Thirty-eighth Annual
University of Central Florida

High School Programming Tournament

Problems

Problem Name	Filename
So !!!False	sotrue
Placing Gazebos	placing
Missing You from Letter Land	letter
Elephant Espionage	espionage
Penpen the Hero	hero
Go with the Flow	flow
Radioactive Decay	decay
Building Gazebos	building
1999, Bizarre Summer	summer
Pseudonym Predicament	pseudonym
Fox on Clocks on Bricks and Blocks	fox
Space Invaders	invaders

Call your program file: *filename*.c, *filename*.cpp, *filename*.java, or *filename*.py
Call your Java class: *filename*

For example, if you are solving Penpen the Hero:
Call your program file: hero.c, hero.cpp, hero.java, or hero.py
Call your Java class: hero

Notes

1. All solutions must read from standard input and write to standard output. The judges will ignore all output sent to standard error.

2. All input sets used by the judges will follow the input specification in each problem. You do not need to test for input that violates the input format specification.

3. Submitted programs will be tested on additional judge cases and not just the sample cases given in the problem.

4. If you need to use the value for π, use the value 3.141592653589793.

5. If you want to know how to compute the absolute error, given the actual value of a quantity, *x*, and the measured value of a quantity, x_0, then the absolute error value, Δx, is calculated as:

$$\Delta x = |x_0 - x|$$

6. The relative error is defined as the ratio of the absolute error of the measurement to the actual measurement. If you want to know how to compute the relative error, given the actual value of a quantity, *x*, and the measured value of a quantity, x_0, then the relative error value, *r*, is calculated as:

$$r = \frac{|x_0 - x|}{x} = \frac{\Delta x}{x}$$

7. The modulo operation (often designated with the operator %) is defined as the remainder obtained after two operands are divided (e.g. 11 % 2 = 1, 29 % 3 = 2).

8. Images used are owned by either the original author, the University of Central Florida, or used under appropriate licenses (such as various Creative Commons licenses).

So !!!False

Filename: `sotrue`

"That's so true" ➞ lame, overused, common

"That's so not false" ➞ cool, hipster, innovative

Thomas has decided to apply this philosophy to programming. Now, when using "True" or "False" booleans in his programs, he adds a certain number of '!' symbols (commonly referred to as *not* symbols) before the "True" or "False" declaration. While Thomas thinks this is hilarious, Tyler is annoyed at having to decipher Thomas's code and has asked you to write a program to automatically convert each unnecessarily contrived boolean declaration into its simplified value.

The '!' symbol negates the boolean value. A negated "True" value becomes "False" and a negated "False" value becomes "True." The following table demonstrates this phenomenon:

!True ➞ False	!False ➞ True	!!!True ➞!!False ➞!True ➞ False

The Problem:

Given a string starting with some number of '!' symbols followed by "True" or "False", determine the simplified boolean value of the statement.

The Input:

The first and only line of input will contain a single string of between 4 and 100 characters (inclusive). This string will begin with some number of '`!`' symbols (possibly 0 of them), which will be immediately followed by either "`True`" or "`False`". No other characters will appear in the string.

The Output:

Output a single line consisting of either "`True`" or "`False`" representing the simplified boolean value of the given string.

Sample Input 1:

```
False
```

Sample Output 1:

```
False
```

Sample Input 2:

```
!!!False
```

Sample Output 2:

```
True
```

Placing Gazebos

Filename: `placing`

Thomas has collected the necessary bricks and is about to build a gazebo, mostly because he thinks the word is fun to say. He doesn't quite know how large he will make it, but he knows the gazebo will be a square. Before he builds his gazebo, he wants to do some planning and would like to know in how many places he can build it.

Thomas has modeled his backyard as an axis aligned n by n square grid with some free cells (represented by '`.`') and some occupied cells (represented by '`#`'). To ensure optimal gazebo placement, he has some self-imposed rules:

- The gazebo must be axis-aligned relative to the yard (that is, one of its sides must be parallel to the vertical or horizontal axis).
- No part of the gazebo can be built outside of the yard.
- If the gazebo covers some portion of a yard cell, then it must cover all of it.
- No part of the gazebo may overlap an occupied cell.

Before committing to building a gazebo with a specific size, Thomas wants to know for every possible size of gazebo he could build (fitting within his yard, of course) how many unique locations he could build a gazebo of that size.

The Problem:

Given a model of the yard, report the number of potential gazebo placements for each possible size of gazebo to be built.

