in a cup of coffee. Brouwer stated that at any moment, there is a point/molecule in the cup that didn't change its place.

We won't study that theorem, but we are left with the idea that, when stirring a cup of coffee, molecules exchange their places to end up in a completely different spot of the cup. Let's take a small example (in which the fixed-point theorem is not even fulfilled): molecule 1 could have occupied the spot that molecule 2 had before, molecule 2 gone to where molecule 3 was and, finally, molecule 3 moved to the original place of 1.



If we were capable of replicating the same movement over and over again, a time would come in which the cup of coffee would go back to its original state:



The question we ask ourselves is, given the description of the exchange of molecules we achieve with the motion in the cup of coffee, how many times do we have to repeat the movement so that the state of the cup goes back to being the same as the original?

Input

The input will consist of several test cases. Each of them consists of two lines, the first one containing the number of molecules in the cup of coffee ($1 \leq n \leq 100$) and the other one indicating the movement of molecules that will happen every time that the cup is stirred.

The movement is defined by n integers indicating, for each position of one molecule, in which position it will end up the molecule that was occupying that place.

The input finishes with an empty cup, which should not be processed.

Output

For each test case, write the number of times that the cup of coffee should be stirred so that each molecule goes back to the original place. It is ensured that this number won't exceed 10^9.

Sample input

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

Sample output

```
1
3
2
```

G
Ice-cream cones

Alba and Blanca have conflicting views on ice-cream cones, even if both of them share vanilla and chocolate as their favourite flavours. When their parents have a good day and decide to buy them a two scoop ice-cream, always the same discussion arises between them. When placing the scoops in the cone, the shop assistant pushes the first scoop towards the bottom so he can put the second one top of it. Inevitably, that means that the second flavour to be placed in the cone is the first one to be eaten.

Alba prefers to put chocolate on the top. That's her favourite, so she wants to eat it first because when she reaches the vanilla scoop, her tongue is too cold and she barely perceives the flavour. Blanca also prefers chocolate rather than vanilla, but she has a different theory. It is safer to put the chocolate scoop on the bottom, because the one on the top is more likely to end up on the floor if they eat the ice-cream while walking.

Their parents cannot decide for them because... they buy three scoop ice-creams for themselves, and they also ask for a different order... This way, it is impossible to agree on anything.

Input

The input starts with a line that contains the number of test cases to be processed. Each of the test cases consists on a line with a couple of numbers indicating the number of chocolate and vanilla scoops that will be used for a cone. There will not be an ice-cream without scoops and some huge cones can hold up to 15 scoops.

Output

For each test case, write every single way of placing the scoops. Each configuration of an ice-cream will be written as a sequence of letters `C` and `V` for chocolate and vanilla, respectively. The different configurations will be written in alphabetical order, separated by a whitespace. Do not add the whitespace after the last configuration.

Sample input

```
2
1 1
2 1
```

Sample output

```
CV VC
CCV CVC VCC
```

H

The traveller squirrel

The popular legend says that in the book *Geography*, written in the first century B.C., the Greek geographer Strabo said that vegetation on the Iberian Peninsula was so dense that a squirrel could cross it from the south to the north jumping from tree to tree without ever touching the ground.

Apparently, Strabo never affirmed such a thing in his book and, in fact, nowadays it is believed that the great achievement of the squirrel wasn't even possible back then.

However, let our imagination fly for a while and think that there was in fact a time in which the number of trees in the Peninsula was large enough to make the achievement possible. Considering that today this is not possible anymore, it is clear that at some point in the past a tree was cut down and caused the separation between the north region and the south region, making it impossible for squirrels to jump from branch to branch.

To simplify the problem slightly, let's assume that the territory is a squared region of size $N \times M$ in which trees (considered of thickness 0) are placed in positions (x, y). The task of the squirrel is to go from the tree in the position $(0, 0)$ to the one in (N, M). You can assume that there is always a tree in both of them. The squirrel can jump from one tree to another if the distance between the two of them does not exceed K units.

The information we have about the territory are the positions of all the trees in the beginning of times. We have to determine the position of the tree that, when cut down, stopped the squirrel from travelling from one part to the other without touching the ground.

Input

Each test case consists of several lines. The first of them contains N, M, K and n ($1 \leq N, M \leq 1{,}000$; $1 \leq K \leq 10$; $1 \leq n \leq 100{,}000$), where N, M and K have the meaning described previously and n indicates the number of trees in the territory (without counting the trees in the origin and the destination of the squirrel's journey, which are always present).

After that, there is a line for each of the trees with two integers x, y (the position of the tree). The order in which they are given is the same order followed to cut them down. It is guaranteed that all the positions are inside the territory and that two trees are never placed in the same position.

