TeamsCode Fall 2018 MIHS Programming Contest

Problem Set

Rules:

1. Each question is worth 50 points. With each incorrect submission, the value of that question decreases by 5 points. You do not need to solve the problems in order.

2. The internet may not be accessed during the contest coding phase. Only one computer per team can be out during this time, meaning phones also need to be put away.

3. The teams with the highest score at the end of the contest will receive awards. In the case of a tie, the team that made their final successful submission first will be the winner.

Problems:

1. Elephant
Input File: elephant.txt

For *Elephant*, you must print out the elephant shown below. There is no input in this problem, but your elephant must match **exactly** with the one shown, down to the last dots and spacing.

Input:

None.

Output:

Output the elephant exactly as shown below.

Example Output:

```
          __     __
         /  \~~~/  \
   ,----(     ..    )
  /      \__     __/
 /|         (\  |(
^ \   /___\  /\ |
  |__|    |__|  -
```

2. Merchant
Input File: merchant.txt

You are a merchant, planning on selling some very valuable goods, but you must sell your goods at the local price and pay the local tax. Given your amount of goods, the price at which you can sell them, and the tax rate, determine your revenue.

Input:

The first line contains an integer N. The following N lines contain 2 space-separated integers and a double representing a, p, and t, respectively. a represents the amount of goods you have. p represents the price per good that you can sell at. t represents the percentage of revenue that you must pay as tax.

Output:

Output your revenue in each case. Your answer will be a double.

Example Input:

```
3
1000 3 0.25
3 200 0.1
0 100 0.2
```

Example Output:

```
2250.0
540.0
0.0
```

3. Secret Message
Input File: message.txt

Someone has left you a message by altering the letters of the titles of your books. Your job in *Secret Message* is to determine the secret message by comparing the letters of the altered message and the unaltered title.

Input:

The first line contains an integer N. There are N following cases. Each case will first consist of a line of unaltered text, then a line of altered text.

Output:

Print N lines of strings. Each string should consist of the changed characters of the case. If the altered line is the same as the unaltered line, print “`No change`”.

Example Input:

```
4
The Great Expectations
The GrHat Exeeclltioos
Les Miserables
Les Worldables
The Importance of Being Earnest
The Importance of Being Earnest
I like CAKE
Inlio  cake
```

Example Output:

```
Hello
World
No change
no cake
```

4. Pyramid
Input File: pyramid.txt

You can build a very basic pyramid, by putting layers of 1 unit thickness squares on top of each other. For example, a pyramid of 3 levels, can be made by putting a 1 x 1 square on top of a 2 x 2 square on top of 3 x 3 square. Your job is to determine the volume of the pyramid of a given level. The example pyramid's volume is 14.

Input:

The first line contains an integer N. There are N following line, each consisting of an integer, defining the levels of the pyramid.

Output:

Output the volume of the pyramid of the given levels. Your output should be an integer.

Example Input:

```
3
1
4
100
```

Example Output:

```
1
30
338350
```

5. Odds and Evens
Input File: oae.txt

Your job in Odds and Evens is to sort integers into two sorted lists, one of odd numbers, the other even.

Input:

The first line contains an integer N. The following N lines contain a string of integers separated by a space.

Output:

For each line of input, output the integers sorted numerically, except with all the odd numbers before even numbers.

Example Input:

```
4
1 2 3 4 5 6 7 8 9 10
9 817 283 74
12 938 1 2 333 72
10 9 8 7 6 5 4 3 2 1
```

Example Output:

```
1 3 5 7 9 2 4 6 8 10
9 283 817 74
1 333 2 12 72 938
1 3 5 7 9 2 4 6 8 10
```

6. Collatz Conjecture

Input File: collatz.txt

The Collatz Conjecture is a theory that states that any positive number, when inputted into the rules below, will reduce to 1.

If number is odd, multiply by 3 add 1.
If number is even, divide by 2.

An exception has yet to be found. Your job is to determine how many steps it will take for a number to reduce to 1.

