Problem A: TV Reports

Neverland has experienced a very bad economic condition over the past few months. The national currency of Neverland, Oshloob, has lost its value against Dollar very rapidly. People in Neverland, all wondering about the falling down of their currency, are checking the exchange rates of Dollar almost every day.

There are two TV channels accessible to the public in Neverland: National TV (abbreviated as NTV) which is only interested in reporting good economic news, and a foreign media, called BBTV, which is interested in reporting (negative) breaking news. At the end of each day, if the price of Dollar goes down compared to its preceding day, National TV makes a headline, saying "Dollar dropped by *n* Oshloobs", where *n* is the difference between the prices of Dollar in that particular day compared to its preceding day (surprisingly, Dollar never drops by 1 Oshloob in Neverland). Whenever the price of Dollar reaches its all-time highest record (i.e. greater than all prices up to that day), BBTV makes a headline, saying "Dollar reached *m* Oshloobs, A record!", where *m* is the price of Dollar in that particular day.

A company has hired you to record the headlines related to the price of Dollar broadcasted on these two TV channels over a specific period. Since you do not like to watch all TV news to extract the Dollar-related ones, you have decided to obtain the price of Dollar from a reliable source, and produce the headlines yourself.

Input (Standard Input)

The first line of the input contains three positive integers *n*, *p*, and *h* (separated by a space), where *n* ($1 \le n,p,h \le 10{,}000$) specifies the number of days you are hired for to record the headlines, *p* is the price of Dollar one day before you start your job, and *h* ($\ge p$) is the all-time highest price of Dollar just before you start your job. The next *n* lines, each contains an integer less than 10,000; the number given in the i^{th} line specifies the price of Dollar at the i^{th} day of your job.

Output (Standard Output)

For each headline reported by the two TV channels, write a single line on the standard output containing the name of the TV channel (either NTV or BBTV), followed by the headline broadcasted by that TV channel. The output must be sorted based on the reported times of the headlines. The format of the output must conform to the format indicated in the Standard Output below.

Sample Input and Output

Standard Input	Standard Output
8 140 180 120 200 150 150 180 170 250 220	NTV: Dollar dropped by 20 Oshloobs BBTV: Dollar reached 200 Oshloobs, A record! NTV: Dollar dropped by 50 Oshloobs NTV: Dollar dropped by 10 Oshloobs BBTV: Dollar reached 250 Oshloobs, A record! NTV: Dollar dropped by 30 Oshloobs

Problem B: ATM PIN Theft

Gabby the Good Guy is entering his PIN (password) on an ATM while Billy the Bad Buddy is looking over his shoulder to find his PIN. The keys on the ATM keypad are such that when pressed and held, several of that key may actually be entered. The numbered entered are displayed as '*' on the screen, so Billy may not be able to find out the exact PIN. He just knows the keys that Gabby has pressed (in order), but does not know how many times each has been entered into the ATM.

PINs are strings of exactly four digits (0 to 9). ATM has 10 keys for digits, plus one key for Backspace which deletes the last entered digit. Like other keys, if we press and hold Backspace, more than one digit may be deleted. Note that Billy sees in the correct order each key that Gabby has pressed (including the backspace key), but he does not know the number of times that the specified key has actually been entered. Although Gabby's PIN is a string of exactly four digits, he may press more than 4 keys including the keys deleted by Backspace as the ATM accepts any number of keys. He may also press less than 4 keys to produce his 4-digit key.

For example, suppose Billy sees that Gabby has first pressed '1', then '3', then '5', and finally '7'. Since PINs have four digits and no backspace key has been pressed, then there can be only one PIN which is 1357. But in case the pressed keys are '1', '3', and '5' (in that order), the actual PIN may have been 1135, 1335, or 1355.

Input (Standard Input)

There are multiple test cases in the input. The first line of each test case has only a single integer n ($\mathbf{0} < n < 10$), which is the number of keys Gabby has pressed. The next line contains $\boldsymbol{n}$ space-separated integers that are either a single digit (0 to 9), or 99 which denotes the Backspace key. These are the sequence of keys that Gabby has pressed. You are guaranteed that the key 99 appears at most once in each test case but other keys may appear more than once. The input terminates with a line containing "0".