Output

For each test case, write a single line with the position of the first tree that made the two corners of the field to be unreachable for the squirrel.

If it was never possible for the squirrel to cross the Peninsula from one part to the other, output `Never had the chance`.

Sample input

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

Sample output

```
2 0
Never had the chance
```

The merchant's riddle

A group of pilgrims, together on their way to Santiago, decided to suggest riddles to each other to make the journey more pleasant. Between them there was a merchant, a pensive and organised man that handled numbers with solvency.

When his turn to propose a riddle arrived, he made them see that, in total, they were part of a group of 12 hikers and, hence, they could walk on a single line, or in groups of two, three, four, six or even build a human wall of 12 people. In addition, he explained, if they were less people it would be impossible to make groups in 6 different ways.

"I know that these stony paths — he told them — are narrow in a lot of stages but, leaving that aside, what is the smallest size that our erudite group should have so that we can walk exactly in 64 different ways?"

Only when all of them obtained the Compostelana did the merchant decide to, as a present, reveal the truth.

Input

The program must read, from the standard input, a set of test cases. Each of them will consist of a single number $1 \leq n \leq 1{,}000{,}000{,}000$.

The input will end with a `0`, which should not be processed.

Output

For each test case, the program will write the smallest number of pilgrims that should be part of the group so that they can be structured in exactly n different ways.

Consider that it makes no sense to make groups of more than 1,000,000,000 people, so if the answer exceeds this number, "`+INF`" should be written instead.

Sample input

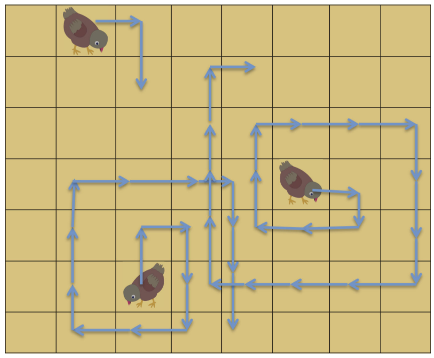
```
6
37
64
0
```

Sample output

```
12
+INF
7560
```

J

Feeding chicks



The chicks spend all day pecking at the floor of the henhouse to eat the grain they find. You've been several days observing their movements and have found that they all follow a curious pattern based on the tiles in the floor. They wake up looking in one direction (north, south, east, west) and start walking in that direction following a clockwise spiral movement. The ride ends when they get tired (each chick has a different endurance) or run into the edge of the henhouse, where they are stunned and sleep until the next day.

For each tile they pass (including the one where they wake up) if there is a grain, they eat it before taking the next step. If there is no grain, they just keep moving forward. Since they are small, when they come together on one point, they eat at the same time without bothering each other and sometimes even sleep in the same place.

To get your chicks grow as quickly as possible without overspend, you decided to distribute the grains in the henhouse so that in every step each chick finds a grain and can eat it. You know the direction in which the chicks wake up, and the number of steps they can take in the spiral before getting tired and stop until the next day. Now you must find the grains that you have to place at each point so that, at the end of the day, none is left and all chicks have eaten as much as possible.

Input

The input begins with the number of test cases to be processed. Each case begins with three numbers, r, c and n meaning, respectively, the size of the chicken coop in the north-south direction, the size in the east-west direction and the number of chicks ($1 \leq r,\ c \leq 50$; $0 \leq n \leq 500$). The next n lines contain the chicks information. The first number, v, means the position in the north-south direction ($1 \leq v \leq r$), the second, h, the position in the east-west direction ($1 \leq h \leq c$), next it is the start direction of the chick (`N`, `S`, `E`, `W`) and finally the maximum number of steps before falling sleep (at least one).

Output

For each test case r lines are written. In each line c values separated by a blank are written, showing the number of grains needed in each point. After each test case write three scripts (`---`).

