Problem A

Minimum Swap

Time Limit: 1s

Sorting a sequence either into ascending or descending order is a common problem in real world.

One of the most common methods in sorting algorithm is comparing and swapping any two elements in the sequence. Supposed I have invented a new sorting algorithm which uses this swapping method. I need to compare the number of swaps occurred in my algorithm with the minimum number of swaps that actually needed to sort some sequences.

Help me determine the minimum number of swaps needed to transform a sequence of characters into a sorted sequence in ascending order. In this problem, you may assume that each character will appear at most once in the sequence.

Input
Each line contains a sequence of lowercase characters S ($1 \leq | S | \leq 26$). There will be no repeated characters in S and each i^{th} character ($0 \leq i < | S |$) is in range [‘a’…’z’]. Input is terminated by EOF.

Output
For each test case, output a line containing the minimum number of swaps to transform it into a sorted sequence of characters in ascending order.

Sample Input	Output for Sample Input
abc cba acb bdca fedcba	0 1 1 2 3

Problem B

GCD!

Time Limit: 1s

The greatest common divisor of two numbers is defined as the largest integer that divides both numbers without leaving any reminder. For example, the greatest common divisor of 8 and 12, written as GCD(8,12) is 4, as 4 is the largest integer that divides both 8 and 12 (the common divisors of 8 and 12 are 1, 2, and 4).

Meanwhile, the factorial of a natural number is the product of all positive integers less than or equal to that number. For example, the factorial of 5, written as 5! is 1*2*3*4*5, which equals to 120. By convention, 0! is 1.

Given two integers, *n* and *k*, you should find the greatest common divisor of *n*! and *k*. For example, if n = 3 and k = 10, then GCD(*n*!,k) = GCD(3!,10) = GCD(1*2*3,10) = GCD(6,10) = 2. Write a program to find this number!

Input
Each line contains two integers, n ($0 \le n \le 1{,}000{,}000{,}000$) and k ($1 \le k \le 1{,}000{,}000{,}000$) respectively.

Output
For each line of input, output one line containing the GCD of *n*! and *k*.

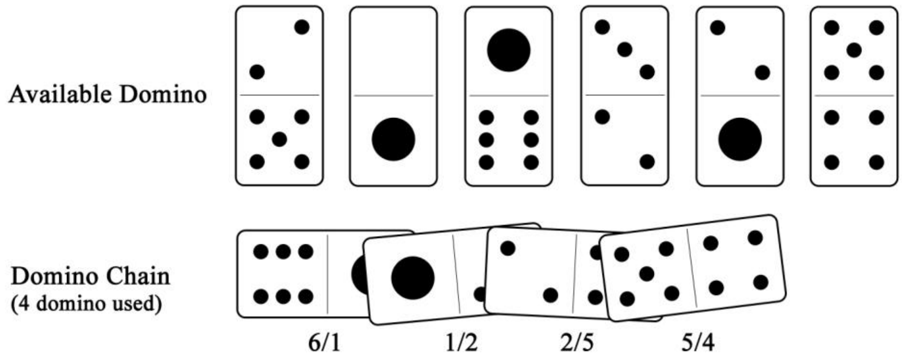
Sample Input	Output for Sample Input
3 10 10 240 12 364 100 2351 629 163547	2 240 28 1 67

Problem C

Playing With Domino

Time Limit: 5s

A domino is a 2x1 rectangular tile with a line dividing its face into two squares, each with a certain number of dots from zero spots to six spots. As you might know, there are many different games which can be played using dominoes. In most games, dominoes are played by arranging them into a single chain where adjacent dominoes must have matching numbers (see example illustration below).



There are 28 possible faces in a complete set of domino: 0-0, 0-1, 0-2, 0-3, 0-4, 0-5, 0-6, 1-1, 1-2, 1-3, 1-4, 1-5, 1-6, 2-2, 2-3, 2-4, 2-5, 2-6, 3-3, 3-4, 3-5, 3-6, 4-4, 4-5, 4-6, 5-5, 5-6, 6-6. Of course you can play each domino in either direction. For example, you can play 1-6 as 1/6 or 6/1 depending on your need.

Write a program to find the longest chain that can be made from a given set of dominoes. There may be more than one domino with the same face.

Input

Input contains several cases. Each case begins with an integer N ($1 \leq N \leq 1{,}000$) denoting the number of available domino. Each of the following N lines contains a pair of number (0 to 6) which represents the domino. The first number in each pair is always smaller or equal to the second number.

