

# icpc International Collegiate Programming Contest

## The 2024 ICPC Greater NY Regional Contest

# Problem Set



---

icpc global sponsor  
programming tools



---

upsilon pi epsilon  
honor society

# ICPC NA East Division

## Problems

- A A Stack of Gold
- B Average Substring Value
- C Balancing Art
- D Brownian Bears
- E Fences Make Good Neighbors
- F Leapfrog Encryption
- G Letter Balloons
- H Marching Orders
- I Ooh I See
- J Pascal Meets Boole
- K Pony-less Express
- L Repetitive Routes
- M Smart Password Validation

Do not open before the contest has started.

This page is intentionally left blank.

# Problem A

## A Stack of Gold

Time Limit: 1 second, Memory limit: 2G

Lt. Columbo, arguably the best detective in the world, was faced with a problem. He was asked: if he was placed in a room with some number of stacks of gold-colored coins, where all but one stack consisted solely of tungsten coins, and the remaining stack consisted solely of pure gold coins, how could he determine which stack had the pure gold coins? Oh, excuse me, there's just one more thing . . . he was also told he is given a modern penny scale and one penny. A modern version of the penny scale operates as follows: when you deposit a penny and put an object (or objects) on the scale, the machine displays the weight on a digital display. The scale is extremely accurate, down to milligrams. Every coin is the same size and color, but the tungsten coins weigh 29 260 mg each and the gold coins weigh 29 370 mg each.

Being the genius that he is, Columbo came up with the solution to figure out which stack contained the gold coins using a single weighing (since he has only one penny for the penny scale). As an example, suppose there are four stacks of coins labeled 1 through 4. If he took one coin from stack 1, two coins from stack 2, three coins from stack 3 and four coins from stack 4, and weighed those 10 coins together on the penny scale, he could determine which stack had the gold coins. "How?" you might ask. Suppose all four stacks were tungsten. The total weight of the 10 coins would be 292 600 mg. If, say, stack 1 was gold (one gold coin in the pile that was weighed), the total weight would be 292 710 mg. If stack 2 was gold (two gold coins in the pile that was weighed), the total weight would be 292 820 mg. So, if you know the number of coins being weighed and the total weight of those coins, you can determine which stack has the gold coins. Just as a reminder, given a number of stacks,  $s$ , the number of coins,  $c$ , being weighed will be:

$$c = \frac{s(s+1)}{2}$$

### Input

Input consists of a single line containing two integers  $w$   $s$ , where  $w$  ( $87\,890 \leq w \leq 147\,774\,000$ ) is the weight in milligrams reported by the scale and  $s$  ( $2 \leq s \leq 100$ ) is the number of coin stacks. The stacks are labeled 1 to  $s$  with  $i$  coins from stack  $i$  moved to the scale for  $1 \leq i \leq s$ . For given values of  $w$  and  $s$ , it will always be possible to determine which stack is composed solely of gold coins.

### Output

Output consists of a single positive integer indicating which stack contains the gold coins.

#### Sample Input 1

292930 4

#### Sample Output 1

3

#### Sample Input 2

147774000 100

#### Sample Output 2

100

This page is intentionally left blank.

# Problem B

## Average Substring Value

Time Limit: 1 second, Memory limit: 2G

Let  $s$  be a nonempty string consisting entirely of base-10 digits (0–9). If the length of  $s$  is  $n$ , number the digits  $1, 2, 3, \dots, n$  from left to right, and for  $1 \leq i \leq j \leq n$ , let  $s[i, j]$  denote the substring consisting of the digits from position  $i$  to position  $j$ , inclusive. (It follows that we are only considering *nonempty* substrings.) Assign a value to each substring that is simply equal to the largest digit in the substring. What is the average value of the substrings of  $s$ ?

Note that two different substrings may be identical (as strings), but for the purposes of this problem they are treated as distinct. For example, if  $s = 1010$ , then  $s[1, 2] = s[3, 4] = 10$  are distinct substrings (both with value 1).

### Input

The input is a single nonempty string,  $s$ , of base-10 digits. The length of  $s$  is at most 200 000.

### Output

Output a line containing the average value of the substrings of  $s$ . If the average is an integer, print the integer. If the average is a proper fraction, i.e., is equal to  $a/b$ , where  $a$  and  $b$  are positive integers and  $a < b$ , print this fraction in lowest terms, with a '/' symbol separating the numerator and denominator. If the average is greater than 1 and does not simplify to an integer, print the whole part followed by the proper fractional part, separated by a space, with the proper fractional part in lowest terms and formatted as described in the previous sentence.

#### Sample Input 1

123
-----

#### Sample Output 1

2 1/3
-------

#### Sample Input 2

4084
------

#### Sample Output 2

6
---

#### Sample Input 3

1010
------

#### Sample Output 3

4/5
-----

#### Sample Input 4

00000
-------

#### Sample Output 4

0
---

This page is intentionally left blank.