Sample input

```
2
7 8 3
1 2 E 2
6 3 N 25
4 6 E 21
4 3 2
1 1 N 3
4 2 W 3
```

Sample output

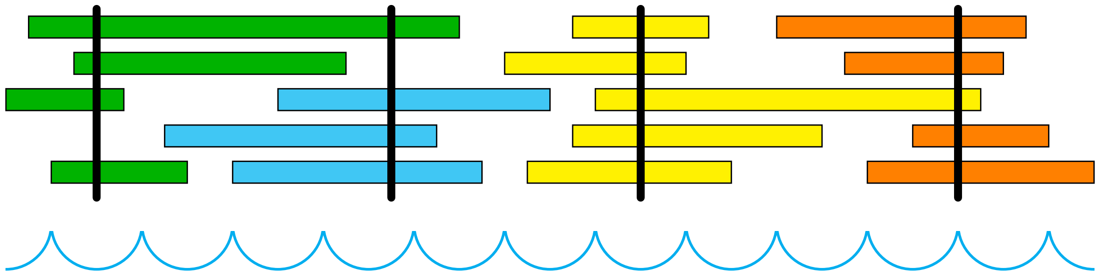
```
0 1 1 0 0 0 0 0
0 0 1 1 1 0 0 0
0 0 0 1 1 1 1 1
0 1 1 2 2 1 1 1
0 1 1 2 2 1 1 1
0 1 1 2 2 1 1 1
0 1 1 1 1 0 0 0
---
1 0 0
0 0 0
1 1 0
1 1 0
---
```

K

On the beach!

When a summer apartment is advertised with a large *On the beach!*, hardly anyone believes that it will be true. Therefore, the owners of several apartment buildings (parallel to the beach but not on the beachfront) have decided to build underground passageways (perpendicular to the beach) connecting all buildings with the sand. Thus, they believe, customers will be happier.

Since building these passageways is not cheap, first they want to know how many tunnels would be necessary at the very least. For example, for the configuration of buildings below (where buildings on the beachfront have been omitted) four tunnels are needed.



Input

Input consists of a series of test cases. Each case begins with a line with the number N of buildings ($1 \leq N \leq 100{,}000$). There follow N lines, each containing two integers representing the westernmost (W_i) and easternmost end (E_i) of each building, with $W_i < E_i$, measured in meters from the westernmost end of the beach. All these measures are integers between 0 and 10^9.

Input will end with a case without buildings, which should not be processed.

Output

For each test case a line will be written with the minimum number of passageways that have to be built. Passageways must be 1 meter wide and to be useful for a building they must be completely beneath it as they pass through.

Sample input

```
4
1 4
6 15
2 10
12 20
2
1 4
4 8
2
1 4
3 8
0
```

Sample output

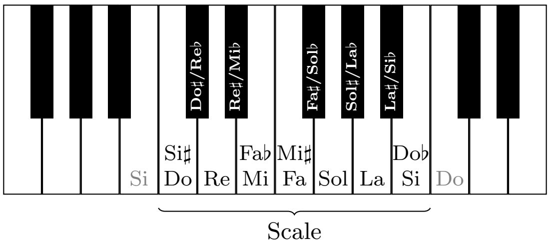
```
2
2
1
```

L
Piano keyboard

Iker is so exited about his new piano! Finally he will be able to play all the songs he likes. The seller has told him that this is a very high quality piano that will last for long, but Iker does not trust him entirely so he has decided to keep track of how many times he presses each key.

The piano keyboard has 7 octaves[2] and each octave has 12 notes (7 white and 5 black keys) each one at half a tone (semitone) of the next one. The white keys correspond to the notes Do, Re, Mi, Fa, Sol, La and Si[3] (and then the Do of the next octave). Do and Re are one tone away so there is a black key between them, but Mi and Fa are only a semitone away. The black keys correspond to those semitones and are named using two special symbols: sharp ($\sharp$) meaning higher in pitch by a semitone, and flat ($\flat$) meaning lower in pitch by a semitone.

This way, the first black key in the octave corresponds to Do $\sharp$ but also to Re $\flat$. Besides, Mi $\sharp$ and Fa are the same key, and Do $\flat$ is the same than Si in the previous octave. What a mess!



Can you help Iker to count how many times he presses each key?

Input

The program will read from the standard input several songs, each one described with 2 lines. The first line shows the number of notes and the second line contains the specific notes. Notes are separated by blanks and they always have the same format: name, accidental (`#`, `b` or nothing), and the octave number. Octave 1 is the lowest pitch and octave 7 the highest.

The input ends with a song with 0 notes that must not be processed.

Output

The program will write a line for each song showing how many times each key was pressed. The keys will be sorted from the lowest to the highest pitch. The first number will correspond to the keystrokes of the lower pitch note in the song and the last number to the highest note in the song, i.e., the solution will never begin or end with zeros.

Sample input

```
6
Do4 Do4 Re4 Do4 Fa4 Mi4
9
Mi5 Re#5 Mi5 Re#5 Mi5 Si4 Re5 Do5 La4
10
Do4 Do#4 Reb4 Re4 Re#4 Mib4 Mi4 Fab4 Mi#4 Fa4
0
```

[2] Actually pianos have some more keys...
[3] Another different notation is C, D, E, F, G, A and B.

Sample output

```
3 0 1 0 1 1
1 0 1 1 0 1 2 3
1 2 1 2 2 2
```