TeamsCode Spring 2019 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.

The teams with the highest scores 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. Owl
2. French Pronunciation
3. Note Length
4. Reverse Words
5. Right or Not?
6. Secret Letter Strings
7. Box Pyramid
8. ASCII Art
9. Wo-Word-rd
10. Shoe Scramble
11. Tunnels
12. Matching Up
13. Crumbling Structure
14. Balance the Chemical Equation
15. Flooding an Anthill

1. Owl
Input File: owl.txt

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

Input:

None.

Output:

Output the owl exactly as shown below.

Example Output:

```
 /\______/\
| (o) __(o)|
|     \/   |
/""""""""""\
............
\............/
/___  __  _\
    \\   \\
```

2. French Pronunciation
Input File: french.txt

The French Language can be very confusing to pronounce to non-native speakers. The short rule is to not pronounce the last letter in a word unless it is c, l, f, or r. Also, h's are not pronounced if they are at the beginning of the word.

In *French Pronunciation*, you'll be helping the language by removing all these silents letters.

Input:

The first line contains an integer, N. The following N lines consist of a string of words. The words can be in uppercase or lowercase. There will be no punctuation.

Output:

Output every word with the last letter removed unless it is c, l, f, or r. Also, remove the first letter if it is an h. One letter words should remain unchanged. Two letter words that start with h should keep the h at the beginning.

Example Input:

```
5
Je me appelle Chris
Je espere que tu as du bon temps
The question writers do not actually know French
But we try my best with the help of French Speakers
Clfr rclf frcl lfrc hhhh
```

Example Output:

```
J m appell Chri
J esper qu t a d bo temp
Th questio writer d no actuall kno Frenc
Bu w tr m bes wit th el of Frenc Speaker
Clfr rclf frcl lfrc hh
```

3. Note Length
Input File: notelength.txt

In music, the most common note is a quarter note; a note twice as long is a half note, and a whole note is four times as long as the quarter note. Your job in *Note Length* is to count the amount of time in terms of the length of a quarter note when given a string of whole notes, half notes, and quarter notes.

Input:

The first line contains an integer N. The following N lines are strings of the letters w, h, and q. The w represents a whole note; h, a half note; q, a quarter note.

Output:

Output the length of time of the combination of notes in terms of quarter notes.

Example Input:

```
4
qqqwhhqhqhqh
hhhhhhhhhhh
q
whhwhqwhqhwwhqhwhwww
```

Example Output:

```
20
22
1
55
```

4. Reverse Words
Input File: reverse.txt

In *Reverse Words*, reverse the characters in each word, but leave the order of the words alone.

Input:

The first line contains an integer N. The following N lines are strings with several words in each. There will always be at least 2 words with a space separating them. There will be no punctuation besides the apostrophes.

Output:

Output the strings with each word reversed, but with the order of the words intact.

Example Input:

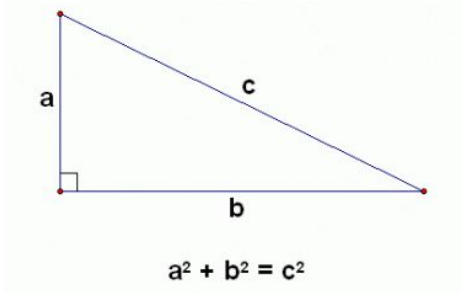
```
4
Hello what is your name
woH era uoy gniod
pineapple belongs on pizzas
Aren’t you supposed to be in bed
```

Example Output:

```
olleH tahw si ruoy eman
How are you doing
elppaenip sgnoleb no sazzip
t’nerA uoy desoppus ot eb ni deb
```

5. Right or Not?
Input File: right.txt

Pythagoras proved that every right triangle has two sides, the sum of the squares of which is equal to the square of the third, as shown below.



In *Right or Not?* You must determine whether a triangle of the three given sides is a right triangle.

Input:

The first line contains an integer N. The following N lines contain three positive integers, each representing a side of a triangle.

Output:

Output whether each triangle is a right triangle based on the given sides; if the triangle is right, print "`right`". If the triangle is not a right triangle, print "`not right`".