Output

For each case, print in a single line the number of domino used in the longest domino chain.

Sample Input	Output for Sample Input
6 2 5 0 1 1 6 2 3 1 2 4 5 2 0 1 3 5	4 1

Problem D

Email from the Professor

Time Limit: 1s

Being a judge in a programming contest like ACM/ICPC INC sometimes means trouble. They have to prepare challenging and interesting problems, and most importantly, they should keep the problems secret before the contest day. Usually, the judges use email as a main device to communicate and discuss on problems. As you might think, this is not a secure way to communicate confidentially since emails can be unintentionally sent to unexpected recipients, e.g. his/her student or worst, the contestant.

Considering those issues, Prof. Nash V. Ruhdney, one of the judges, has proposed a method to communicate confidentially amongst judges. Supposed there is a message of length *n*, write the message in a rectangle of width *k*, row by row, and read it column by column in a permuted order. The order of columns will then be the encryption key for the message. For example,

```
Message      : I am Prof. Nash V. Ruhdney
Key          : 3 7 4 1 2 6 5
Plain Text   : I   a m   P r
               o f .   N a s
               h   V .   R u
               h d n e y * *      note: '*' means blank and is not part of the message.
Column       : 1111222233334444555666677777
Cipher Text  : m .e N yIohha.VnrsuPaR f d
```

Prof. Nash then send this encrypted message (cipher text) through email, while the encryption key is sent through other means, such as phone or SMS. This way he can reduce the risk of erroneously sent email.

Of course there is no way that the professor would use paper & pencil to encrypt those messages. So, would you be kind to help him while he is preparing a nice problem for you?

Input

Input consists of several test cases. The first line of each test case contains a message of length *n* ($1 \le n \le 1{,}000$) in a line (the message consists of ASCII characters). The next line contains *k*, ($1 \le k \le 1{,}000$) the width of rectangle, followed by *k* integer(s) separated by a single space, the permutation of 1 to *k* which represents the encryption key.

Output

For each message, output the encrypted message in a single line.

Sample Input	Output for Sample Input

`I am Prof. Nash V. Ruhdney`
`7 3 7 4 1 2 6 5`
`give no easy problem this year.`
`5 1 4 3 2 5`
	`m .e N yIohha.VnrsuPaR f d`
	`gnso  .eepeiav  lheioybty armsr`

Problem E

The Adventure in Panda Land Part I: Panda Number

Time Limit: 1s

In Panda Land, pandas have their own numeral system to represent a number. Surprisingly, this numeral system is similar to the famous Roman Numerals in our ancient civilization. Although pandas are aware about numeral system, they can not write (imagine how can a panda write!). Instead, they cut and arrange bamboos to form a number.

These are the production rules of Panda Number:

- Letters which represent powers-of-ten (I, X, L, C, M, W, Y, H, A) can be repeated at most three times. The others (V, L, N, E, F, K, T) can not be repeated.
- If one or more letters are placed after another letter of greater value, add that amount. The letters should be written in descending order from the largest to the least value. For example:
  FXVIII = 50000 + 10 + 5 + 1 + 1 + 1 = 50018
- If a letter is placed before another letter of greater value, subtract that amount. Only subtract one letter from another. For example:
  XIV = 10 + (5 - 1) = 14
  CIX = 100 + (10 - 1) = 109
  MECIX = (5000 - 1000) + 100 + (10 - 1) = 4109
- Only powers-of-ten can be used to subtract a value (that is, you can subtract I from V, or X from L, but not V from X (V is not powers-of-ten.)
- Do not subtract a letter from one that is more than 10 times greater (that is, you can subtract I from X, but not I from L – there is no such number as IL.)

Decimal	Panda Number	Required Bamboo
1	I	1
5	V	2
10	X	2
50	L	2
100	C	3
500	N	3
1,000	M	4
5,000	E	4
10,000	W	4
50,000	F	3
100,000	Y	3
500,000	K	3
1,000,000	H	3
5,000,000	T	2
10,000,000	A	3

The aforementioned rules imply that there is exactly one representation in Panda Number for each number in decimal system.

Unlike those Roman Numerals, pandas do recognize zero and negative number (which prove pandas are more advanced than our ancient Romans). To represent a negative number, they add one bamboo as a negative sign in front of the letters. Zero is a special number which doesn't fall into any rules above. To form a zero, pandas need five bamboos.

