Spring Programming Contest 2018

Lower Division

March 31st, 2018

Do not open until contest starts
Instructions for Participants

- Contest URL: `https://contest.acmatfsu.org`
- You have 5 hours to answer questions.
- There are a total of 6 questions.
- You may submit solutions in the following languages:
  - C/C++ (1999, 2011)
  - Java 8
  - Python (2.7 or 3.x)
  - Perl 5.22.1
  - Javascript (Node.js v4.2.6)
- You are only allowed access to official language documentation and COP3014 reference material. You are restricted to:
  - C/C++: `http://www.cplusplus.com/reference/`
  - Java: `http://docs.oracle.com/javase/8/docs/api/`
  - Python 2.7: `https://docs.python.org/2/`
  - Python 3.x: `https://docs.python.org/3/`

- Perl: `http://perldoc.perl.org/`
- Javascript: `http://developer.mozilla.org/en-US/docs/Web/Javascript/Reference`
- COP3014 Reference:
  - `http://www.cs.fsu.edu/~vastola/cop3014/`
  - `http://www.cs.fsu.edu/~jayarama/prog1.php`

- You are also allowed one textbook or material no larger than 8.5" x 11" x 2" by volume.
- No other resources (e.g. Stack Overflow, Google, Wikipedia) are permitted. Using non-permitted materials will lead to disqualification.
- Teams are restricted to using one workstation (computer) each.
- Use of a cell phone to circumvent these restrictions will lead to disqualification. Use of cell phones in contest rooms is not permitted.
- The Clarifications tab on Domjudge may be used to submit questions pertaining to each problem. Do not use this feature to request troubleshooting/help.
- **Scoring:**
  - Teams are ranked according to score. A higher score is rewarded by answering more questions while acquiring fewer penalties.
  - The team that solves the greatest number of questions in the quickest time wins.
  - Teams which solve the same number of problems are ranked by least total time.
  - Teams may resubmit solutions as many times as needed, but incorrect submission attempts will result in time penalties (and thus a lower score.)
  - The scoreboard may be accessed during the first four hours of the contest. The scoreboard will freeze during the final hour, but teams can still move up/down in score. The scoreboard is frozen only as a suspense factor.
- **IMPORTANT**
  - All input is redirected via STDIN (cin).
  - You do not need to prompt the user for input, and you do not have to test for incorrect inputs.
  - All output must be formatted to specification in terms of capitalization and spacing. Please refer to the example output for each question.

1 Zombie Tower Defense

FSU has been overrun!

Zombies are attacking from all directions, and there seems to be no end in sight, there are billions!

You and other survivors quickly run to a nearby building on campus, where you see the number of entrances you will need to defend, and the type of defense available to you. Luckily, some buildings have strong defenses, (the engineering students saw this coming before anyone else), but sometimes all you have are humans to defend against the undead horde.

Your job is to determine how much of the available defense is needed to fend off the incoming zombies. Thankfully you are able to determine the number of entrances to the building, and how many zombies will be coming though each entrance.

1.1 Input

The first line of input will be an integer N, the number of entrances, followed by a character C, the type of defense you have. The next line will have N numbers, each separated by a single space, representing the number of zombies per entrance.

The total number of zombies that will be attacking the building, will always be a multiple of the given strength for the type of defense, so no need to worry about "half a Tesla tower", for example.

The following table shows the character C, along with the defense associated with it, and its overall strength (number of zombies a single unit of defense can fend off):

Character	Defense	Strength
H	Humans	10
B	Ballista	50
T	Tesla Tower	100

NOTE: You will only have one type of defense available.

1.2 Output

Your output should be a single integer, representing the amount of the given defense needed to fend off the zombie horde.

Looking at the sample below, the defense is H, meaning human, with 5 total entrances to defend. Since a Human can only defend 10 zombies at a given time, and there a total of 150 zombies, (number of zombies at each entrance added up), then 15 humans are needed to defend the building.

1.3 Sample Input/Output

Sample Input	Sample output
5 H 10 35 20 60 25	15

2 The Duck Question

Listen, you need to get your ducks in a row.

It's important, okay? Your ducks should always be in row, it's just how life is meant to be.

You forgot about your ducks, didn't you? It's probably why they are not in a row as of now. Look how sad they are... you monster.