Output (Standard Output)

For each test case, write a single line containing the number of possible PINs consistent with the given sequence of pressed keys. Of course, if there is no PIN consistent with the given sequence of pressed keys, the output must be 0.

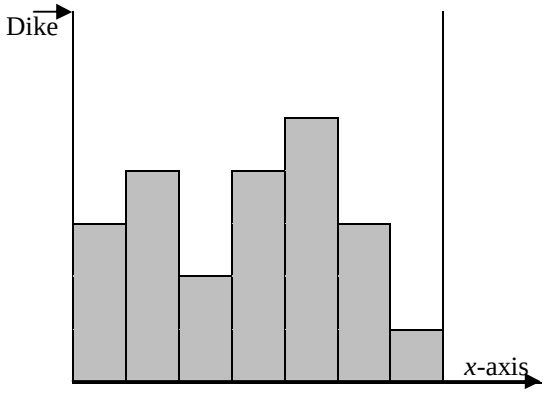
Sample Input and Output

Standard Input	Standard Output
4 1 3 5 7 3 4 6 8 8 1 2 3 4 5 99 3 5 3 2 2 5 0	1 3 7 2

Problem C: Leaking Dike

Petrus was a boy who saved his village by putting his finger inside a leaking dike all night, despite the cold, until the villagers found him and made the necessary repairs. Another time, the dike has started leaking, but unfortunately Petrus is not alive to save the village. The selfish villagers prefer to collect their belongings and run away. As collecting their belongings takes some time, each one wants to compute the time left for his building to go under water completely.

You can assume that the village is a collection of two-dimensional buildings over the *x*-axis. The buildings are attached to each other and all have horizontal roofs with the same width of 1 meter, but the roof heights may be different. The dike, located at the beginning of the village, is at least 1 meter taller than all other buildings and is leaking from its top with the rate of $1m^2$ per minute. There is a wall as high as the dike at the end of the village attached to the last building. You are asked to write a program to help one specific villager to compute the time left for his/her building to go 1 meter under the water.



Input (Standard Input)

There are multiple test cases in the input. Each test case starts with a line containing an integer n ($1 \leq n \leq 10{,}000$) where n is the number of buildings. Assume that the dike is based at *x*-coordinate 0 and start leaking at time 0. The next line contains n space-separated non-negative integers not exceeding 10,000. The i^{th} number is the height of the building constructed at the *x*-interval $[i-1, i]$. Finally, the last line contains the building number for which you have to compute the time left that the building goes **1 meter** under the water. You may assume the buildings are numbered from left to right and the leftmost one is numbered 1. The input terminates with a line containing "0".

Output (Standard Output)

For each test case, output a line containing the time left (in minutes) that the given building goes 1 meter under water.

Sample Input and Output

Standard Input	Standard Output
2 4 5 1 3 3 2 1 2 7 3 4 2 4 5 3 1 5 0	1 3 20

Problem D: Fire Evacuation Plan

In a sunny day, many people of the Orangeland are visiting a flower garden located in an amazing rectangular area in the country. Suddenly, the fire alarm puts all into chaos. Everyone tries to save his/her life by reaching the exit door. Unfortunately, the garden has only one exit door. While all try to reach the exit door as soon as possible, they care about not to trample the flowers (maintained in fire-proof glass boxes) and other people. You are responsible to write a program that finds the best fire evacuation plan for these well-civilized people.

You may assume the garden is modeled as a rectangular grid. In each cell, there exist either some flowers or a spot that one can stand on it. It takes one second for everybody to step from one cell to one of its four edge-adjacent cells. This movement can be done if the adjacent cell is not occupied by others in the next second and of course it is not occupied by flowers. Note that if two persons can simultaneously enter a cell, your fire evacuation plan must admit only one of them to enter the cell; the other one should wait unless he has other options to move. Your program must compute the minimum time needed to evacuate the garden, i.e. everyone reaches the exit door. You are guaranteed that for each person there exists a path from his/her initial cell to the exit door avoiding flowers.