The Input:

The first line will contain a single integer, n ($1 \leq n \leq 1{,}000$), representing the side length of the yard. The following n lines will each contain n characters (either '`.`' or '`#`'), which represents the yard's layout.

The Output:

Output n lines, each containing a single integer such that the k^{th} ($1 \leq k \leq n$) line represents the number of places in which a k x k gazebo can be placed.

Sample Input 1:

```
6
#...#.
.#....
...#..
......
.....#
..##..
```

Sample Output 1:

```
29
10
1
0
0
0
```

Sample Input 2:

```
3
...
..#
...
```

Sample Output 2:

```
8
2
0
```

Missing You from Letter Land

Filename: `letter`

Letter Land is a uniquely wonderful kingdom where words are formed by putting individual letters together. The Queen of Letter Land regularly assembles words of love and encouragement and sends them to her daughter, who rules across the sea in La La Land. The three words the queen likes to send to her daughter are as follows:

brightness: a reminder to always spread happiness and light throughout her kingdom

fairness: a reminder to rule with a firm yet fair hand

princess: a reminder of her royal worth and connection to her family

The queen sits upon a throne made entirely of lowercase English letters. Whenever the queen sends a word to her daughter, she assembles the word by removing letters from her throne. For instance, in the first sample input, the queen can construct two instances of the word *brightness*. In the second sample input, the queen can construct a single instance each of *brightness*, *fairness*, and *princess*. When an instance of a letter is used to form one of the words, it cannot be reused for another word.

The queen would like to send her daughter as many words as possible. Can you, the queen's loyal advisor, tell her how many words that is?

The Problem:

Given the frequencies of the 26 English letters that make up the queen's throne, determine the maximum total count of instances of the words *brightness*, *fairness*, and *princess* that the queen will be able to send her daughter.

The Input:

The first line of input contains an integer, n ($0 \leq n \leq 26$), representing the number of different types of letters present in the queen's throne. The following n lines of input each consist of a lowercase alphabetic character, c, and an integer, x ($1 \leq x \leq 10^5$), indicating that the character c appears x times in the queen's throne. If a letter does not appear in the input, it can be assumed that it does not appear in the queen's throne. Each letter will appear at most once in the input.

The Output:

Output a single integer: the maximum number of words the queen can send her daughter.

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

9 b 2 e 2 g 2 h 2 i 2 n 2 r 2 s 4 t 2	2

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

13 a 1 b 1 c 1 e 3 f 1 g 1 h 1 i 3 n 3 p 1 r 3 s 6 t 1	3

Sample Input 3: **Sample Output 3:**

1 x 1000	0

Elephant Espionage

Filename: `espionage`

After years of having peanuts withheld from them, the elephants have decided to rise up against their tyrannical zookeepers. To do this, they need to gather some intel about their foes and have deployed the Elephant Espionage Brigade. Three of the elite members of the brigade, Ellie, Elliot, and Ellanah, have been tasked with installing Hyper Soft-Pitched Transmitters (HSPTs).

They have identified two circular offices where the zookeepers often conspire against the elephants. They want to place an HSPT in both of the offices to gather intel on the zookeepers' schemes. The two HSPTs must relay information to each other, so it is important that they are as close to each other as possible.

Ellie, Elliot, and Ellanah are quite busy being expert spies and have asked for your help in determining the optimal placement of the HSPTs, given the x- and y-coordinates of each office's center point and their radii.

The Problem:

Given a map of the two circular offices, determine the optimal placement (minimum distance apart) of the HSPTs where an HSPT is within (or on) the circular boundary of each office.

The Input:

The first line of input contains a single integer, n ($1 \leq n \leq 10^5$), representing the number of different maps you have been asked to analyze. Each map will consist of two lines. Both lines will consist of three integers, x, y and r ($-10^4 \leq x, y \leq 10^4$; $1 \leq r \leq 10^4$), corresponding to the x- and y-coordinates of an office's center and its radius, respectively. It is guaranteed that the two offices will not intersect and that neither office will contain the other.

The Output:

Output n lines, each with four real numbers: x_1, y_1, x_2 and y_2, the x- and y-coordinates of where the HSPTs should be placed in the first and second circles, respectively. Your value for each coordinate will be considered correct if its absolute or relative error does not exceed 10^{-4}.