Input:

The first line contains an integer N. The following N lines contain a single integer. That integer represents the starting number for the function.

Output:

Output the number of steps, following the protocol above, are necessary to reduce the inputted number to 1.

Example Input:

```
4
1
4
5
123
```

Example Output:

```
0
2
5
46
```

7. Word Choice

Input File: word.txt

You have a very sloppy friend, who always switches words when writing sentences. Your job in Word Choice is to fix your friend's sentences.

Input:

The first line contains an integer N. The following N lines contain strings defining the switches you must make. They'll be formatted like this: `wordA wordB`. Whenever `wordA`, regardless of capitalization or attached punctuation, appears in an inputted sentence, you should replace it with `wordB`, and conversely. No word will ever appear in two switches. After the switches, the next line contains another integer M. The following M lines contain the sentences that you need to manipulate.

Output:

Output the sentences with any instance of a word in the switches, switched with the corresponding word.

Example Input:

```
5
affect effect
except accept
alternately alternatively
bare bear
censor sensor
4
The colors had a great affect on him, accept blue.
We alternatively took turns fighting the bare, bear handed.
China wants to sensor the affect of a bare in the cities.
The college will except me into their college, or alternately will reject me.
```

Example Output:

```
The colors had a great effect on him, except blue.
We alternately took turns fighting the bear, bare handed.
China wants to censor the effect of a bear in the cities.
The college will accept me into their college, or alternatively will reject me.
```

8. Fencing
Input File: fence.txt

You have been commissioned to build a fence around a community of houses. The three stipulations are: the fence should be small as possible, it should to be rectangular, and it should enclose all houses.

Input:

The first line contains an integer N. There are N following cases. Each case will first consist of a line with 2 integers; the first defines the number of rows of the inputted map, and the second defines number of columns. The inputted map will consist of the `.` and `s` characters: the . represents an empty space. The x represents a house. Each character map will have at least 1, x.

Output:

For each case, output the number of rows and columns of the smallest rectangle that encloses all of the houses. The number of rows should be first, then the number of columns, separated by a space.

Example Input:

```
3
4 6
......
.x.x..
....x.
..xx..
8 8
........
...xx...
..x...x.
.....x..
.x.xx...
..xx.x..
...x....
........
1 1
x
```

Example Output:

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

9. Unit Conversion

Input File: conversion.txt

In science class, you must convert quantities using known ratios. Your job is to use the given ratios to convert a value of a certain unit to a different unit. It may be necessary to flip the given mole ratios.

Input:

The first line contains an integer N. There are N following cases. Each case will first consist of a number M; that number will indicate the number of lines of ratios. Each inputted ratio will be in the format of `X unitA equals Y unitB`. This means the ratio between `unitA` and `unitB` is `X / Y`. The final line in every case will be the given value and the units which it needs to be converted into, in the form of `X unitA to unitB`.

Output:

Output the value in the new units. Your answers should be rounded to the nearest integer and in integer form.

Example Input:

```
2
2
1 dime equals 10 cent
100 cent equals 1 dollar
20 dime to dollar
3
300 florp equals 1 zink
1 dinc equals 20 zink
1 dinc equals 5000 rand
1000 rand to zink
```

Example Output:

```
2
4
```

10. Area Under the Curves

Input File: area.txt

A very common activity in math classes is finding the area under a curve; your task for this challenge is very similar. Given two straight lines which intersect in Quadrant I (positive x, positive y) and form a quadrilateral with **both** the x-and y-axis, you must find the confined area.

Input:

The first line contains an integer N. For every N, you will receive 2 lines, each defining one mathematical line. The lines will be formatted like this: `A x + B y = C`, where `A`, `B`, and `C` are double coefficients.

Output:

You should output the areas confined by the given lines. The output should be in doubles, rounded to the nearest thousandth. The value 1 should still be printed as `1.000`.