Input (Standard Input)

There are multiple test cases in the input. Each test case starts with a line containing two integers n and m ($3 \le n, m \le 100$) that specify the length of the horizontal and vertical sides of the garden, respectively (each cell is 1×1). The next n lines, each contains m characters from the set {`#, F, P, -,*`}, representing the garden at the fire-alarm time. The boundary of the garden are denoted by "#" characters, except the exit door which is denoted by "`*`". `F` and `P` indicate that the corresponding cell is respectively occupied by flowers or a person at the fire-alarm time. The free cells at the fire-alarm time are represented by "`-`". The input terminates with a line containing "0 0".

Output (Standard Output)

For each test case, display a single line containing the minimum time needed to evacuate the garden.

Sample Input and Output

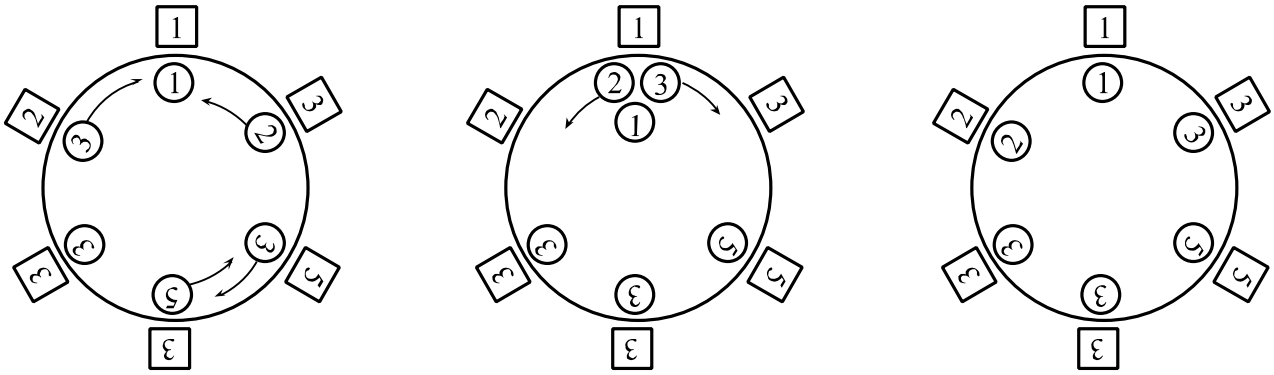
Standard Input	Standard Output
3 3 ### #P# #*# 4 5 ##### #PFP# #P--* ##### 0 0	1 4

Problem E: Forgetful Waiter

Bob is a waiter, working for Pizza Round, a restaurant that offers different types of pizzas. Traditionally, all tables in Pizza Round are round tables with different sizes. Whenever a group of customers enters the restaurant, Bob helps them find a table of proper size and then gets their orders by writing down the number of orders for each type of pizza. Bob is also responsible for serving the pizzas by putting them on the table correctly such that each customer receives the type of pizza which she/he has ordered.

Unfortunately, Bob is forgetful and cannot memorize the type of pizza that each customer has ordered. So, most of the time, he puts the pizzas on the table in a wrong order. However, the customers kindly cooperate to pass along the pizzas to the right customers in several turns. In each turn, each customer can pick up a pizza in front of him with his left hand and pass it to the customer on his left side, or he can pick up a pizza in front of him with his right hand and pass it to the customer on his right side. A customer can use his both hands simultaneously to pass two pizzas to his two adjacent customers in one turn. But there is a space limitation of at most 5 pizzas in front of each customer on the table. So, a customer can only pass a pizza to one of his adjacent customers, when there is enough space in front of that customer.