Example Input:

```
4
3 5 4
2 2 3
25 24 7
1 1 1
```

Example Output:

```
right
not right
right
not right
```

6. Secret Letter Strings
Input File: sletters.txt

You have received a string of letters with a message hidden in it. A “regular” letter string contains only the letters a, b, c, and d, but an agent of yours has snuck in other letters to send you information. However, as a, b, c, and d are necessary to spell words, adjacent repeated characters of a, b, c, or d represents the repeated letter. A period represents a space.

Input:

The first line contains an integer N. The following N lines are strings of letters. No other characters besides letters and periods will appear in the strings. All letters will be in lowercase.

Output:

Output the strings of letters without a, b, c, and d, unless there are several a, b, c, or d’s in a row, in which case print the repeated letter once for every repetition. So three a’s in a row should be printed as two a’s. Periods must be replaced with a space.

Example Input:

```
4
cdmcy.cbfaacdvdocritabeb.apbiebc.is.baacbdpdbpcacale
computer.science.rocks
bmbaabtbhb.bibsb.bfbubnb
abscunacbflodcawear.seaeddcabsbcdbcdbcd
```

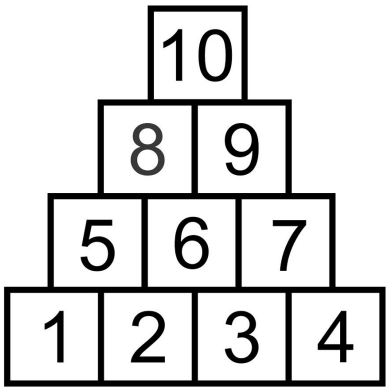
Example Output:

```
my favorite pie is apple
omputer siene roks
math is fun
sunflower seeds
```

7. Box Pyramid
Input: box.txt

As you're stacking boxes to make a triangle, you realize you left something important in one of the boxes, but thankfully you numbered them all. You just need to go to the correct floor of the pyramid and grab the box. In *Box Pyramid*, you need to determine the floor of a box of a given number in a pyramid of a given height, which will always be greater than 0 and less than a million.

The boxes are numbered ascendingly (increasing) from left to right and bottom to top; The bottom left box is one, and the box to the right is 2. After all the boxes on the first floor are numbered, the next box is the one on top of box 1 and 2. A pyramid of 4 levels, would be stacked as shown below:



Input:

The first line contains an integer N. The following N lines consist of 2 integers; the first integer defines the number of levels in the pyramid. This number will always be greater than 0. The second integer is the number of the box that you must output the level of. The second number will always signify a box in the pyramid described by the first number.

Output:

Output the level of the box indicated by the second integer, when the boxes are stacked in the method described above; with the boxes stacked in a pyramid formation and numbered ascendingly from left to right and top to bottom. If the box is on the first floor, output 1. If on the floor above, output 2, and so on.

Example Input:

```
5
1 1
3 5
2 3
4 10
7000 16978200
```

Example Output:

```
1
2
2
4
3121
```

8. ASCII Art
Input File: aart.txt

Computers save all things including art as a series of numbers. In *ASCII Art*, you must translate a series of numbers back into a piece of art based on this set of rules:

Anytime one of the numbers below is inputted replace it with the respective character:

```
1 = A,      11 = F,     21 = K,     31 = P,     41 = U,     51 = Z
2 = a,      12 = f,     22 = k,     32 = p,     42 = u,     52 = z

3 = B,      13 = G,     23 = L,     33 = Q,     43 = V
4 = b,      14 = g,     24 = l,     34 = q,     44 = v

5 = C,      15 = H,     25 = M,     35 = R,     45 = W
6 = c,      16 = h,     26 = m,     36 = r,     46 = w

7 = D,      17 = I,     27 = N,     37 = S,     47 = X
8 = d,      18 = i,     28 = n,     38 = s,     48 = x

9 = E,      19 = J,     29 = O,     39 = T,     49 = Y
10 = e,     20 = j,     30 = o,     40 = t,     50 = y

53 = SPACE
54 = .
55 = :
56 = *
```

Input:

The first line contains an integer N. The following N lines have two-digit integers separated by a space.

Output:

Output the image created when all the integers in the input file are replaced according to the rules in the description.