It's okay, life is hard, and so I understand that sometimes ducks are in rows, and sometimes they aren't, but listen, it is **VERY IMPORTANT** that they be put in a row now. It is part of a 27 step guide to better life, see:

1. Put ducks in a row.
2. Look how happy they are!

   Yada, Yada, Yada...

27. Take over world with army of robot Ducks.

See? It's important that you put your ducks in a row.

To put your ducks in a row, I will tell you the number of ducks (because you probably forgot...so cruel), and then give you the ducks. Then, for them to successfully be considered to be in a row, they must be sorted from smallest to largest.

2.1 Input

The first line of input will be a single integer N, which is at most 100, representing the number of ducks. The next line of input will be N ducks, separated by a single space.

Thankfully, there are only three types of ducks, represented as lowercase characters; 's', 'm', and 'l', and the ordering of the three types of ducks are as follows:

$$s < m < l$$

2.2 Output

Your output should be the ducks, sorted from smallest to largest, with each duck separated by a single space. (Each duck should be printed as a single lower-case character).

2.3 Sample Input/Output

Sample Input	Sample output
7 l s s l s m m	s s s m m l l

3 Alien Translator

llo?He nCa uyo derstandun ?me eOn mentmo ease..pl.

Wow that thing is broken. Thankfully I have an old spare translator, though it's kinda on the *PIZZA*, and will replace some words with random *SNAILS*.

We need someone to fix our newer, but broken, *POTATO SALAD*, so that we can stop using this *RUDE* spare translator. Thankfully you do not seem that busy, do you think you are up to the *ONION*?

Seems like the broken *SALAD* is shifting all the letters in a word by a certain amount...and sometimes it will shift the letters to the left, or to the *BALLOON*.

What I'll do, is give you the number of *DUCKS* you need to shift back to normal, the amount to shift them by, and in what direction to do so.

From there, you just need to shift them back to normal, and print them out in a certain way. That sound *PEPPERONI*?

3.1 Input

The first line of input will be an integer N, representing the number of words, followed by another integer S, representing the amount to shift the words by, followed by a character C, representing the direction to shift the letters (either 'L' for left, or 'R' for right). Each word is no larger than 50 characters.

You can assume that the shift amount, S will not be larger than any of the words. Also, when a word is shifted, the letters at the start/end will move to the end/start, depending on the direction of the shift. For example, we will shift "lloHe" by 3 to the left, and show each shift one at a time:

Word	# of shift
loHel	1
oHell	2
Hello	3

Shifts to the right behave similarly.

3.2 Output

Your output should be each word shifted back to normal, followed by a single '.' Print the words in the order they appear in the input.

3.3 Sample Input/Output

Sample Input	Sample output

`3 3 L`
`lloHe`
`ngeStra`
`resCreatu`
	`Hello.Strange.Creatures.`

4 Functional Programming

Cornelius the graduate student is sick and tired of grading introductory algebra homework sheets, but instead of taking his frustration out on the students' grades or his liver, he decides to automate his way out of a job.

As an abstract mathematics graduate, his academic training has not prepared him with any useful skills for himself or society. Ever sly, he starts complaining on online Mongolian Throat Singing forums about how impossible it is to detect hand written formulae and convert it to ascii.

Thankfully, the Internet Tryhard Brigade^{TM} heard his cry for help and wrote him a classifier just to prove him wrong.

Cornelius now has files with strings of single-variate polynomials that the students give as solutions to homework problems.

To check if the polynomial is correct, he has to make sure that they evaluate to the correct value. But he doesn't want to write this program either, so he turns to the Internet for help.

Thankfully, it is you, brave citizen, that have decided to answer his call to action! Your job is to solve an expression, given a value for X, and print out the value.

4.1 Input

The first line of input will be a single integer N, representing the number of terms. The second line of input will be a single integer X, representing the value for X in the expression. The third line will be an mathematical expression with N terms.
A term for the expression can be the following:

- integer$X$^integer
- A single integer

Note that the first integer before X can be any valid integer. Each term is separated by an operation, either $-$ or $+$, with a single space before and after each operation. There is no whitespace in the terms themselves.

You can assume that the first term will always be positive, and that the exponents will ≥ 0 Also, there will only be one variable X.

4.2 Output

Your output should be a single integer, positive or negative, representing the value of the entire expression.