Example Input:

```
3
0 x + 1 y = 1.0
1.0 x + 0 y = 1.0
1.0 x - 2 y = -4.0
2.0 x - 1 y = 2.0
4.5 x - 1 y = 10
2 x - 0.5 y = -1
```

Example Output:

```
1.000
4.333
132.889
```

11. Escape Room
Input File: escape.txt

An evil genius has locked you in a maze with only one exit. You need to escape the maze as quickly as possible. Thankfully you have the map of the maze, but you notice strange markings with each room; you discover that each room has special effects. Nonetheless, you must devise a the quickest path out.

Input:

The first line contains an integer, N; There are N cases. For each case, the first line will consist of 2 integers, which indicate the number of rows and columns of the map, respectively. The map will consist of the characters: “X”, “S”, “A”, “F”, “G”, “L”, “N”, and “O”. The characters represent specific effects that each room has:

“S”: The room marked S is the starting room.

“X”: The room marked X is the ending room.

“A”: In rooms marked A, you can enter through any direction and leave through any direction. It takes 10 seconds to pass through room A.

“F”: It takes 20 seconds to pass through rooms marked F. Otherwise it functions like an A room.

“G”: You must move left when you walk through rooms marked G. A room marked G will never appear on the left-most column of the map. You may enter through any side, and it takes 10 seconds to pass through these rooms.

“L”: You must leave the room in the direction opposite from the one you entered from. (i.e. if you enter a L room from the right side, you must leave from the left side) It takes 0 seconds to pass through these rooms.

“N” : Rooms marked N are unpassable.

“O”: Rooms marked O function like the whatever room is to the left of them. A room marked O will never appear on the left-most column of the map or to the right of the S or X rooms.

Output:

Output the shortest amount of time it takes to get from room S to room X. Your answer should be in integer form.

Example Input:

```
2
2 8
SLOOOOOX
NNNNNNNN
5 8
AFGNOFFA
FSFOOLGG
ALNNNGGG
AGONALLA
ALOXANNL
```

Example Output:

```
0
30
```

12. Flattened Cube
Input File: cube.txt

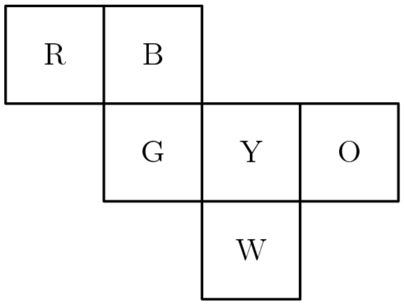
It is possible to make a 3D cube by folding a 2D diagram, such as the one below. Your job for cube is to determine which side is opposite of a given side.

Input:

The first line contains an integer N. There are N following cases. Each case will first consist of 4 lines defining the cube flattened out. These lines will have a character representing the character on the side of the cube or a `.` representing nothing. The format of a cube is shown below:

```
RB..
.GYO
..W.
....
```

is equal to



After the cube is inputted, a single character will be inputted. The character will correspond to one of the sides.

Output:

Your output should be the character on the opposite side of the side corresponding to the inputted character.

Example Input:

```
3
RB..
.GYO
..W.
```

```
....
B
.A..
EBF.
.C..
.D..
D
B...
ADCE
..X.
....
X
```

Example Output:

```
W
B
B
```

13. Rubik’s Cube
Input File: rubiks.txt

It has been proven that if you take a solved rubik’s cube and apply some formula enough times, it will return back to the solved formation. For example if you turn the right side 90 degrees clockwise and turn the top side 90 degrees clockwise, you will resolve the cube after 105 iterations or 210 turns. Your job is given a formula to determine the smallest number of iterations to return the cube back to normal.

Input:

The first line contains an integer, N. The following N lines will be a string, each defining a formula. The formulas will be defined using R, R’, L, L’, F, F’, U, and U’, with space between each character. These characters represent a turn. R means a 90 degrees rotation clockwise of the right side; R’ means a 90 degrees rotation counterclockwise of the right side. L means a 90 degrees rotation clockwise of the left side; L’ means a 90 degrees rotation counterclockwise of the left side. The F and U follow the same rules rules of turning clockwise/counterclockwise, except with respect to the Front side and Up (top) side, respectively. Clockwise and counterclockwise are defined by the side in question.