# Problem C

## Balancing Art

Time Limit: 8 seconds, Memory limit: 2G

Pete Tencious is a world-renowned artist who specializes in mobiles. The San Francisco Museum of Art is currently displaying a collection of his artwork entitled Balance I, Balance II, Balance III, . . . , you get the idea. Each of these works contains two or more spheres with wires strung between them. At the ends of each wire are a number of disks that are attached to the two spheres that the wire connects. When you sum up the number of disks at each sphere you get the same number (hence the word “Balance”) — Pete calls this the *balance number* for each mobile. On some of these wires there are also extra disks suspended between spheres. The artist says this work represents the balance of nature (the disks attached to the spheres) corrupted by the influence of humankind (the extra disks between the spheres). Clearly Pete has a much better impression of nature than people.

The funny thing is that Mother Nature decided to step in and throw Pete a curve. A minor earthquake in the Bay area has dislodged the disks, and they are now all hanging loose between spheres. Pete is currently working on Balance CCXLI and can’t be reached, so the museum curators have to fix the mobiles themselves. They can’t remember exactly what the balance number should be for each mobile, but they decide it should be as large as possible, leaving as few extra disks in between the spheres as possible (they apparently have a little better view of humankind than you-know-who). However, determining the minimum number of disks left between the spheres is a bit difficult, so they have come to you for help.

Figure C.1 shows the state of one mobile after the earthquake, corresponding to the first sample input. Figure C.2 shows one way the disks could be moved by the curators so that each sphere has the same number of disks, while leaving the minimum number of disks between the spheres.

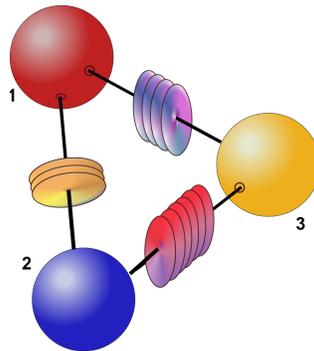


Figure C.1: After the earthquake

### Input

The first line of input contains two positive integers  $n$   $m$ , where  $n$  ( $2 \leq n \leq 200$ ) is the number of spheres in a mobile (numbered 1 to  $n$ ) and  $m$  ( $1 \leq m \leq 500$ ) is the number of wires connecting the spheres. This is followed by  $m$  lines of the form  $s_1$   $s_2$   $d$ , where  $s_1$  and  $s_2$  ( $1 \leq s_1, s_2 \leq n, s_1 \neq s_2$ ) are two spheres connected by a wire and  $d$  ( $0 \leq d \leq 10\,000$ ) is the number of disks hanging on the wire after the earthquake. There is at most one wire between any two spheres.

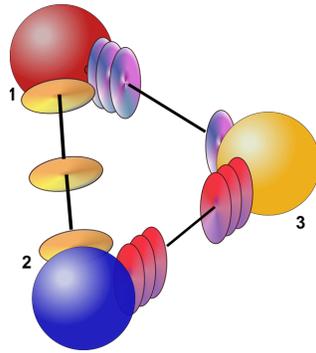


Figure C.2: Disks moved to maximize balance number

## Output

Output the minimum number of disks left hanging on the wires between spheres after the maximum balance number has been reached.

### Sample Input 1

```
3 3
1 2 3
1 3 4
2 3 6
```

### Sample Output 1

```
1
```

### Sample Input 2

```
5 4
1 2 2
1 5 2
2 3 2
2 4 20
```

### Sample Output 2

```
16
```

# Problem D

## Brownian Bears

Time Limit: 4 seconds, Memory limit: 2G

Dr. Ursula Major is an internationally renowned expert in the study of bears, specifically brown bears, which are known in many parts of North America as grizzly bears. She is most famous for her discovery of an extremely rare brown bear subspecies — the *Brownian bear*, whose feeding behavior seems to be guided by an intriguing mixture of regularity and randomness.



Image by Alex Taylor; Used with permission

At present, Dr. Major is studying a pair of Brownian bears living on a narrow strip of land that is oriented east-west. This territory has  $n$  evenly spaced locations where the bears can forage for berries, and Dr. Major has labelled these locations  $1, 2, \dots, n$  from west to east. Every morning, each Brownian bear wakes up in one of the  $n$  locations and randomly chooses to move either east or west to the neighboring location. What is remarkable is that the probability of moving in either direction is exactly 50%! (Dr. Major's working theory is that this is rooted in some kind of quantum mechanical phenomenon in the bears' brains.) If a bear happens to start the day in one of the end locations ( $1$  or  $n$ ), then it will randomly choose between moving to the sole neighboring location or staying where it is. After making its choice, the bear spends the day foraging for berries in the chosen location, and then also sleeps there that night. The next morning, the process begins all over again.

Dr. Major is particularly interested in days when the two bears forage together, i.e., in the same location, so she plans to observe them over a period of time, possibly as long as a month (or until her funding runs out). At the beginning of the first observation day, the two bears wake up in different locations. Can you help Dr. Major determine the probability that the two bears will forage together for at least one day during the course of her experiment?

Note that the two bears can move past each other without foraging together. For example, if the bears start a particular day in locations  $4$  and  $5$ , and if the bear in location  $4$  moves east and the bear in location  $5$  moves west, then the bears simply pass each other that morning without foraging together.