Example Input:

```
8
53 53 54 38 38 37 38 38 54 53 54 38 38 37 38 38 54
54 38 37 37 37 30 30 37 37 54 37 37 30 30 37 37 37 38 54
54 38 37 30 30 30 30 30 30 30 30 30 30 30 30 37 38 54
54 38 37 53 17 53 23 30 44 10 53 49 30 42 53 53 37 38 54
53 53 54 38 37 30 30 30 30 30 30 30 30 30 37 38 54
53 53 53 54 38 37 37 30 30 30 30 30 37 37 38 54
53 53 53 53 53 54 38 37 30 30 30 37 38 54
53 53 53 53 53 53 53 53 54 38 54
```

Example Output:

```
  .ssSss. .ssSss.
.sSSSooSS.SSooSSSs.
.sSooooooooooooSs.
.sS I Love You  Ss.
  .sSooooooooSs.
   .sSSooooSSs.
     .sSoooSs.
        .s.
```

9. Wo-Word-rd
Input File: wowordrd.txt

You have just come across an alien race which constructs sentences by inserting the second word in a sentence into the first word and the third word into the second word, and so on until the final word is inserted into the second-to-last word. Your job in Wo-Word-rd is to construct an alien sentence normally when given a list of all possible words.

Input:

The first line contains an integer N. The following N lines have a single word. After the N lines of words are passed in, an alien sentence, according to the rules above and composed only with the words passed in, is inputted. The alien sentence may not use all the given words and may use some words multiple times. All words will be inputted in lowercase but the sentence may have uppercase words which should be preserved. There will be only one correct way to reconstruct the sentence with the words given.

Output:

Output the alien sentence as it would be in a normal structure: with the first word first, then a space, then the second word, and so on.

Example Input:

```
9
cat
park
walk
ark
the
likes
like
to
in
Tcaliktwalithparkenkoesthe
```

Example Output:

```
The cat likes to walk in the park
```

10. Shoe Scramble
Input File: shoe.txt

In my house, we can never keep shoes with the correct pair. In *Shoe Scramble*, you must determine how many swaps of shoes are needed to arrange the shoes into the correct pairs.

Input:

The first line contains an integer N. There are N cases. The first line in every case contains an integer M, which defines the number of pairs of shoes. M lines of shoe pairs are then inputted. Each pair consists of two characters separated by a space. Each character will appear twice once on the right of the space and once on the left, signifying a right and left shoe. The given lines define the current arrangement of the shoes. After the pairs, the next case is inputted.

Output:

For each case, output the number of swaps needed to rearrange the pairs into the correct order. A swap is considered switching the positions of two right shoes. The shoes are in the correct order when every pair consists of two shoes of the same character.

Example Input:

```
2
9
a c
D D
1 6
c 1
6 y
y h
h a
2 3
3 2
4
a a
b b
c c
d d
```

Example Output:

```
6
0
```

11. Tunnels
Input File: tunnel.txt

In *Tunnels*, you must find the shortest path from a starting point to an ending point on a map with tunnels.

Input:

The first line contains an integer N. There are N cases. The first line in every case contains two integers, the first of which defines the number of columns in the map of characters, the second of which defines the rows. The map described by those two integers is then inputted. After the map is inputted, the current case ends and the next begins.

The map will be defined with periods, one-digit numbers, an s, and an x. The periods represent empty spaces. The s represents the starting spot, and the x represents the ending spot. The numbers will come in pairs. Each pair represents a tunnel which you can step from one end to the other. It takes one step to move from a space to an adjacent space, but no steps to cross a tunnel. You may only move up, down, right, and left, and you may not move diagonally.

Output:

Output, for each case, the smallest number of steps it takes to travel from the starting s to the x.

Example Input:

```
4
3 2
s.1
1.x
6 6
......
.x2...
3.....
..3...
....s1
.21...
4 4
1.1.
.s..
....
...x
8 8
.......4
2...s...
........
.3......
.3..2..4
........
........
1x.....1
```

Example Output:

```
2
3
4
8
```

12. Matching Up
Input File: match.txt

Valentine has just passed, but people are still trying to find their soulmate. In *Matching Up*, you must determine whether everyone can be couple up when given a list of possible matches.

Input:

The first line contains an integer N. There are N cases. The first line in every case contains an integer M. The following M lines contain two names separated by a space, each representing a possible match between the two names. After M lines, the current case ends and the next starts.

