2021 CCSC Eastern Conference Programming Competition

October 23rd, 2021

Marymount University, Arlington, Virginia

This page is intentionally left blank.

Problem 1 — Zig-Zag

A "zig-zag sequence" is a sequence of integer values in which the values alternate between positive and negative. Any sequence in which there are two positive values or two negative values consecutively is not a zig-zag sequence. Also, any sequence with at least one zero value is not a zig-zag sequence.

In this program, each normal input line will contain a sequence of 1 or more integer values. For each normal input line, the program should print a single line of output as follows:

- If the line contained any zero values, the output line should be "`Zero value`"
- If the line did not contain any zero values, and is a zig-zag sequence, the output line should be "`Yes`"
- If the line did not contain any zero values, and is not a zig-zag sequence, the output line should be "`No`"

As a special case, if the input line consists of the text "`quit`", the program should exit immediately without producing any additional output.

Example input:

```
1 -2 3 -4
1 -2 3 4 -5
1 -2 0 -3
-4 5 -6
quit
```

Example output (corresponding to the input shown above):

```
Yes
No
Zero value
Yes
```

Problem 2 — Vowel Movement

We know that it's generally possible to understand a phrase if the vowels (a, e, i, o, and u) are removed. For example:

The rain in Spain falls mainly on the plain

becomes

Th rn n Spn flls mnly n th pln

But, what if instead of removing the vowels, we *moved* them? Specifically, collect all of the vowels in the phrase, sort them alphabetically, and substitute the sorted vowels for the originals? Then,

The rain in Spain falls mainly on the plain

would become

Tha raan an Spaen fells miinly in thi plion

Write a program which reads lines of text from standard input, collects just the vowels (a, e, i, o, and u, both upper and lower case), converts them to lower case, sorts them alphabetically, substitutes them for the vowels in the original line. The program should convert cases as needed — e.g., if the original line had an upper-case vowel in a specific position, it should still have an upper-case vowel in that position after the substitution. As a special case, if a line containing just "`quit`" is read, the program should exit without producing additional output.

Example input:

```
The rain in Spain falls mainly on the plain
A strong smell of turpentine prevails throughout
O Canada! Our home and native land!
quit
```

Example output (corresponding to the input shown above):

```
Tha raan an Spaen fells miinly in thi plion
A strang smell ef terpentini provools throughuut
A Canada! Aar heme ind notovo lund!
```

Problem 3 — Roman Numerals

In Roman numerals, the letters M, D, C, L, X, V, and I represent the integers 1000, 500, 100, 50, 10, 5, and 1, respectively.

Numbers are normally specified using *additive* notation, with larger numerals appearing before smaller numerals. For example, the integer 121 is specified as CXXI, meaning $100 + 10 + 10 + 1$.

As a special case, if a smaller numeral appears before a larger one, it is a *subtractive* notation, which the value of the smaller numeral being subtracted from the larger. For example, IV means $5 - 1 = 4$. You may assume that subtractive notation only ever affects one adjacent pair of numerals at a time, never groups of 3 or more. For example, the integer 85 would be written as LXXXV, $50+10+10+10+5$. It would *not* be written as VXC ($100 - 10 - 5$.) Note, however, that it is possible for there to be multiple subtractive pairs. For example, the integer 1904 could be written as MCMIV ($1000 + (1000 - 100) + (5 - 1)$.

Each normal input line will contain one integer specified in Roman numerals. For each such input line, your program should print a single line of output with the value of the input integer specified in base-10. As a special case, if an input line consists of the text "`quit`", the program should immediately exit without producing additional output.

Example input:

```
MCMIV
MCMLXXIV
DCIV
DCXVII
XXVIII
XXIX
quit
```

Example output (corresponding to the input shown above):

```
1904
1974
604
617
28
29
```

This page is intentionally left blank.

Problem 4 — Where Are We?

Residents of Flatland, who live in an infinite 2-D coordinate plane, enjoy road trips where they travel at a constant speed along a path consisting of connected line segments. Because their speed is constant (acceleration and deceleration are instantaneous in Flatland), and because they know the exact length of their planned route, they can precisely locate where they are as long as they know what fraction of the journey they've completed. For example, Shreya Circle is driving with her daughter Rachana along the path described by the points $(0, 0), (4, 3), (-4, 9)$. She knows the trip will take two hours. The following exchange occurs:

Rachana: Ma, where are we?
Shreya: It's been 90 minutes since we started, so our journey is 0.75 complete.
That means we're at $(-1, 6.75)$.