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

Sample Input 1	Sample Output 1
2 0 0 1 1 2 1 1 1 4 -3 -4 1	0.447214 0.894427 0.552786 1.105573 -1.498780 -2.123475 -2.375305 -3.219131

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

Sample Input 2	Sample Output 2
1 3 5 1 0 0 3	2.485504 4.142507 1.543487 2.572479

Penpen the Hero

Filename: `hero`

In a modest village terrorized by the great dragon, Tempest, hope for salvation was thin on the ground. Several of the village warriors had charged confidently into Tempest's lair, demanding that he stop the barrage of attacks, but they had always returned with scorched armor and a decided slump to their shoulders. Hysteria mounted, and a young shepherdess named Penpen could bear it no longer. That night, she stalked into Tempest's lair and waited until the dragon's great yellow eyes found her in the darkness.

Penpen held her head high and asked Tempest for a challenge, with the condition that Tempest would leave the village forever if she completed it. The great dragon grinned, baring his teeth, and posed Penpen the very same challenge that the warriors from her village had failed to complete. It was a challenge that could not be sliced by a sword or blocked by a shield. It could only be conquered by the mind.

Tempest proceeded to lay out an ordered sequence of gems from his hoard, each with a positive integer value. As he did so, he spoke forth the terms of the challenge for Penpen to hear.

1. Penpen shall select a contiguous subsequence of gems that is possibly empty.

2. Tempest shall select a contiguous subsequence of gems that is possibly empty and does not intersect with Penpen's chosen subsequence.

3. Penpen and Tempest calculate their individual scores as the average value of gems they selected in their respective subsequences. If a selected subsequence was empty, then its score is 0.

4. Let Penpen's score be p and Tempest's score be t. Penpen must aim to maximize the value of $p - t$, whereas Tempest must aim to minimize this value. Penpen must report the final value of $p - t$ to successfully complete the challenge.

Penpen stared at the gems for a moment, her eyes shining. Then, to Tempest's immense disquiet, she grinned. She knew what to do.

And thus, the legend of Penpen the Hero was born.

The Problem:

Given the value of each gem in Tempest's sequence, determine the difference between Penpen's score and Tempest's score, assuming they both play optimally.

The Input:

The first line of input contains an integer, n ($1 \leq n \leq 10^5$), representing the number of gems. The second line of input contains n integers, where the i^{th} integer, a_i ($1 \leq a_i \leq 10^9$), represents the value of the i^{th} gem.

The Output:

Output the difference between Penpen's score and Tempest's score. Your answer will be considered correct if its absolute or relative error does not exceed 10^{-3}.

Sample Input 1:

```
7
1 10 10 10 9 9 9
```

Sample Output 1:

```
8.5
```

Sample Input 2:

```
4
100 100 100 1
```

Sample Output 2:

```
99
```

Sample Input 3:

```
1
10
```

Sample Output 3:

```
10
```

Go with the Flow

Filename: `flow`

Marty is a world-renowned Greek plumber (despite his lackluster plumbing skills). Because he lacks knowledge of fluid mechanics, he often builds plumbing networks that are vulnerable to rupturing.

Plumbing networks consist of pipes that carry water and junctions that connect two or more pipes. Because of Marty's superstitions, he installs valves on every pipe, forcing the water to flow in one direction in each pipe. To keep the plumbing network stable, Marty thinks it is best for each junction to have an equal number of pipes leading into the junction as well as leaving the junction. He measures the stability of a single junction as the number of pipes leading into the junction minus the number of pipes leaving the junction.

Marty has asked you to determine the stability of each junction in a plumbing network so that he can assess the stability of the entire network.

The Problem:

Given a plumbing network, determine the stability of each junction.

The Input:

The first line of input will consist of two integers, n and m ($2 \leq n \leq 10^5$; $1 \leq m \leq 10^5$), where n represents the number of junctions and m represents the number of pipes, respectively. After this, m lines will follow each consisting of two integers, u and v ($1 \leq u \leq$ n; $1 \leq v \leq n$; $u \neq v$), representing a pipe leaving junction u and leading to junction v.

The Output:

Output n lines each with a single integer, where the i^{th} integer represents the stability of junction i.

Sample Input 1:

```
3 5
1 2
2 1
3 1
3 1
2 3
```

Sample Output 1:

```
2
-1
-1
```

Sample Input 2:

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

Sample Output 2:

```
0
0
0
0
```

Radioactive Decay

Filename: `decay`

Marie is a nuclear engineer designing her first nuclear reactor. To do this, she is using a system composed of two vats: one for the nuclear material and one for a cooling agent. Since the radioactive material will decay over time, she logs the amount of material remaining in a table and notices the following pattern: the current day's quantity of radioactive material is reduced by some divisor of the previous day's quantity. Formally, on successive days (other than the first, of course) if a_i is the amount of material on the i^{th} day, then there exists a positive integer, x, such that $a_{i-1} = a_i \cdot x$. Note that x may be different for each day.