### Input

The input consists of a single line containing four integers,  $n x y d$ , where  $n$  ( $2 \leq n \leq 100$ ) is the number of locations,  $x$  and  $y$  ( $1 \leq x, y \leq n, x \neq y$ ) are the locations where the two bears wake up at the beginning of the first observation day, and  $d$  ( $1 \leq d \leq 31$ ) is the number of observation days.

### Output

Output a line containing the probability that the two bears forage together for at least one day during the  $d$  observation days. Express this probability as a fraction  $a/b$  in lowest terms, where  $a$  is a non-negative integer and  $b$  is a positive integer. See the sample outputs for examples.

**Sample Input 1**

4 1 2 2
---------

**Sample Output 1**

3/8
-----

**Sample Input 2**

6 2 5 2
---------

**Sample Output 2**

0/1
-----

**Sample Input 3**

2 1 2 1
---------

**Sample Output 3**

1/2
-----

# Problem E

## Fences Make Good Neighbors

Time Limit: 4 seconds, Memory limit: 2G

The King of Gletrian has a large estate that he wishes to divide up into triangular parcels which he then plans to give to deserving (that is, wealthy) followers. The estate is a convex polygon and already has fences around its borders, so the only cost now will be putting in new fences to perform the partitioning of the land. All fences will lie on straight lines between existing corners of the estate and no two fences will cross each other. Being fiscally minded (that is, cheap) he wants to put in the minimum amount of fencing possible.

But there is a problem (there always seems to be, doesn't there!). He has two sons who already have homes on the estate. One is a cute young lad and the other is a bit obtuse, but the problem lies in that they don't really get along with each other. Because of this, placing a single fence between their two parcels of land is out of the question — there will be constant bickering between them and at worst some type of physical altercation. However, the King has hope that his two sons might eventually learn to appreciate one another and he feels that all is needed is one good arbiter to serve as a liaison between them. To accomplish this the King wants to place exactly two fences between the brother's parcels, separating the brothers' lands by a single parcel where he'll place some voluntary (that is, conscripted) person to serve as go-between. Given these constraints, the King still wants to minimize the cost of the project, which means minimizing the length of fencing used. In addition, to avoid the brothers' houses, no potential fence which passes directly through a brother's location may be used in the triangulation.

An example (corresponding to Sample Input 1) is shown in Figure E.1, where the brother's locations are indicated by the two plus signs. The partitioning to the right, while using less fencing, is not a solution since there are more than two fences between the brothers' locations. A correct triangulation is shown on the left.

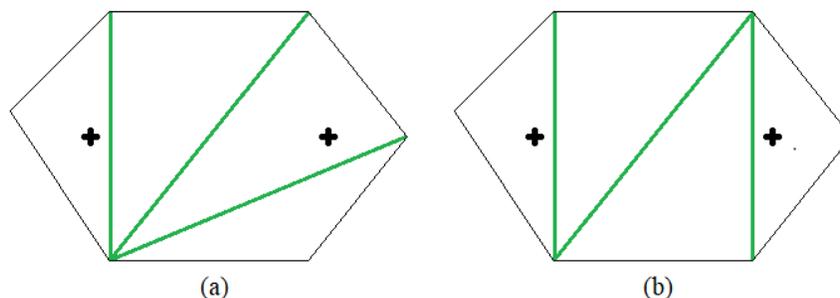


Figure E.1: Sample Input 1. (a) Correct solution. (b) Incorrect solution.

### Input

Input starts with a single integer  $n$  ( $6 \leq n \leq 500$ ), indicating the number of corners of the estate. Following this are  $n$  pairs of integers  $x_i y_i$  ( $|x_i|, |y_i| \leq 3000$ ) specifying the location of each corner, given in clockwise order. No two corner locations are the same and the polygon formed by connecting these corners is convex. No three consecutive corner locations are collinear. The last two lines each contain a pair of coordinates: the first of these lines contains  $bx_1 by_1$  ( $|bx_1|, |by_1| \leq 3000$ ), indicating the location of the first brother, and the second contains  $bx_2 by_2$  ( $|bx_2|, |by_2| \leq 3000$ ), indicating the location of the second brother. The two

brother's locations are distinct and lie within the interior of the polygon. All coordinates are in kilometers.

## Output

Output the minimum length of fencing in kilometers needed to satisfy all the constraints specified above. Answers within an absolute error of  $10^{-3}$  of the judges' answer will be deemed correct. If it is not possible to satisfy the conditions stated above, output the word `IMPOSSIBLE`.