With these limitations, there are various strategies for a group to pass the pizzas. We are interested in finding the minimum number of turns needed for passing the pizzas. Assume that it is always possible to do that, i.e., the number of pizzas of each type on the table is exactly equal to the number of pizzas ordered for that type by the customers. For example, in the following setting, there is a group of six customers sitting around a round table, there are labels on each customer, which indicates the type of pizza ordered by that customer. The numbers on each pizza indicates the type of the pizza. As you can see in the figure, it is possible to pass the pizzas in just two turns. However, it is not possible to pass the pizzas in one turn, as the single pizza with type 2 needs at least two turns to reach its destination.



(a) (b) (c)

Write a program to find the minimum number of turns needed to pass all pizzas on the table to their correct customers. The exact strategy of passing the pizzas is not important to us. We are only interested in the minimum number of turns needed.

Input (Standard Input)

There are multiple test cases in the input. The first line of each test case contains an integer n, the number of customers in the group ($1 < n \leq 1000$). Suppose that the customers in the group are numbered from 1 to n and sit around a table of size n consecutively in a counter-clockwise order, so that customer 1 is adjacent to customer n. The next n lines (numbered from 1 to n) describe the customers' orders and the current arrangement of the pizzas on the table. In each line i, there are two integer numbers a_i and b_i in range $[1, n]$, separated by a space, where a_i indicates the type of pizza ordered by customer i, and b_i indicates the type of pizza actually placed by Bob in front of that customer. The sequence of b_i's is actually a permutation of a_i's sequence. The input terminates with a line containing "0".

Output (Standard Output)

In the output file, for each test case, just write a non-negative number *m* in a separate line, which is the minimum number of turns needed to transfer all pizzas, for that test case. If in a test case the pizzas are already placed correctly, you must write number 0 in the corresponding line of the output.

Sample Input and Output

Standard Input	Standard Output
6 1 1 3 2 5 3 3 5 3 3 2 3 7 7 2 4 7 1 4 3 1 5 3 2 5 2 2 0	2 1

Problem F: Computerizing a Stockroom

Jobs is a stock clerk who has been working in a computer manufacturing company since its establishment time. He is responsible for any transaction done in the stockroom where computer parts are stored. Jobs is an old-fashioned employee who yet insists to register any transaction in his booklet and resists to make transaction registration computerized. He did not believe in modern technologies till today when his boss asked him to provide a summary report on the current state of the stockroom, including the computers given out to employees, and working and non-working computer parts remaining in the stockroom. He is so nervous as the report is due tonight. He does not have enough time to manually prepare a report from many transactions in his booklet. Now, he believes in modern technologies and asks you to write a program which gets all transactions as input, and outputs the report soon.

Each transaction written on his booklet starts with a date-time, and is followed by one of the following templates:

- Bought <*NUM*> <*PIECE*>.
- Assembled a computer for <*PERSON*> using <*PIECES*>.
- Got the computer of <*PERSON*> back and disassembled it.
- Found that <*A*> <*PIECE*> is not working.
- <*A*> <*PIECE*> is repaired and now can be used again.

The placeholders used above are defined as below (the first letter is capitalized in the case of appearing at the beginning of a sentence):

- <*A*>: it is either "a" or "an" depending on its successive word.
- <*NUM*>: it is either <*A*> (which means 1), or "*X* items of" where *X* is an integer greater than 1.
- <*PERSON*>: it represents the full name of an employee, containing a number of space-separated words all starting with capital letters.
- <*PIECES*>: it is a list of "<*NUM*> <*PIECE*>", separated with commas. There are always at least 2 entries in the list, and an extra "and" will follow the last comma (before the last entry).
- <*PIECE*>: it is a phrase including a computer-part name (like RAM or CPU) and possibly some extra data which may represent the model, speed, capacity, etc.

You can assume that each entity (computer-part type or employee name) is always referenced with a unique case-sensitive phrase, and that no two different entities are referenced with the same phrase, even case-insensitively. All computer parts that are bought are initially working; only working computer parts are used for assembling a computer. Each employee has at most one computer at any time and all transactions are logically valid at the time of writing.