The number of bamboos needed to form a number is the sum of required bamboos for each letter that appears in that number. For Example:

```
4108 = 4000+100+8 = (5000-1000)+100+5+1+1+1 → MECVIII (16 bamboos).
4109 = 4000+100+9 = (5000-1000)+100+(10-1) → MECIX (14 bamboos).
-205 = [-], 200+5 = 100+100+ 5 → -CCV (9 bamboos)
```

Given two numbers A and B, find out how many bamboos needed by panda to form (remember, they can't write) all number between A and B inclusively.

Input

The input begins with a single positive integer *T* in a line indicating the number of test cases. Each case contains two numbers A and B (-25,000,000 ≤ A ≤ B ≤ 25,000,000) in a line.

Output

For each case, print in a single line the number of needed bamboos to form all numbers between A and B (inclusive).

Sample Input	Output for Sample Input
7 -1 1 4018 4019 -25000000 25000000 -100 -57 -100 0 0 100 43 100	8 28 2121000021 399 778 678 434

Problem F

Jakarta Traffic Jam

Time Limit: 1s

The traffic in Jakarta is terrible. Streets in Jakarta may get congested with traffic jam during rush hour. This means it will take longer time to pass through those streets since you can only drive half your speed during rush hours. Therefore, you need to plan your trip carefully if you don't want to be late for your activities.

For example, consider a street from P to Q which will need 20 minutes to drive in a normal time. The street is congested with traffic at 15:00 until 16:00. Now, let's consider some sample conditions that might happen related to rush hours:

a. If you start your trip 15 minutes before the rush hour (start from P at 14:45), then when the traffic starts to get congested, you can only drive half of your normal speed, thus, you will need 10 (2x5) more minutes to reach Q. Thus, the total time you need to go from P to Q is 15 + 10 = 25 minutes.
b. If you start your journey at the last 5 minutes of the rush hour (start from P at 15:55), the distance you have reached in the first 5 minutes is equal to the distance you can reach in 2.5 minutes during normal hours. You will need 17.5 more minutes to get to Q. Thus, the total time you need to go from P to Q is 5 + 17.5 = 22.5 minutes.
c. Other combinations of (a) and (b).

Given a map which has at most *N* intersections and *M* streets, find out how many minute(s) you need to go from *s* to *d*. You want to reach *d* as soon as possible without wasting any seconds or microseconds, or even nanoseconds.

Input

The input contains several test cases. Each case begins with two integers *N* ($1 \le N \le 20$) and *M*. Each intersection will be numbered from 0 to *N*-1.

The first three integers *P*, *Q*, and *T* ($1 \le T \le 50$) in each of the following *M* line means there is a street connecting intersection *P* to intersection *Q* which need *T* minutes driving in normal time. These integers will be followed by either character '`N`' or '`R`'. '`N`' will not be followed by any characters and it means the street has no rush hours. '`R`' will be followed by the starting and ending time of the rush hour in the format of *hh:mm* (00:00 to 23:59). No rush hours will rollover past midnight.

The last line of each case contains *s* - your starting point, *d* - your destination and *w* - current time.

Input is terminated by a line containing two zeroes. This line should not be processed.

Output

For each case, output a line containing the minimum time you need to go from *s* to *d* in minutes. Your output should always contain two digits after the decimal point.

Sample Input	Output for Sample Input
2 1 0 1 20 R 15:00 16:00 0 1 14:45 3 3 0 1 20 R 15:00 16:00 1 3 10 N 2 1 35 R 16:30 17:00 0 2 15:55 0 0	25.00 82.50