Sample Input 1	Sample Output 1
6 0 -50 -40 10 0 50 80 50 120 0 80 -50 -10 0 100 0	354.553591
Sample Input 2	Sample Output 2
6 0 -50 -30 0 0 50 90 50 120 0 90 -50 0 5 90 -5	IMPOSSIBLE
Sample Input 3	Sample Output 3
12 0 100 50 86 86 50 100 0 86 -50 50 -86 0 -100 -50 -86 -86 -50 -100 0 -86 50 -50 85 60 60 -60 -60	1113.370332

# Problem F

## Leapfrog Encryption

Time Limit: 1 second, Memory limit: 2G

We've come up with a new encryption method we call Leapfrog Encryption. It is a key-based encryption scheme where an alphabetic key specifies how letters in the *plaintext* (the text to be encrypted) are placed in the *ciphertext* (the resulting encrypted string). Here's how Leapfrog Encryption works:

1. Remove all non-alphabetic characters from the plaintext and convert all remaining letters to lowercase.
2. Convert the letters of the key to their location in the alphabet +1 (so that 'a' converts to 2, 'b' converts to 3 and so on). This gives us a sequence of numbers  $d_1, d_2, \dots, d_n$  where  $n$  is the length of the key.
3. Going left-to-right, place the first letters in the plaintext in every  $d_1$ -th location of the ciphertext until you run out of positions in the ciphertext (where the length of the ciphertext is equal to the number of letters in the plaintext). So for example, if  $d_1 = 5$  the first letter in the plaintext goes in position 5 of the ciphertext (numbering the first location in the ciphertext as position 1), the second letter in the plaintext goes in position 10 in the ciphertext, and so on.
4. Repeat this with  $d_2$  but this time going right-to-left through the ciphertext, only counting the empty positions (leapfrogging over the letters already in the ciphertext).
5. Continue with  $d_3, d_4$ , etc., alternating the direction you go through the ciphertext each time.
6. If there are still letters left in the plaintext after using  $d_n$ , fill in the remaining empty locations in the ciphertext with these remaining letters, again going in the opposite direction of the previous pass (this is equivalent to having  $d_{n+1} = 1$ ).

For example, if our plaintext is "Send more monkeys!" and our key is "bea", the encryption proceeds as follows:

```

b → 3, left-to-right:  _ _ s _ _ e _ _ n _ _ d _ _ m
e → 6, right-to-left:  _ _ s _ _ e o _ n _ _ d _ _ m
a → 2, left-to-right:  _ r s _ e e o _ n m _ d o _ m
last pass, right-to-left: s r s y e e o e n m k d o n m

```

Decryption is done by . . . hey, you know what? We're going to let you figure that out.

### Input

The first line of input contains two strings  $t k$  where  $t$  is either E or D indicating whether to perform encryption or decryption and  $k$  is the lowercase alphabetic key. The length of  $k$  will be between 1 and 100, inclusive. The second and final line of input contains the plaintext to encrypt (if  $t$  is E) or the ciphertext to decrypt (if  $t$  is D). This string is non-empty and has a maximum length of 2 000. A ciphertext string consists of lowercase letters only, while a plaintext string may contain uppercase letters, numbers, punctuation and spaces as well (all counted as part of the length of the string) and is guaranteed to contain at least one letter.

## Output

Output the encrypted or decrypted text. Your output should only contain lowercase letters.

### Sample Input 1

```
E bea  
Send more monkeys!
```

### Sample Output 1

```
srsyeeoenmkdonm
```

### Sample Input 2

```
D bea  
srsyeeoenmkdonm
```

### Sample Output 2

```
sendmoremonkeys
```

### Sample Input 3

```
D zyxwvutsrqponmlkjihgfedcba  
lafogrpe
```

### Sample Output 3

```
leapfrog
```

# Problem G

## Letter Balloons

Time Limit: 1 second, Memory limit: 2G

You are organizing a programming contest, and have decided that the first team to solve each problem will get a balloon in the shape of the problem's letter. For example, suppose there are twenty-three problems in the contest, labeled A through W. The team members for Wossa Motta University are hoping to be first to solve problems M, U, and W so that they can fly their university's initials above their programming station. In fact, a lot of teams have the same idea: try to be first-solvers of problems that spell out their school's abbreviation. It might be possible for both Wossa Motta U. and the Spittinyer Institution to achieve this goal, but neither will be able to do so if Muddinyer Institute manages to solve problems M and I before they do (we are assuming there will never be a tie for the first solution to any problem). On the other hand, Muddinyer I. can't achieve it if either Wossa Motta U. or Spittinyer I. solves M or I first. Schools like Toe Tac Tech are out of luck no matter what, since a team can get at most a single letter balloon for any problem. And Xerxes College is also out of luck because there is no problem X in this example.



NightCafe

You've been wondering—what is the maximum number of teams that can proudly display their school's initials with “first-solver” balloons at the end of the contest?

### Input

The first line of input contains two integers  $p$   $t$ , where  $p$  ( $1 \leq p \leq 26$ ) is the number of problems in the contest, and  $t$  ( $1 \leq t \leq 20$ ) is the number of teams. Problems are labeled with the first  $p$  letters of the alphabet. Each of the following  $t$  lines contains a nonempty string of at most 80 uppercase letters describing a school's initials. There is only one team from each school, but several schools may have the same initials.

### Output

Output a single integer consisting of the maximum number of teams that can be first solvers of all the problems that form the initials of their school name.