Input (Standard Input)

There are multiple test cases in the input. Each test case starts with a line containing the integer n ($1 \le n \le 500$), the number of transactions. A transaction is given on each of the next n lines. The date-time format is "*year*`-`*month*`-`*day* *hour*" where parameters *year*, *month*, *day*, and *hour* are in the ranges $[2000, 2012]$, $[1, 12]$, $[1, 31]$, and $[0, 23]$ respectively and may be padded with a "0" in the case of being less than 10. The date-times are unique within each test case. The date-time is separated from the transaction sentence with the string " `-` ". All numbers used in <*NUM*> are less than 10^5. For your convenience, each two consecutive words in the input are separated with a single space, and employee names and computer-part types are wrapped with quotation marks and all characters appearing in the input are alphanumeric. Refer to the sample input for the details. The input terminates with a line containing a single "`0`" which should not be processed as a test case.

Output (Standard Output)

For each test case, write several lines of information. On the first line, write *X*, the number of employees who currently have a computer in the following format:

- If $X > 1$, write "`There are X employees who currently have a computer:`".
- If $X = 1$, write "`There is one employee who currently has a computer:`".
- If $X = 0$, write "`No computer is currently given out to the employees.`".

Then, print *X* lines, each with the full name of one of those employees. These *X* lines must be sorted in the lexicographic order.

On the next lines, print some sentences about the current state of all computer-part types written in the booklet. There must be exactly one line for each type. The lines must be sorted in the lexicographic order based on the piece phrases. For each piece phrase <*PIECE*>, its corresponding line must be in one of the following forms:

- If there is no item of the part type currently remaining in the stockroom, then write
  "`There is no "<PIECE>" left in the stockroom.`".
- If there is only one item of the part type remaining in the stockroom, and it is working, then write
  "`There is one "<PIECE>" left in the stockroom which is working.`".
- If there is only one item of the part type remaining in the stockroom, and it is not working, then write
  "`There is one "<PIECE>" left in the stockroom which is not working.`".
- If there are X ($X > 1$) items of the part type left in the stockroom, and all of them are working, then write
  "`There are X items of "<PIECE>" left in the stockroom, all working.`".
- If there are X ($X > 1$) items of the part type left in the stockroom, and all of them are not working, then write
  "`There are X items of "<PIECE>" left in the stockroom, all not working.`".
- If there are X ($X > 1$) items of the part type left in the stockroom, and Y number of them are working and Z number of them are not working ($Y, Z > 0$), then write
  "`There are X items of "<PIECE>" left in the stockroom, Y working and Z not working.`".

Print a line containing "###" between every two consecutive test cases. Note that when you compare two multi-word phrases, you first compare the first words of the two phrases. If they are equal, you must continue with the second words of the two phrases and so on in the case of equality of the second words. If all words of the shorter phrase match the corresponding words of the longer phrase, the shorter phrase comes first. In lexicographic comparison of the words, digits have a higher priority than letters, and the letters are compared case-insensitively. So, "A AB" < "A10" < "A2" < "aa" = "AA" < "AA B".