Each day, she will empty the second vat of its old cooling agent and refill it with b_i units of cooling agent, such that b_i does not exceed the capacity of the second vat. Because of what seems like superstition amongst us non-nuclear engineers, Marie wants the amount of cooling agent and the amount of remaining radioactive material on each day to share no common divisors other than 1. That is, for all i, she wants the greatest common divisor of a_i and b_i to be 1.

Given her logs on the radioactive material, Marie has asked you to determine the number of ways she can fill the second vat across all days.

The Problem:

Given an array, a, representing Marie's log of radioactive material and an integer, c, representing the capacity of the vats, count the number of ways to construct an array of positive integers, b, where each element b_i in array b represents the amount of cooling agent added to the second vat on the i^{th} day.

The Input:

The first line of input will consist of 2 integers, n ($1 \leq n \leq 10^6$) and c ($1 \leq c \leq 10^{12}$), representing the number of days in Marie's log and the capacity of the vats (each having capacity c), respectively. The next line will consist of n integers where the i^{th} integer, a_i ($1 \leq a_i \leq c$), represents the amount of radioactive material remaining on the i^{th} day. It is guaranteed that for all i from 2 to n (inclusive), $a_i \cdot x = a_{i-1}$ where x is some positive integer. For each i, x is not necessarily the same.

The Output:

Output a single integer, representing the number of ways of creating the array b, where b_i represents the amount of cooling agent that Marie will put in the second vat. Each value b_i should be a positive integer up to c and share no common factors greater than 1 with a_i. Because the answer can be large, output it modulo $10^9 + 7$.

Sample Input 1:

```
3 30
30 30 1
```

Sample Output 1:

```
1920
```

Sample Input 2:

```
4 10
8 4 2 1
```

Sample Output 2:

```
1250
```

Sample Input 3:

```
2 1000000000000
1000000000000 1000000000000
```

Sample Output 3:

```
7840000
```

Building Gazebos

Filename: `building`

After thoroughly considering its placement, Thomas finally decided to build a gazebo in his backyard, not because he needed one, but simply because he thought the word "gazebo" sounded cool and trendy. To do this, he started with the foundation. Thomas purchased a pallet of 2x1 bricks (each of which may be rotated 90 degrees to become 1x2 bricks), and he planned to lay all of them (he does not want to waste a single one!) so that they formed a square with no holes or gaps between bricks. However, after many hours trying to create a square out of the bricks, he thinks it may not be possible.

Not to be deterred, Thomas has decided to buy more bricks! Since bricks are expensive and heavy to carry, he wants to buy as few extra bricks as needed to construct his square foundation. Please help Thomas determine how many more bricks he will need to buy to make a square foundation.

The Problem:

Given the number of 2x1 bricks Thomas already has, compute the number of additional bricks he needs to buy to construct a square foundation for his gazebo.

The Input:

The first and only line of input will contain a single integer, b ($1 \leq b \leq 10^9$), representing the number of bricks Thomas already bought that day.

The Output:

Output a line containing a single integer, x, representing the minimum number of additional bricks Thomas needs to buy to build the foundation of the gazebo.

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

Sample Input 2:	**Sample Output 2:**
5	3

Sample Input 3:	**Sample Output 3:**
12	6

1999, Bizarre Summer

Filename: `summer`

One golden morning in Morioh, Koichi decided to pay a surprise visit to his love, Yukako. Koichi painstakingly gift-wrapped an array of integers, put it in the basket of his bike, and pedaled excitedly to Yukako's house.

Yukako was so pleased with her present that she decided to frame it and put it on her wall. However, Yukako decided to first perform an operation on the array so that it would better match her wallpaper. The operation is as follows:

1. Let Yukako's array be array *a*.

2. Construct a new array of integers, *b*, where the i^{th} element of *b* is equal to the sum of the first *i* elements in *a*.

3. Set array *a* equal to array *b*.

For example, if Yukako's array is [1, -2, 3, -4], then the array will become [1, -1, 2, -2] after she performs the operation.

As his golden day turned to a golden night, Koichi pedaled home. He passed by a wishing well and tossed a coin inside, absently thinking, *I wish every day could be just like this one*.