#### Sample Input 1

```
23 5
WMU
SI
MI
TTT
XT
```

#### Sample Output 1

```
2
```

**Sample Input 2**

6 6  
ABC  
BDE  
ABE  
BF  
BF  
CEF

**Sample Output 2**

1

# Problem H

## Marching Orders

Time Limit: 1 second, Memory limit: 2G

Dean Bob Roberts is in charge of the order in which the professors of his college march in the graduation ceremonies. Because of complaints among certain professors from the newly created DEI Studies Department, it has been decided that the order in which they march should not be based on seniority but should be random. Bob thinks this is fine, and to be totally transparent he has communicated the following method he will use to create the marching list: He starts with an alphabetically ordered list of  $n$  professors numbered  $0, 1, \dots, n - 1$  and a non-negative integer  $m < 10^9$ . Then the first person in the marching list is the one in position  $m \bmod n$  in the alphabetic list. This shortens the alphabetic list by one, shifting down all those in positions greater than  $m \bmod n$ . The second person in the marching list is the one in position  $m \bmod (n - 1)$ , and so on.

For example, assume we have 6 professors A, B, C, D, E and F. If  $m = 11679$ , then the marching list is created as follows:

$n$	$m \bmod n$	alphabetic list	marching list
6	3	A B C D E F	D
5	4	A B C E F	D F
4	3	A B C E	D F E
3	0	A B C	D F E A
2	1	B C	D F E A C
1	0	B	D F E A C B

This sounds fair, but is not as transparent as some professors would like as Dean Roberts does not make public the value of  $m$  that he uses. This makes it difficult to determine if he is actually following the method he has stated or has just selected the marching order based on his personal whims and biases. What the faculty would like to know is, for a given marching order, is there a value of  $m$  which would produce that order?

### Input

The first line of input contains a single decimal integer  $n$  ( $5 \leq n \leq 20$ ), the number of professors who will be marching. The second line consists of a string containing a permutation of the first  $n$  uppercase letters of the alphabet, giving the proposed marching order.

### Output

If the given marching order could not have come from the algorithm, output a single line containing the word NO. Otherwise, output two lines, the first containing the word YES and the second containing the smallest non-negative value of  $m$  which could produce the given marching order.

**Sample Input 1****Sample Output 1**

6 DFEACB	YES 39
-------------	-----------

**Sample Input 2****Sample Output 2**

7 DFEGACB	NO
--------------	----

# Problem I

## Oooh I See

Time Limit: 1 second, Memory limit: 2G

Captain O'Capten has hidden some treasure and created a map to mark where it is buried. Rather than using 'X' to mark the spot, he has decided to obfuscate the location by using a grid of uppercase O (the letter O) characters and 0 (the number 0) characters. The treasure's position is given by the location of a 0 character surrounded on all sides by 8 O characters. That is, a 0 character with an O immediately above, below, to the left, to the right, diagonally above to the left, diagonally above to the right, diagonally below to the left, and diagonally below to the right.

Captain O'Capten wants to recover the location of his treasure but he is finding his map hard to read (huh, go figure). Help Captain O'Capten find the location of his treasure, or exclaim Oh no! if such a location is not marked on the map or there is more than one such location.

### Input

The first line contains two integers  $r$  and  $c$  ( $1 \leq r, c \leq 50$ ), where  $r$  indicates the number of rows and  $c$  indicates the number of columns of characters in the map. Rows are numbered 1 to  $r$ , top to bottom, and columns are numbered 1 to  $c$ , left to right. This is followed by  $r$  lines, each with  $c$  characters, where each character is either O or 0.

### Output

If there is no 0 character surrounded on all sides by 8 O characters, then output one line consisting of the exclamation, Oh no!. If there is more than one 0 character surrounded on all sides by 8 O characters, then output one line consisting of the exclamation, Oh no! N locations, with the number of locations instead of N. If there is exactly one 0 character surrounded on all sides by 8 O characters, then output one line containing two integers. The first integer is the index of the row of the 0 character and the second integer is the index of the column of the 0 character.

#### Sample Input 1

```
3 6
000000
000000
000000
```

#### Sample Output 1

```
2 5
```

**Sample Input 2**

```
5 3
000
000
000
000
000
```

**Sample Output 2**

```
Oh no! 2 locations
```

**Sample Input 3**

```
4 4
0000
0000
0000
0000
```

**Sample Output 3**

```
Oh no!
```

# Problem J

## Pascal Meets Boole

Time Limit: 1 second, Memory limit: 2G

Many people are familiar with Pascal's Triangle, a triangular arrangement of integers named after the French mathematician and philosopher Blaise Pascal (1623–1662). If we number the rows of Pascal's Triangle  $1, 2, 3, \dots$ , starting from the top, then row  $r$  contains  $r$  elements, which we will number  $1, 2, \dots, r$  from left to right. The 1<sup>st</sup> and  $r$ <sup>th</sup> elements in row  $r$  are set equal to 1, and for  $r \geq 3$  and  $1 < i < r$ , element  $i$  in row  $r$  is the sum of elements  $i - 1$  and  $i$  in row  $r - 1$ . More informally, each “non-edge” element is the sum of the two elements immediately above it. Figure J.1(a) depicts the first 8 rows of Pascal's Triangle.