Sample Input and Output

Standard Input

```
6
2011-3-18 12 - Bought 3 items of "CPU Pentium IV 2GHz".
2011-11-1 10 - Found that a "1GB RAM" is not working.
2012-1-20 11 - Bought an "optical mouse".
2011-11-21 15 - A "CPU Pentium IV 2GHz" is repaired and now can be used again.
2011-03-18 9 - Bought 2 items of "1GB RAM".
2011-10-18 14 - Found that a "CPU Pentium IV 2GHz" is not working.
16
2012-2-1 08 - Bought 3 items of "motherboard".
2012-2-1 09 - Bought 3 items of "Green case".
2012-2-1 10 - Bought 3 items of "dual core CPU".
2012-2-1 11 - Bought 3 items of "keyboard".
2012-2-1 12 - Bought 2 items of "optical mouse".
2012-2-1 13 - Bought 4 items of "2GB DDR3 RAM".
2012-2-1 14 - Bought 4 items of "500GB Hard".
2012-2-1 15 - Bought a "DVD Drive".
2012-2-2 09 - Assembled a computer for "Abbaas" using a "motherboard", and a "Green case".
2012-2-2 10 - Assembled a computer for "Dehghaan Fadaakaar" using a "motherboard", a "Green
    case", a "dual core CPU", a "keyboard", an "optical mouse", 2 items of "2GB DDR3 RAM", a
    "DVD Drive", and a "500GB Hard".
2012-2-2 11 - Assembled a computer for "Kokab Khaanum" using a "motherboard", a "Green
    case", a "dual core CPU", a "keyboard", an "optical mouse", 2 items of "2GB DDR3 RAM",
    and a "500GB Hard".
2012-2-3 10 - Got the computer of "Kokab Khaanum" back and disassembled it.
2012-2-4 10 - Found that a "motherboard" is not working.
2012-2-4 11 - Found that a "2GB DDR3 RAM" is not working.
2012-2-4 12 - Found that a "2GB DDR3 RAM" is not working.
2012-2-4 13 - Found that a "500GB Hard" is not working.
0
```

Standard Output

```
No computer is currently given out to the employees.
There are 2 items of "1GB RAM" left in the stockroom, 1 working and 1 not working.
There are 3 items of "CPU Pentium IV 2GHz" left in the stockroom, all working.
There is one "optical mouse" left in the stockroom which is working.
###
There are 2 employees who currently have a computer:
Abbaas
Dehghaan Fadaakaar
There are 2 items of "2GB DDR3 RAM" left in the stockroom, all not working.
There are 3 items of "500GB Hard" left in the stockroom, 2 working and 1 not working.
There are 2 items of "dual core CPU" left in the stockroom, all working.
There is no "DVD Drive" left in the stockroom.
There is one "Green case" left in the stockroom which is working.
There are 2 items of "keyboard" left in the stockroom, all working.
There is one "motherboard" left in the stockroom which is not working.
There is one "optical mouse" left in the stockroom which is working.
```

Problem G: Harvesting a Farm

Mr. Farmer owns a rectangular farm in Newfoundland. He has partitioned his farm into $n \times m$ equal-size squares. In each square, he has planted either of the two grains: wheat or corn. By many years of experience, Mr. Farmer has found a simple rule for improving the quality of the grains: if "1" represents wheat, and "2" represents corn, then there must be no 2×2 adjacent squares in the farm forming one of the following "crossing" patterns:

1	2
2	1

2	1
1	2

Mr. Farmer has a combine for harvesting the grains. To harvest each type of grain, the farmer needs to attach a special cutter to his combine. At the beginning, when no cutter is attached to the combine, and also, during the course of replacing the combine's cutter, the farmer can move the combine to any place of the farm, without harvesting the grains. However, once a cutter is attached, the combine can only move either on the squares containing grains compatible with the combine's cutter, or on the squares with no grain (namely, those harvested before).

Since replacing the cutter is a tedious task, the farmer wishes to hire you to show him the best way for harvesting the whole farm using a minimum number of cutter replacements. Your task is to write a program to help the farmer find such a minimum number.

Input (Standard Input)

There are multiple test cases in the input. Each test case starts with a line containing two integers n and m ($1 \leq n \times m \leq 10^5$) that specify the number of rows and columns in the farm, respectively. The next n lines, each contains m characters from the set {1, 2}, representing the type of grain in the corresponding square of the farm. The input terminates with a line containing "0 0".

Output (Standard Output)

For each test case, output a single line containing the minimum number of times that the farmer needs to replace the combine's cutter. The first time that a cutter is attached to the combine is also counted as a replacement.