Output:

You should output the amount of iterations of the given formula it takes to return a Rubik’s Cube back to the solved state.

Example Input:

```
4
R U
R U R’ U’
F U F’ R L U
R R R R U
```

Example Output:

```
105
6
105
4
```

14. Target

Input File: target.txt

Target is a fun mental math game. It entails using 3 given numbers, each between 1 and 6, to try and make a non-prime number between 1 and 144. You can add, subtract, multiply, divide, factorial, and use parentheses. The closest number to the target number wins; if you manage to make the given number, you reply “target”. Your job in Target is to play target.

Input:

The first line contains an integer, N. There will be N following cases. Each case consists of 2 lines: one giving the 3 numbers, which you use, the other giving the non-prime number you are trying to make with the 3 numbers.

Output:

For each case, you must output the number which can be made from the 3 given numbers and closest possible to the target number. If it is possible to make the target number exactly, print “Target”.

Example Input:

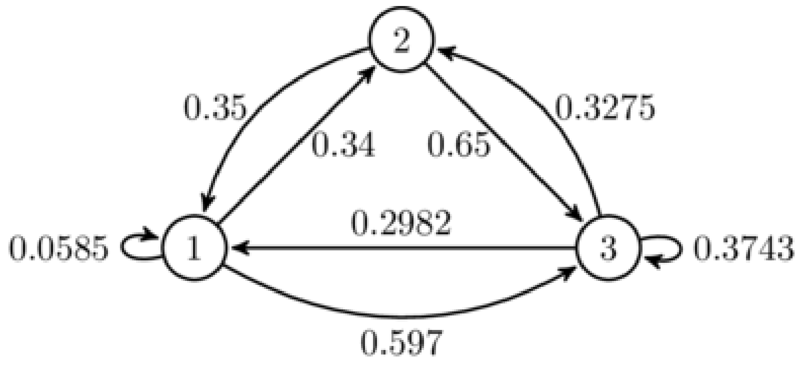
```
3
1 1 1
144
2 6 4
24
6 5 2
122
```

Example Output:

```
6
Target
123
```

15. Markov Chain
Input File: markov.txt

In mathematics, there is a concept called Markov Chain. A markov chain is a set of points, each of which has a numerical value, and with every "step", each point distributes its value based on defined instructions. Below is an example of a markov chain:



In the example above, for every step, point 1 gives 34% of its value to point 2, and point 2 gives 35% of its value to point 1. Point 1 also gives 5.85% of its value to itself (i.e. it keeps 5.85% of its value) and gives 59.7% of its value to point 3.

Eventually, after enough steps, the values of the points are in equilibrium, meaning they don't change. Your job is to determine the equilibrium value of each point of a given markov chain.

Input:

The first line contains an integer, N; There are N cases. For each case, the first line will be an integer, M. M lines of text will follow. These lines are defining the markov chain. They will be structure like this "A P - B", which means point A gives P percent of its value to point B. For any undefined links between points, you may assume the value is 0. The total percentage distributed from a point will never rise above or below 1.

Output:

You should output the double values of each point in equilibrium, rounded to the nearest hundredth. It should be sorted, by alphabetical order, meaning point A's value should be displayed before point B's value.

Example Input:

```
3
8
```

```
A 0.0585 - A
A 0.3445 - B
A 0.597 - C
B 0.35 - A
B 0.65 - C
C 0.2982 - A
C 0.3275 - B
C 0.3743 - C
2
S 1 - X
X 1 - X
9
B 0 - B
A 0.25 - C
C 0.50 - B
C 0 - C
C 0.50 - A
A 0.50 - A
A 0.25 - B
B 0.33 - A
B 0.67 - C
```

Example Output:

```
0.25
0.25
0.50
0.00
1.00
0.46
0.26
0.29
```