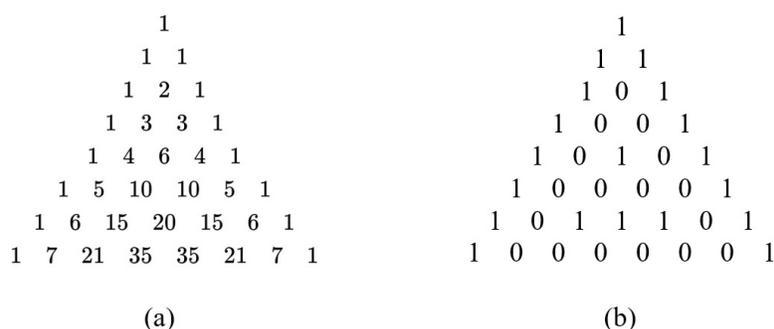


Figure J.1: (a) Pascal's Triangle, (b) Pascal-Boole triangle for function 1000

But what if we consider a rule other than standard addition for combining values? Since the edge elements are bits (1's), a natural option is to use any two-input Boolean function, named after the English mathematician and philosopher George Boole (1815–1864). For example, the Boolean function given by the following truth table generates the triangle depicted in Figure 1(b) (where we also show the first 8 rows). In this truth table,  $x$  and  $y$  correspond to elements  $i - 1$  and  $i$ , respectively, in row  $r - 1$ , and  $f(x, y)$  is the resulting element  $i$  in row  $r$ .

$x$	$y$	$f(x, y)$
0	0	1
0	1	0
1	0	0
1	1	0

In general, if we label the bits in the rightmost column of any such truth table  $b_{00}, b_{01}, b_{10}, b_{11}$  from top to bottom, then we can compactly represent a two-input Boolean function by the 4-bit string  $b_{00}b_{01}b_{10}b_{11}$ . So the example function above is represented by 1000.

Your challenge is to answer two kinds of questions about “Pascal-Boole” triangles:

1. For a given Boolean function,  $f$ , what is the bit in row  $r$ , position  $i$ ?
2. For a given Boolean function,  $f$ , how many 1's are there in the first  $r$  rows?

## Input

The first line of input contains an integer,  $n$  ( $1 \leq n \leq 250$ ), the number of test cases. This is followed by  $n$  lines, each of which has one of two forms:

1.  $f \text{ B } r \ i$
2.  $f \text{ N } r$

In both cases,  $f$  is a 4-bit binary string representing a two-input Boolean function, and  $r$  is an integer ( $1 \leq r \leq 10^6$ ). In the first case,  $i$  is an integer ( $1 \leq i \leq r$ ).

## Output

For a test case of the form  $f \text{ B } r \ i$ , output a line containing the bit in row  $r$ , position  $i$  of the Pascal-Boole triangle generated using  $f$ . For a test case of the form  $f \text{ N } r$ , output a line containing the number of 1's in the first  $r$  rows of the Pascal-Boole triangle generated using  $f$ .

Sample Input 1	Sample Output 1
3	1
1000 B 5 3	28
1111 N 7	0
0100 B 6 4	

# Problem K

## Pony-less Express

Time Limit: 8 seconds, Memory limit: 2G

Howdy, pardner! You've been put in charge of mail delivery in these here parts. Your headquarters is in Capital City, and Miss Penelope has asked us to tell all of the local farmsteads about her fashionable doin's. The roads hereabouts were built so that each farmstead connects to Capital City by only one set of roads, which may connect through multiple other farmsteads. Each road takes exactly one day to travel on a horse.

Each farmstead wants to receive news on their own specific timetable (so as not to distract the hired hands from their work), and gets peeved if the delivery misses the desired date, either early or late. Specifically, each farmstead  $i$  has a perfect date  $D_i$  for receiving news. If the news arrive on day  $d$ , the anger level at the farmstead is  $C_i(D_i - d)^2$ . Obviously, we want to minimize the total anger level over all the farmsteads, so we need to get on our horses and send this important information to each farmstead as close to their desired date as possible! The only problem is ... well ... we actually don't HAVE any horses. Instead, we can requisition exactly one horse from each location (farmstead or Capital City) each day, ride it for 1 day, and let it return to its starting location at the end of the day (don't worry, these horses know how to find their way back). The next day, another rider can take that horse and travel to a different location (if necessary). This process repeats in all of the farmsteads as well as Capital City.

Oh, and one more thing: we ain't payin' riders to sit idle at any farmstead, gittin' into who knows what kind of trouble. So if news arrives at farmstead  $i$  on day  $d$  and there are  $m$  roads leading to  $m$  other farmsteads that have not received any news yet, a horse will leave farmstead  $i$  on each day  $d + 1, d + 2, \dots, d + m$  until every neighboring city has been visited, even if waiting extra days might lead to lower anger levels for some of the farmsteads.