Sample Input and Output

Standard Input	Standard Output
2 3 112 211 4 4 1212 2211 1112 1222 0 0	2 3

Problem H: Rectangular Painting

A rectangular painting consists of several rectangles that meet the following conditions:

1) Each rectangle is either completely within another rectangle, or it does not overlap with other rectangles.
2) The two sides of each rectangle are parallel to the *x* and *y* axes.
3) The sides of any two rectangles are at least *d* units apart even if one of them contains the other one.
4) For each rectangle, the smallest rectangle containing it is called its super-rectangle. Rectangles with the same super-rectangles are called co-rectangles. All co-rectangles are aligned either vertically or horizontally. Two co-rectangles are horizontally (vertically) aligned if their bottom (left) edges are co-linear.
5) Each rectangle with no inner rectangle is called a photo-rectangle and is filled with a photo of its size.
6) Each rectangular painting has exactly one rectangle with no super-rectangle. This rectangle is called the **root** rectangle.
7) The coordinates of the rectangle corners are all integer numbers.

Given the structure of a rectangular painting (see input), we want to find the minimum possible area of the root rectangle by selecting the directions of the alignments of all co-rectangles. Note that the dimensions and orientations of the photo-rectangles are given and cannot be changed.

Input (Standard Input)

There are multiple test cases in the input. Each test case starts with a line containing $1 \le n \le 100$ and $0 \le d \le 30$, where n is the number of rectangles in the rectangular painting. In the next n lines, the i^{th} line is the description of the i^{th} rectangle (rectangle with id i). The **root**-rectangle id is 1. Let R_i be the id set of rectangles whose super-rectangle is the rectangle with id i. If R_i is not empty, the description of the i^{th} rectangle is the size of R_i followed by the members of R_i all separated with spaces. Otherwise, the rectangle is a photo-rectangle and its description is of form "0 a b" where $1 \le a \le 30$ and $1 \le b \le 30$ are the sizes of its *x*-axis and *y*-axis sides, respectively. The input terminates with a line containing "`0 0`"

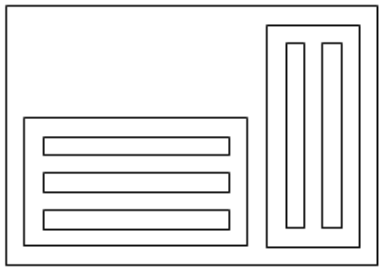
Output (Standard Output)

For each test case, write a single line containing the minimum area of the **root** rectangle among all possible conformations of the given rectangular painting.

Sample Input and Output

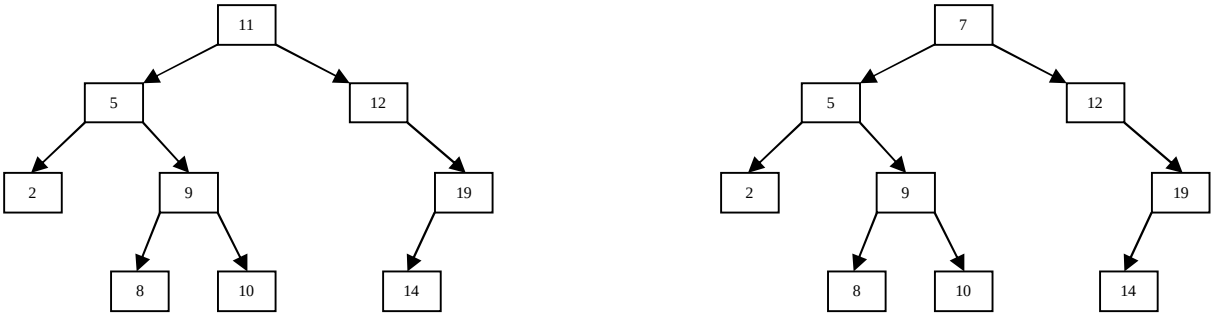
Standard Input	Standard Output
8 1 2 2 3 3 4 5 6 2 7 8 0 10 1 0 10 1 0 10 1 0 1 10 0 1 10 0 0	280

Sample output configuration:



Problem I: Corrupted BST

Memories in computers are not always fully reliable, and sometimes the content of a memory word can be corrupted. This may be caused by manufacturing defects, power failures, or environmental conditions such as noise and high temperature. Obviously, a memory fault may seriously affect the computation. For instance, assume you maintain a Binary Search Tree (BST) over real numbers in order to perform search operations faster. A BST indeed is a binary tree data structure which has the following properties: for any node with a key x (i) its left sub-tree contains only nodes with key less than x, (ii) its right sub-tree contains only nodes with key greater than x, (iii) both left and right sub-trees are BST and (iv) duplicate keys are not permitted. A BST of size 9 is depicted below (left one). If 11 is changed to 7 in the BST due to a memory fault (right figure), the BST is invalid, i.e. some BST properties do not hold anymore. In the invalid BST depicted below (right figure) searching 9 follows a wrong path and reports a wrong answer.