Koichi woke up the next morning and found himself in a bizarre predicament: the events of the previous day seemed to be repeating themselves. Once again, Koichi pedaled to Yukako's house. Once again, Yukako performed the operation on her updated array. Once again, Koichi pedaled back to the wishing well, but this time, the wishing well was gone. Standing in its place was the fearsome Stand of Judgement, which spoke to him:

To reverse your wish, you must tell me how it ends. If the events of this day were to repeat indefinitely, tell me the sign of the last element in Yukako's array.

The Problem:

If Yukako were to continue performing the described operation on her array indefinitely, determine whether the last element would be positive, negative, or zero.

The Input:

The first line of input contains an integer, n ($1 \leq n \leq 10^5$), representing the number of elements in Yukako's array. The second line of input then contains n integers, where the i^{th} integer, a_i ($-10^9 \leq a_i \leq 10^9$), is the i^{th} element of Yukako's array.

The Output:

Output a single line. If the last element in Yukako's array would be positive, output "`Good morning, Morioh!`" If the last element in Yukako's array would be negative, output "`Good night, Morioh!`" If the last element in Yukako's array would be zero, output "`Diamond is unbreakable!`"

Sample Input 1:

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

Sample Output 1:

```
Good morning, Morioh!
```

Sample Input 2:

```
4
-1 1 -1 1
```

Sample Output 2:

```
Good night, Morioh!
```

Sample Input 3:

```
1
0
```

Sample Output 3:

```
Diamond is unbreakable!
```

Pseudonym Predicament

Filename: `pseudonym`

On the UCF Programming Team, we care a lot about efficiency, and saying people's names takes way too much time! Instead, people will be referred to by their first and last initials. However, this is problematic because of team members such as Tyler Marks and Thomas Meeks, who share initials. To avoid ambiguity, team members will also be assigned a number based on when they entered the room, starting with 0. So, if Tyler enters the room, he will be given the pseudonym TM0; when Thomas enters, he will be given the pseudonym TM1.

When someone leaves the room, the number in their pseudonym can then be reassigned if someone else with the same initials enters later. Numbers will be assigned such that the smallest non-negative integer not currently in use by anyone with the same initials is given to the next person entering. Note that multiple people can have the same number assigned to them if they have different initials.

This naming scheme has gotten quite confusing, so you have been asked to write a program that identifies who a certain pseudonym refers to as people enter and leave.

The Problem:

Given queries of pseudonyms, determine to whom the queried pseudonym refers as people enter and leave.

The Input:

The first line will consist of a single integer, q ($1 \leq q \leq 2 \times 10^5$), representing the number of queries. The following q lines will take one of the following forms:

- `Enter <FirstName> <LastName>`
  Representing that a person with the corresponding first and last name has entered the room. It is guaranteed that no one in the room will already have the same first and last name.
- `Leave <FirstName> <LastName>`
  Representing that a person with the corresponding first and last name has left the room. It is guaranteed that the person is currently in the room.
- `Who's <Pseudonym>`
  Querying the name of the person whom the pseudonym represents. It is possible that no one in the room currently has the queried pseudonym.

First and last names will each have at least one letter and at most ten letters, and each will start with an uppercase letter followed by zero or more lowercase letters. A pseudonym will begin with two uppercase letters, followed by at least one and at most six digits. It is guaranteed that there will be no leading zeros in the number (with the exception of the value 0 itself, which will be a single 0).

The Output:

For each "`Who's`" query, output a single line containing the first and last name of the person (as it appears in the input) that the pseudonym represents, separated by a single space. If no one currently in the room has that pseudonym, output "`404 PERSON NOT FOUND`" instead.

Sample Input 1:

```
7
Enter Tyler Marks
Enter Benjamin Prins
Enter Brian Barak
Who's TM1
Enter Thomas Meeks
Who's TM1
Who's BB0
```

Sample Output 1:

```
404 PERSON NOT FOUND
Thomas Meeks
Brian Barak
```

Sample Input 2:

```
10
Enter Tyler Marks
Enter Travis Meade
Who's TM1
Enter Thomas Meeks
Leave Tyler Marks
Leave Travis Meade
Enter Travis Meade
Enter Tyler Marks
Who's TM1
Who's TM2
```

Sample Output 2:

```
Travis Meade
Tyler Marks
Thomas Meeks
```

Fox on Clocks on Bricks and Blocks

Filename: `fox`

And here's a new trick, Mr. Knox....

Socks on chicks and chicks on fox.