The input to the program is a series of two-line records. The first line in a record contains a sequence of at least two x/y coordinate pairs describing a path in the 2-D coordinate plane. Each pair is simply two decimal values, and each value will be separated by at least one horizontal whitespace character. The second line contains a single fraction value p. The program should determine the x/y coordinates of the point that is found by starting from the beginning of the path and then traversing a distance that is p times the total length of the path. For each record, the output should be a line of the form

`x=`*xcoord*`, y=`*ycoord*

where *xcoord* and *ycoord* are the x and y coordinates of the computed point, formatted with exactly 2 digits after the decimal point. There should be a single space after the comma.

There are two special cases. If the fraction p is less than 0 or is greater than 1, an output line with the text "`Invalid fraction`" should be printed. The other special case is that if the first line of a record consists of just "`quit`", then the program should not attempt to read the record's second line, and should exit immediately without producing additional output.

Example input:

```
0.0 0.0 4.0 3.0 -4.0 9.0
0.75
0.0 0.0 4.0 3.0 -4.0 9.0
0.5
0.0 0.0 4.0 3.0 -4.0 9.0
.833
0.0 0.0 4.0 3.0 -4.0 9.0
1.0
```

```
0.0 0.0 2.0 3.0 6.0 4.5 2.5 8.0 -3.0 9.33
0.71
0.1 1.2 2.3 3.4
1.1
quit
```

Example output (corresponding to the input shown above):

```
x=-1.00, y=6.75
x=2.00, y=4.50
x=-2.00, y=7.50
x=-4.00, y=9.00
x=2.21, y=8.07
Invalid fraction
```

Problem 5 — Roto League Scoring

In fantasy sports leagues (sometimes called Roto Leagues), owners draft professional players and then earn points, depending on the combined performance of their players, as compared to other owner's players. Typically, performance is broken into categories, with the best team in each category earning N points for the category, the next best team earning $N - 1$ points, and so on — assuming there are N owners in the league.

For example, suppose Albert, Fox and Gabes are in a three-owner baseball Roto League which uses only two categories, home runs and wins. Suppose the totals for each category are as shown in the following table:

Owner	HRs	Ws
Albert	13	10
Fox	15	8
Gabes	20	15

In this case, Gabes leads the league with 6 points (3 for HR and 3 for W), with Albert and Fox tying for second with 3 points each (Albert gets 1 for HR and 2 for W, Fox gets 2 for HR and 1 for W).

Scoring is more interesting if there are ties in a category. In that case the teams who are tied evenly split the available points for the positions that are tied. For example, if the category totals for the league described above were

Owner	HRs	Ws
Albert	13	8
Fox	13	8
Gabes	13	15

then each owner would earn $2 = (3 + 2 + 1)/3$ points for HR, Gabes would earn 3 points for W with Albert and Fox each earning $1.5 = (2 + 1)/2$.

Your job is to write a program that calculates the total points for each owner in a league. You can assume that all input is integral and that in every category it is best to have the highest total as in the HR and W categories above.

Program Input: The first line of input contains an integer greater than 0 and less than 100 that indicates how many leagues are represented in the rest of the input. This is followed by information about each league in turn.

For each league, the first line of input contains two numbers, N the number of owners and C the number of Categories in the league. $2 \leq N \leq 20$; $1 \leq C \leq 20$.

The following N lines each contain C integers, the scores for the Nth owner in each category, 1 through C.

Program Output: For each league output a line indicating the league number, followed by N lines with the total points of the N owners, showing one decimal point of accuracy.

Example input:

```
2
3 2
13 10
15 8
20 15
3 2
13 8
13 8
13 15
```

Example output (corresponding to the input shown above):

```
League 1
3.0
3.0
6.0
League 2
3.5
3.5
5.0
```

Problem 6 — The Fences Are Always Greener

Bunny Bob was born in the wild, but he's always wondered what it would be like to live in fenced-in yard in the suburbs. Wouldn't it be great to trade in the wide-open freedom for a square plot of land in a crowded neighborhood? Well, BB thinks so. He heard squirrels are neat, and really, he's past his prime and doesn't need to be hopping around all day. That said, he asks you to demonstrate what a fence might look like, so he can start to plan for this change. He has a good imagination, so let's print out some ASCII art fences for him.Fence problem.

Input: The first line of input represents the number of test cases to follow. Each test case contains an integer F between 0 and 1000 followed by two characters—the posts and the connectors.

Example input:

```
3
3 | =
10 ^ -
1 ! ~
```

Output Print a fence with F posts, with each post connected with a connector. Why does this satisfy Bunny Bob? Don't think about it too much.