For example, consider the layout of farmsteads shown in Figure K.1 (corresponding to Sample Input 1), where the  $C_i, D_i$  values are shown to the left of each farmstead. The optimal way to deliver the news is as follows:

- Day 1: send a horse from Capital City to farmstead 3
- Day 2: send a horse from Capital City to farmstead 1 and a horse from farmstead 3 to farmstead 7
- Day 3: send a horse from Capital City to farmstead 2, a horse from farmstead 1 to farmstead 4 and a horse from farmstead 3 to farmstead 6
- Day 4: send a horse from farmstead 1 to farmstead 5

The total anger level, adding up from farmstead 1 to 7, is  $3(1 - 2)^2 + 3(2 - 3)^2 + 2(2 - 1)^2 + 2(4 - 3)^2 + 1(6 - 4)^2 + 3(3 - 3)^2 + 4(1 - 2)^2 = 3 + 3 + 2 + 2 + 4 + 0 + 4 = 18$ .

Can you help us figure out just how angry people will be in total, so we know what we're in fer?

### Input

The first line of input contains a positive integer  $n$  ( $n \leq 200$ ) indicating the number of farmsteads, numbered 1 to  $n$ , with Capital City labeled 0. Following this are  $n$  lines each containing three integers  $c d r$  ( $1 \leq c \leq 20, 1 \leq d \leq n, 0 \leq r \leq n$ ), where the  $i^{th}$  of these lines specify the  $C_i$  and  $D_i$  values for the  $i^{th}$  farmstead

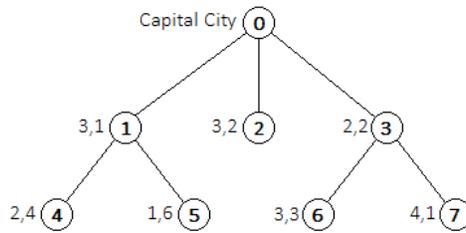


Figure K.1: Example layout.

and  $r$  indicates that there is a road from farmstead  $i$  to farmstead  $r$  (or Capital City if  $r = 0$ ). The roads are set up so that there is a unique path between any two farmsteads, and between each farmstead and Capital City.

## Output

Output the minimum anger level that can be achieved assuming the first horse leaves Capital City and arrives at a first farmstead on day 1.

### Sample Input 1

7
3 1 0
3 2 0
2 2 0
2 4 1
1 6 1
3 3 3
4 1 3

### Sample Output 1

18
----

### Sample Input 2

3
5 2 2
4 1 0
6 3 1

### Sample Output 2

0
---

# Problem L

## Repetitive Routes

Time Limit: 8 seconds, Memory limit: 2G

Tory operates a dial-in mobility service where customers can book a vehicle to come and pick them up from one location and drive them to another location to be dropped off. The vehicles used can seat many passengers, so sometimes additional stops are made along the way to pick up and drop off other passengers. Users of the service are generally tolerant of some inefficient routes, but they are less tolerant of revisiting a location that they have visited at least once on their journey already. Tory has come up with a sequence of pickups and drop offs to serve all of the passengers but he wants to know how many complaints he is likely to receive. From past experience, Tory knows that he can expect one complaint for each time a customer returns to a location that they have already been to on their journey. Note that this means that the same customer may lodge multiple complaints, and possibly for visiting the same location more than two times! Given a sequence of pickups and drop offs and their locations, determine how many complaints Tory can expect to receive.

The customer's pickup location and drop off location both count towards locations that they may return to and complain about. Consecutive pickups or drop offs at the same location also count as repeated visits to the same location for anyone in the vehicle at the time. The customer's drop off location may be the same as their pickup location. As per above, the customer will complain about this!

### Input

The first line of input is the number of customers,  $n$  ( $1 \leq n \leq 200\,000$ ). This is followed by  $2n$  lines, each consisting of two integers. The first is a customer number from 1 to  $n$  and the second represents a location and is in the range 1 to  $2n$ . Distinct location numbers correspond to distinct locations. Each customer number will appear exactly twice. The first occurrence of customer number  $C$  corresponds to the pickup of customer  $C$  and the second occurrence of customer number  $C$  corresponds to the drop off. All lines in between correspond to locations visited and pickups and drop offs completed while customer  $C$  is in the vehicle. Customer 1 will be the first customer to be picked up. Customer  $C$  will be picked up before customer  $C + 1$  is picked up. Similarly, location 1 will be the pickup location of customer 1 and a location  $L + 1$  will only be visited if location  $L$  has been visited. There is no limit on the number of passengers that can be in the vehicle at the same time.

### Output

Output a single integer, the number of complaints Tory can expect to receive for the sequence of pickups and drop offs.

**Sample Input 1**

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

**Sample Output 1**

```
5
```

**Sample Input 2**

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

**Sample Output 2**

```
16
```

# Problem M

## Smart Password Validation

Time Limit: 1 second, Memory limit: 2G

People often make errors when entering passwords for one of their many accounts. This is becoming more of a problem with the rather lengthy passwords that many online portals require (the more characters in the password, the more chance there is of making a typing mistake). Practitioners of the “touch typing” technique are prone to make serious errors if one of their hands is not in the proper home row position when typing, since touch typists do not need to look at the keyboard as they type. This can cause all characters typed by a misplaced hand to be “one key off”. E.g., on a QWERTY keyboard, “lion” might become “kuib” if the right hand is placed one key to the left of normal home position.