Fox on clocks on bricks and blocks.

Bricks and blocks on Knox on box.

- From Dr. Seuss's *Fox in Socks*

There is something so irresistibly fun about a good tongue twister. As you can see, Mr. Fox's best tongue twisters have plenty of rhyming pairs.

After spending a wonderful day with Mr. Fox, Mr. Knox wanted to try coming up with some of his own tongue twisters. Mr. Knox borrowed Mr. Fox's rhyming word dictionary and opened it eagerly. Listed on the inside cover were the three Golden Rules of Rhyming:

1. Every word rhymes with itself.

2. If word *a* rhymes with word *b*, then word *b* rhymes with word *a*.

3. If word *a* rhymes with word *b*, and word *b* rhymes with word *c*, then word *a* rhymes with word *c*.

As he flipped further through the dictionary, Mr. Knox wrote down several of his favorite pairs of rhyming words. Then, he started to write his first tongue twister. Mr. Knox wrote and rhymed all day and all night. As the moon started to set, Mr. Knox gazed at last upon the fruits of his labor: the greatest tongue twister he had ever unsuccessfully attempted to read.

Mr. Knox wants to ensure that his tongue twister is good enough before sharing it with Mr. Fox. Here, Mr. Knox has encountered his first obstacle: determining the number of rhyming pairs in his tongue twister. Can you help him?

As an example, if "fox" rhymes with "clocks" and also "blocks", then the tongue twister "fox on clocks on bricks and blocks" has 4 rhyming pairs of words: "fox" with "clocks" as given, "fox" with "blocks" as given, "clocks" with "blocks" using rules 2 and 3, and "on" with "on" using rule 1.

The Problem:

Given a list of pairs of rhyming words and Mr. Knox's tongue twister, determine the number of pairs of words in the tongue twister that rhyme. Formally, determine the number of integer pairs (i, j) such that $i < j$ and the i^{th} and j^{th} words in the tongue twister rhyme.

The Input:

The first line of input contains an integer, n ($0 \leq n \leq 10^4$), representing the number of pairs of rhyming words in the dictionary. The next n lines of input each contain two single-space-separated strings, s and t ($s \neq t$), indicating that s rhymes with t. The strings s and t will each contain between 1 and 10 lowercase English letters (inclusive). Furthermore, the input will not contain duplicate pairs of rhyming words (so if the input includes the rhyming pair x y, then no following line will include x y and y x in the input).

The next line of input is a string containing only lowercase English words (each also inclusively between 1 and 10 letters in length) separated by single spaces, representing Mr. Knox's tongue twister. The tongue twister will contain at least 1 and no more than 10^5 characters.

The Output:

Output an integer: the number of pairs of words in the tongue twister that rhyme.

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

2 fox clocks fox blocks fox on clocks on bricks and blocks	4

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

3 fox blocks blocks docks docks socks fox on blocks on docks on socks	9

Sample Input 3: **Sample Output 3:**

3 three free cheese trees fleas cheese through three cheese trees three free fleas flew	6

Space Invaders

Filename: `invaders`

You are playing the classic video game Space Invaders! In this game, you move a laser cannon at the bottom of the screen left and right while attempting to shoot as many aliens as possible. You have three controls: move left one unit, move right one unit, and fire the cannon. After several hours of playing this game, you decide it is too easy and add a secondary challenge for yourself: finish with the cannon in the middle of the screen. Given a sequence of left and right moves you make and your starting position, you want to determine if you will successfully end in the middle of the screen. Note that the screen does not have bounds and the laser cannon can move left and right as much as desired.

The Problem:

Given a sequence of left and right moves and the starting position, determine if the cannon will end in the middle of the screen.

The Input:

The first line of input consists of two integers, n ($1 \leq n \leq 100$) and x ($-100 \leq x \leq 100$), representing the number of commands and the starting position of the laser cannon on the screen, respectively. The middle of the screen is represented by $x = 0$. The next line will contain a single string of n characters, composed of '`L`'s and '`R`'s. An '`L`' represents a move left, decreasing the position by one, and an '`R`' represents a move right, increasing the position by one.

The Output:

Output a single line. If you will successfully end in the middle of the screen, output "`Challenge completed!`" Otherwise, output "`Better luck next time!`"

Sample Input 1:

```
3 1
LLR
```

Sample Output 1:

```
Challenge completed!
```

Sample Input 2:

```
5 -1
RRRRR
```

Sample Output 2:

```
Better luck next time!
```