Problem G

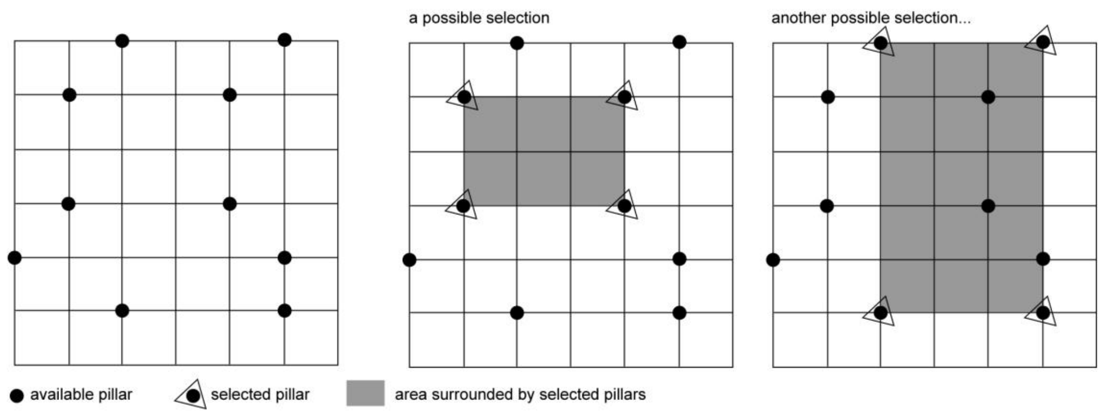
Sultan's Land

Time Limit: 1s

Sultan Al-Bandar, the ruler of Old West Sumatra Sultanate, has decided to give his land to his only son. Although the Sultan believes that his son will not misuse the land for some bad purposes, he still has some doubts on his son's capability to rule the land. Therefore, he decided to give his son a test, a puzzle test.

The Sultan drew *N* x *N* grids (virtually of course) on his land where any two adjacent grid intersections would have the same length. Then he placed some pillars on the land where each pillar was placed at one intersection. Then he summoned his son to come to the land.

"Well, my dear son, if you are to choose four pillars to form an area where each pillar serves as a corner of that area, how many possible selections are there?" asked the Sultan. His son replied him with a big smile. Just at the time the Sultan remembered that his son works for the Sultanate's Advance Combinatorial Ministry, so this problem might be too easy. Then he quickly added, "Oh! Each side of the area should be parallel to the grid I have drawn before."



To tell the truth, the Sultan himself actually had not prepared any answers since he spontaneously asked his after the big smile. Help this Sultan by writing a program to count how many possible selections can be made by applying the rules above.

Input

The input contains several test cases. Each case begins with two integers, N ($2 \le N \le 100$) the grid-size, and *P* ($2 \le P \le N^2$) the number of pillars. Each grid will be numbered from 1 to *N*. The next *P* following lines each will contains two integers: *r* and *c* ($1 \le$ r, c $\le N$) the row and column where the pillar is placed.

Input is terminated by a line containing two zeroes. This input should not be processed

Output

For each case, output a line containing the number of possible selections can be made to form an area whose sides are parallel to the grid.

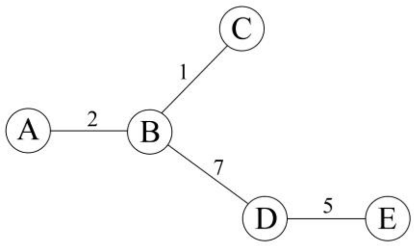
Sample Input	Output for Sample Input
7 10 1 3 1 6 2 2 2 5 4 2 4 5 5 1 5 6 6 3 6 6 3 6 2 1 2 2 2 3 3 1 3 2 3 3 0 0	2 3

Problem H

Tree Median

Time Limit: 1s

A tree is a connected graph containing no cycles. A vertex is called a median vertex if the total cost to reach all vertices is minimal. There could be more than one median vertex in a tree, that's why we define median as the set of all median vertices. To find median in a tree with small number of vertices is fairly easy task as you can solve this by a simple brute force program.



In the left figure, we can see a weighted tree with 5 vertices. The tree median is {B} because the total cost from vertex B to all other vertices is minimal.

B-A = 2 B-D = 7
B-C = 1 B-E = 7 + 5 = 12

TOTAL = 2 + 1 + 7 + 12 = 22

What if the number of vertices is quite large? This might be a problem since brute force program cost too much time. Given a weighted tree with N vertices, output the total cost to reach all vertices from its median.

Input

Input consists of several cases. Each case begins with an integer n ($1 \le n \le 10{,}000$) denoting the number of vertices in a tree. Each vertex is numbered from 0… n-1. Each of the next n-1 lines contains three integers: a, b, and w ($1 \le w \le 100$), which means a and b is connected by an edge with weight w.

Input is terminated when n is equal to 0. This input should not be processed.

Output

For each case, output a line containing the sum of cost of path to all other vertices from the tree median.

Sample Input	Output for Sample Input
5 0 1 2 1 2 1 1 3 7 3 4 5 6 0 1 1 1 2 4 2 3 1 3 4 4 4 5 1 0	22 21