4.3 Sample Input/Output

Sample Input	Sample output
3 1 2x^5 − 4x^1 + 5	3

5 Infinity Mosaics

Thanos is not quite ready to get up from that really nice floaty chair. He figures he has a little bit of time before he has to make his appearance at Earth to gather up all the Infinity Stones. In his infinite(??) wisdom, he has contacted the Chitauri to forge his infinity gauntlet. Somehow, his requirements got lost in translation.

Thanos is now the proud owner of a magnificent gauntlet, except there is room for only one large circular infinity stone. No matter. Thanos decides to solve this problem by using his power to somehow place one stone inside another. However, not being super spatially aware (even Mad Titans have limits), he is "requesting" your help to figure out the order in which the stones should be placed in the gauntlet.

Please note that the Infinity Stones come only in 3 shapes - circles, squares and triangles. Your input would be the area of a circle, followed by the amount of shapes that should fit within the area, and your output should be the area of each shape that fits within the area of that circle. The ordering in which the shapes are placed inside each other is provided below (leftmost shape in this list is outermost shape):

circle → square → triangle → circle → square → triangle . . .

For consistency, your system must accommodate for up to 10 shapes. For your output please use double floating point precision with your calculations, and set the precision to a set fixed 5 decimal places.

5.1 Input

The input will be a single integer, representing the area of the outermost shape, followed by a single space, followed by another single integer N, representing N number of shapes.

N has the following constraint: $1 \leq N \leq 10$

5.2 Output

The output should be the shape, followed by a single space, followed the word "area", followed by a ':', followed by the area of the shape. This repeats for the number of shapes, with each shape on its own line. Please make sure all letters are lowercase.

For calculating area, assume the following: $\pi = 3.14159$

5.3 Sample Input/Output

Sample Input	Sample output
1000 5	circle area:1000.00000 square area:636.62031 triangle area:275.66468 circle area:166.66667 square area:106.10339

6 The Challenge of Three Labyrinths

You have somehow managed to wish away your professor to GoblinLand. Ooops. You now regret that very hasty decision. You can't get that extra credit anymore. As a service to humankind, you now need to navigate the Labyrinth of Doom to plead your case to Jareth the Goblin King. His Highness is in a good mood. He is willing to give you maps. Three of them, with clearly marked entry and exit coordinates. But, you now have to examine them and decide which Labyrinth to attempt. Piece of cake.

Wait! There is a second constraint:

"The only labyrinth with an exit that actually exists, is the labyrinth that takes an odd number turns to navigate, in the fewest number of turns. Every other labyrinth will dump you into the bog of eternal stench."

If a labyrinth takes an even number of turns, then it is not valid. Furthermore, if two or more labyrinths have the same number of fewest odd turns, then those labyrinths are valid. Select the correct labyrinth(s) to embark on this trial, and you might just be able to succeed, and dance magic dance your way to eternal glory (and extra credit).

6.1 Input

The first line contains a single number N, that represents the size of the labyrinth such that #rows = #columns = N. This is followed by the starting position, expressed as two numbers separated by a single space which gives the X and Y coordinates respectively.

The third line contains the exit coordinates, expressed in a similar fashion as the entry coordinates.

The three maps are then provided below in order, each preceded by an empty newline where '-' denotes an available path and 'w' denotes a wall which may not be crossed.

6.2 Output

The number or numbers of the maps which have the fewest amount of odd turns. The number for each map is the order in which the maps appear in. If there are multiple maps, then print the numbers separated by a single space.

6.3 Assumptions & Constraints

- You can only move in straight paths that you are facing.
- You must turn at least once to face the correct first path that you will take
- Every starting coordinate is a valid position on the map of each labyrinth
- Every exit coordinate is reachable
- There is always at least one labyrinth with the fewest odd turns to navigate
- All three labyrinths may be traversable if they all have the same amount of fewest odd turns
- The size of the labyrinth, n, is constrained such that $1 \leq n \leq 100$
- The coordinates are 0-indexed, and the first map is map 1.

6.4 Sample Input/Output

Sample Input	Sample output
`6`	`1 2`
`0 0`	
`0 5`	
`-W----`	
`-W----`	
`-W----`	
`-W----`	
`-W----`	
`------`	
`-W----`	
`-W----`	
`-W----`	
`-W----`	
`-WWWW-`	
`------`	
`-W----`	
`-W----`	
`-W-WWW`	
`-W----`	
`-WWWW-`	
`------`	