To maintain a valid BST, we can verify the BST regularly and transfer it into a valid BST (of course if it is invalid). One easy way to make the BST valid is to select some nodes (not necessarily those whose keys are changed due to memory faults) and change their keys to appropriate keys that make the BST valid. As we are willing to change few keys in the BST, you are to compute the minimum number of nodes whose key can be changed to make the BST valid.

Input (Standard Input)

There are multiple test cases in the input. For each test case, a BST is given in the following format. The first line contains a single number $n \leq 50{,}000$ which is the size of the BST. In each of next $n - 1$ lines you can find three items separated by spaces which together specify a node. The first item is the key stored at the node, the second one is the key stored at the parent of the node and finally the last one which is either "L" or "R" specifies whether the node is the left ("L") or the right ("R") child of its parent. You may assume all keys are non-negative integers (at most 10^6) and distinct but note that the keys stored in valid BST may be real numbers. The input terminates with a line containing "`0`".

Output (Standard Output)

For each test case, write a single line containing the minimum number of changes that must be applied to the BST to make it valid.

Sample Input and Output

Standard Input	Standard Output
9 5 7 L 2 5 L 9 5 R 12 7 R 19 12 R 8 9 L 14 19 L 10 9 R 0	1

Problem J: Sightseeing

Mark and his sister plan to visit some landmarks in Zibacity. To plan their sightseeing, they have bought a map which shows the main landmarks (including their geographic coordinates) in the city as well as the existing streets joining the landmarks. The streets are either for walking or for biking. As Mark loves biking, he has decided to tour the city by a bike. Conversely, his sister who dislikes biking has decided to walk. They agree to start their trip from the central train station and end at the west train station which both are amazing landmarks and exist in the map. As the landmarks are so beautiful, they have decided to spend **at least** one night in each of their visiting landmarks on their paths (not necessarily all in the city). Fortunately, going from a landmark to another can be done in a few hours in a day by biking or walking. Since camping in streets is not permitted, they have decided to camp close to the landmark they visit each night and talk to each other every night before falling sleep. To make such calls, they have decided to buy a pair of two-way radio which is used for radio calls. Since any two-way radio has a limited coverage radio range and its price depends on its coverage radio range, they plan to find their paths (biking path and waking path) such that the needed coverage radio range is minimized. As computing this value is not easy by hand, they have submitted their problem to ACM programming contest in Tehran site to get their answer efficiently.

Input (Standard Input)

There are multiple test cases in the input. Each test case is a graph that models the map: vertices and edges show landmarks and streets, respectively. The first line of each test case contains two non-negative integer numbers n and m where $n \leq 50$ is the number of vertices and m is the number of edges. In the next n lines, the i^{th} line contains the x and y coordinates of the vertex with id i. Then, the output is followed by m lines describing edges. Each line contains two vertex ids (defining the edge) and one character namely "W" or "B". "W" means the edge is a walking street and "B" means the edge is a biking street. Note that between two landmarks may exist both walking and biking streets. Finally, in the last line, the ids of the central and west stations are given respectively. You may assume there are walking and biking paths between the central and west stations and all number are non-negative integer less than 10^4. The input terminates with "0 0".

Output (Standard Output)

For each test case, you must print the **square** of the minimum coverage radio range required in a single line.

Sample Input and Output

Standard Input	Standard Output
4 4 0 0 0 1 1 0 1 1 1 2 W 2 4 W 1 3 B 3 4 B 1 4 0 0	1