Example output (corresponding to the input shown above):

```
|=|=|
^-^-^-^-^-^-^-^-^-^
!
```

Problem 7 — Printout with Butterfly Wings

AKA Palindromes and Obnoxious Text on a Page

Hey you, who doesn't love palindromes? I know I do, and I'm just a bunch of text on a page. Hey kid, do me a favor? Text on pages ain't great at code, so I need you to do this page a solid, ya? I want a mirrored line of text that counts down toward the middle and then back up. Don't it look like butterfly wings? Like if they the butterfly was a 2D string of numbers? No? Well, like, this gibberish is just here to confuse you. Anyway, I'll give you a bunch of numbers and you'll print out the wings, ya? Thanks a bunch, kid! What's in it for you? Well sorry, I got nothing. Despite all my rage, I'm still just text on a page. (This is a 90s reference. Ask your parents).

Input: The first line of input represents N, the number of test cases to follow. The following N lines contain a single integer corresponding to the number of digits in that line.

Example input:

```
3
6
7
8
```

Output: Each line will print a numbered palindrome. The center of the of each palindrome will be 1 for odd input and "1 1" for even, counting outwards.

Example output (corresponding to the input shown above):

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

Problem 8 — Infinity Words

What did it cost? $\theta(Everything)$

After years of buildup and dozens of lesser algorithms, you've nearly reached the Computer Science Endgame. Thanosort is the biggest and baddest new sort, straight out of the reaches of space. Sick of sorting algorithms taking up all your system resources, we've given up on traditional sorting methods. To start, let's just snap our fingers and get rid of half the data or so. We'll just call whatever's left sorted.

Uh oh, there's a problem that we didn't consider. For some reason, the snapped data returns anyway… and it must be some kind of contrived time travel situation because it all comes back in reverse.

Input: The first line of input represents N, the number of test cases to follow. The following N lines contain a single integer, S, followed by some text, T. The text will just be a string of some length up to 1000. S will be our Snap Factor. Every S'th character starting with the first in each string will be snapped out of existence. N<=100. S<100.

Example input:

```
3
2 dHlerlolWo
3 HelloWorld
5 HelloWorldHelloWorldHelloWorld
```

Output Print the Thanosorted string for each line.

Example output (corresponding to the input shown above):

```
HelloWorld
eloWrldolH
elloorldelloorldelloorldWHWHWH
```

Problem 9 — Bored of the Rings?

Dungeons and drag on and on, amirite?

Frodot and his fellows get together to plan a trip to the trademark-free land of Mordur. They are on a journey to save the world or something, but what's a fellowship without a treasure hunt? His Uncle Bilbot has given them a series of maps to help them safely travel to various dungeons along the way. These dungeons are full or dangerous traps, so they can only travel forward through them. Only dungeons that loop back to the entrance will let them safely escape with their loot. Help these adventurers out by letting them know which dungeons are safe to explore.

The dungeons are made up of a series of rooms connected via short hallways full of traps. The adventurers must always move forward into adjacent rooms and they can't go back through these tunnels, but some rooms have multiple ways in and out. In other words, it is safe to revisit a room if a hallway connects back to an already visited room.

Input: The first line of input represents N, the number of dungeon maps. The following N lines contain a series of integers. The first in every line is the number of hallways there are in that dungeon and each succeeding pair of digits represents a room and a connecting room. (e.g. 1 0 1 means There is one hallway out of Room 0 to Room 1). The starting room is always 0.

Example input:

```
4
4 0 1 1 2 2 3 3 0
3 0 1 1 2 2 3
5 0 1 0 2 0 3 1 2 2 3
7 0 1 1 2 2 3 3 4 4 2 2 5 5 0
```

Output: If there is a safe passage through all the rooms, print "Yes". Otherwise print "No".

Example output (corresponding to the input shown above):

```
Yes
No
No
Yes
```

Problem 10 — Santa Convention

At the annual Santa Claus convention, all the Santas around the world gather. The descendants of the real, original Santa Claus also attend the convention, but they have trouble identifying each other among all the fake Santas. This year, they devised a secret code for their name tags so that they can identify their relatives out of the crowd.

A descendant of the real Santa will use exactly one occurrence of either "Santa" or "Claus" on their name tag, but it will be misspelled by adding a single letter. If either "Santa" and "Claus" appears spelled correctly, or if there are two or more instances of either "Santa" or "Claus" correctly spelled *or* with a letter added, then it is not a descendant of the real Santa.

The first input line specifies the number of name tags, and each subsequent line specifies the words of one name tag, separated by spaces. There is no space at the end of any line. For each name tag, output "`yes`" or "`no`" to indicate whether or not the name tag belongs to a descendant of the real Santa. Ignore capitalization and punctuation.

Example input:

```
10
Santa Claus
Santta Fred
St. Nicholas Clause
Santa Claaus
Santa Fred Kringle
Mary Ann Clauss
Terry Gary Clarus
Ssanta Clauss
Slanta Cclauss
Magdalena Tsa'nta Soanta
```

Example output (corresponding to the input shown above):

```
no
yes
yes
no
no
yes
yes
no
yes
no
```