The Frobozz Magic Security Company has decided to implement a new password validation algorithm that takes into account some common mistakes (but not all) made by folks when entering passwords on a QWERTY-style keyboard (see Figure M.1).

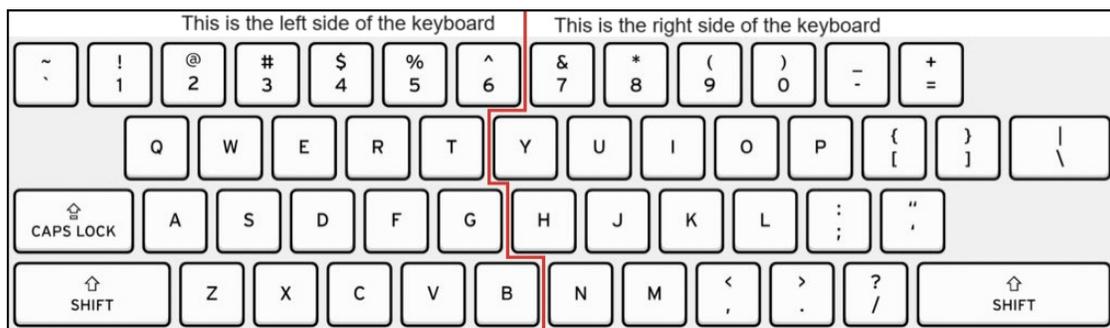


Figure M.1: QWERTY keyboard

The effect of pressing CAPS LOCK is limited to alphabetic characters and toggles the effect of the shift key for those characters. In others words, when CAPS LOCK is not in effect, all letters are typed as lowercase unless one or both shift keys are pressed to produce an uppercase letter. The opposite behavior occurs when CAPS LOCK is on—uppercase is the default unless one or both shift keys are pressed, producing lowercase letters. The algorithm designers assume that the state of the CAPS LOCK key will not change in the middle of entering a password.

Their algorithm looks for deviations from a correct sequence of keystrokes. For example, if the correct password is “ALg\*”, the correct sequence, using the keyboard above, is “SHIFT-A, SHIFT-L, G, SHIFT-8”. If the user were to accidentally hit the CAPS LOCK key before entering the password, but then used the correct keystroke sequence, they would end up typing “aLg\*”. Likewise, if a touch typist accidentally placed the fingers of the left hand one space to the right of the correct position, made no other errors, and attempted to follow the correct sequence (unaware of the shift in finger positions), they would type “sLh\*”. And if, in addition, they also accidentally activated the CAPS LOCK key before entering the password, they would type “sLH\*”. Finally, a user might accidentally insert or delete a character. For example, the password “ALg\*” might be mistyped as “AL\*” or “ALLg\*”.

The Frobozz Magic Security Company has decided to permit certain deviations, or combinations of deviations, from the correct sequence, as described in Figure M.2. At most one of the LS, RS, EC and MC errors

Name	Acronym	Description	Examples
Left side off by one	LS	All characters on the left side of the keyboard are shifted right by one key. The left-shift key shifted right by one key is still the left-shift key.	FlatHead $\Rightarrow$ GlsyHrsf
Right side off by one	RS	All characters on the right side of the keyboard are shifted left by one key. The right-shift key shifted left by one key is still the right-shift key.	FlatHead $\Rightarrow$ FkatGead
Single extra character	EC	The entered password has an extra character	Zorkmid $\Rightarrow$ Zorkmiid (extra i)
Single missing character	MC	The entered password is missing a character.	FCD#3 $\Rightarrow$ FCD3 (missing the #)
<b>CAPS LOCK</b> was in effect by mistake	CL	All lowercase letters appear as uppercase and all uppercase letters appear as lowercase	WE do it! $\Rightarrow$ we DO IT!

Figure M.2: Permitted Error Types

is permitted, but any of these may occur in conjunction with a CL error.

## Input

There are multiple lines of input. The first line contains a string,  $p$ , which is the correct password ( $2 \leq \text{length}(p) \leq 24$ ). The second line contains a single positive integer  $n$  ( $1 \leq n \leq 1000$ ), which is the number of lines that follow. Each of the following  $n$  lines contains a single password to test,  $t_i$  ( $\text{length}(p) - 1 \leq \text{length}(t_i) \leq \text{length}(p) + 1$ ). The characters for all passwords come only from those that appear on the QWERTY keyboard picture above, as well as the lowercase letters.

## Output

Output consists of  $n$  lines. For each  $t_i$  the line should consist of the word YES if  $t_i$  matches the password  $p$  using Frobozz's algorithm, NO if it does not.

### Sample Input 1

```

Password= `A: '
9
Password= `A: '
pASSWORD= `a: '
Psddeotf=1S: '
oASSWIRD- `al;
Password= `A: ' x
pASWORD= `a: '
pASSWORD= `a: ' X
Osddeitf-1SL;
Psdeotf=1S: '

```

### Sample Output 1

```

YES
YES
YES
YES
YES
YES
YES
NO
NO

```