Output:

Output, for each case, whether there is a suitable match for everyone described by the possible matches without anyone being in two matches. If there is a suitable match for everyone, output "`Match`". If there is not, output "`No Match`".

Example Input:

```
3
8
Carl Jessie
Andy Carl
Jessie Adrian
Adrian Andy
Andy Samantha
Samantha Carl
Jessie Samantha
Fred Jessie
3
Beast Belle
Gaston Belle
Gaston LeFou
2
Juliet Romeo
Juliet Paris
```

Example Output:

```
Match
Match
```

No Match

13. Crumbling Structure
Input File: structure.txt

Old buildings can be very dangerous; their walls crack and crumble, causing instability in the higher regions of the buildings. In *Crumbling Structure*, you must determine how much of a building is destroyed when the walls of a given strength are damaged to a certain degree.

Input:

The first line contains an integer N. The following N lines define a 2D building in integers. Each integer represents a room with wall strength equal to that integer. The building is constructed with the first line being the top floor and the last line being the bottom. There may be periods in the lines defining the building; they represent an absence of a room. After the N lines are inputted, a single integer is inputted representing the damage done to all the walls.

Output:

Output the total number of rooms that are destroyed by the damage. Any room with wall strength equal to or less than the integer representing the damage done to the walls collapses. Additionally, any room without a chain of adjacent rooms connecting it to the ground collapses. In the example input below, the top floor of 2s are not destroyed but the rooms holding it up are. So it collapses, but if the second `1` in the second row was a two, the entire top floor would be saved because now there is a chain of adjacent rooms connecting them to the ground. Likewise, the `1` on the ground floor does not cause the room above it to collapse because it is connected to the ground by the adjacent `3`.

Example Input:

```
6
222222222
.1.....1.
.1222334.
.4.....5.
333333333
333...331
1
```

Example Output:

```
13
```

14. Balancing the Chemical Equation
Input File: chemequation.txt

One of the most basic tasks all chemistry students must know how to do is balance a chemical equation with the lowest whole number coefficients.

The equation will be comprised of a list of compounds and elements, separated by plus-signs, on both sides of an equal sign. An element will be represented by either one or two letters; the first is always capitalized, and if the element has a second character, it will be lowercase. Compounds are composed of several elements added together and with a number following each element. This number represents the number of atoms of the element in front of the number in the compound. If there is only one atom of an element in the compound, no number will appear after the element.

You must output the lowest whole coefficients - the number - of each compound such that the equation is balanced. An equation is balanced when the number of atoms of each element on the left side is equal to that of the right side. Coefficients are applied to the complete compound; so a coefficient of two for `FeCl` represents two atoms of `Fe` and two atoms of `Cl`.

Input:

The first line contains an integer N. The following N lines will define equations according to the rules described above.

Output:

For each equation, output the lowest whole number coefficients such that the equation is balanced.

Example Input:

```
4
Fe + Cl2 = FeCl3
KMnO4 + HCl = KCl + MnCl2 + H2O + Cl2
C8H18 + O2 = CO2 + H2O
N2O2 = N4O4
```

Example Output:

```
2 3 2
2 16 2 2 8 5
2 25 16 18
2 1
```

15. Flooding an Anthill
File Input: anthill.txt

You have a gigantic 2-dimensional anthill in your backyard, and you want to flood it with one pour of water. However, to stop you, the ants have built flood-proof walls, which you must breakdown. Thankfully, you have a tiny drill, but you hate using it. Calculate the minimum amount of walls that need to break down to flood the entire anthill.

Input:

The first line will contain an integer M. The first two will define the rows and columns of a d-dimensional map, respectively. After the two integers, the map will be inputted. The map represents the anthill with the `.`'s being empty spaces and `x`'s being blocks of dirt. All empty spaces are considered part of the anthill.

Output:

For each case, output the minimum number of blocks of dirt that would need to be removed for the anthill to ensure that the anthill would be completely filled with water. You may assume water is flowing in from all sides of the anthill.

Example Input:

```
3
5 6
xxxxxx
x...xx
xxxx.x
xxx.xx
..xxxx
1 4
....
4 7
xxxxxxx
x.xxx.x
xx.x.xx
xxx.xxx
```

Example Output:

```
2
0
3
```