CCSC
Consortium for Computing Sciences in Colleges
Central Plains Region

2026 CCSC

Central Plains Region Programming Contest

Official Problem Set

April 11, 2026

CollegeBoard Rephactor acm2y acm CCECC Committee for Computing Education in Community Colleges Blossoms map·cs MISSION-ALIGNED PROGRAMS IN COMPUTER SCIENCE

CCSC
Consortium for Computing Sciences in Colleges
Central Plains Region

APRIL 11, 2026

Contest Problems

A: Black Velvet
B: Ardi, the Hungry Aardvark
C: Pinewood Derby
D: Row, Row, Row your Boat
E: Running Lights
F: Subset Sum
G: Will It Float?
H: Woodchucks Chucking Wood

CCSC

Consortium for Computing Sciences in Colleges
Central Plains Region

This 4 hour contest includes 8 problems over 22 pages. Good luck.

Prior to the CCSC competition, the judges will have solved all problems in languages from at least two of these three distinct language groups: Java/Kotlin, C/C++, and Python3.

Definition 1

For problems that ask for a result modulo m:
If the correct answer to the problem is the integer b, then you should display the unique value a such that:

- $0 \leq a < m$

  and

- $(a - b)$ is a multiple of m.

Definition 2

A string $s_1 s_2 \cdots s_n$ is lexicographically smaller than $t_1 t_2 \cdots t_\ell$ if

- there exists $k \leq \min(n, \ell)$ such that $s_i = t_i$ for all $1 \leq i < k$ and $s_k < t_k$

  or

- $s_i = t_i$ for all $1 \leq i \leq \min(n, \ell)$ and $n < \ell$.

Definition 3

- Uppercase letters are the uppercase English letters $(A, B, \ldots, Z)$.
- Lowercase letters are the lowercase English letters $(a, b, \ldots, z)$.

This page is intentionally left blank.

CCSC
Consortium for Computing Sciences in Colleges
Central Plains Region

Problem A
Black Velvet

Ok, this is not really black velvet - the nap doesn't show up well in black!

Picture a lush black velvet blanket with a deep nap that the owner enjoys brushing with her hand which leaves a pattern as the nap is laid down in the direction of the hand's movement. Your task is to analyze this pattern. However, to make this task easier to handle, we will impose some constraints:

1. Initially, the nap will not be laid down in any direction.
2. The hand movement will be strictly left or right across the blanket so you only have to deal with one dimension.
3. The hand will be lifted off the blanket at the end of each sweep during the testing period, which may include many left and right sweeps.
4. Each hand sweep will be specified as a pair of integers indicating the starting position on the blanket and the distance/direction swept.
5. Each sweep of the hand will replace the effect of any previous sweeps over that same area.

Input

The first line of input will contain a single non-negative integer n (at most 10,000) that indicates the number of sweeps. Following this first line will be n pairs of integers, 5 pairs per line, indicating the starting position of the sweep (non-negative) and the distance swept right (if positive) or left (if negative). If the number of sweeps is not a multiple of 5, the last line will be padded with 0's. The sweeps will not extend beyond the ends of the blanket which will be at most 10^9 units in length.

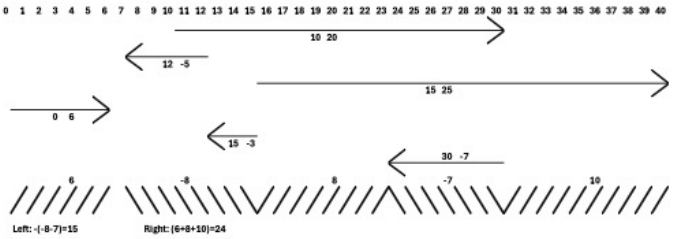


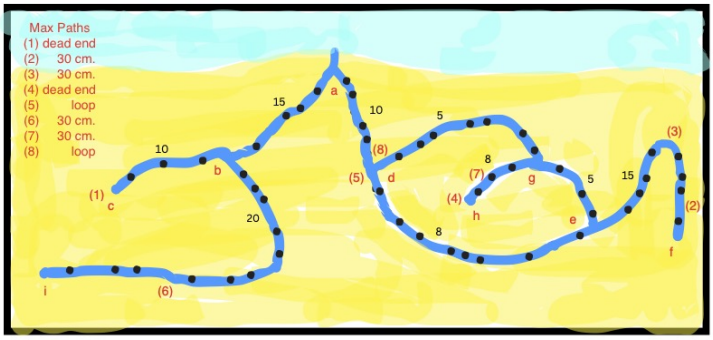
Illustration of Sample 1. Solution: Left 15, Right 24

Output

First print the total resulting distance of the velvet lying to the left, and on the next line the total resulting distance lying to the right. Format as in the sample.

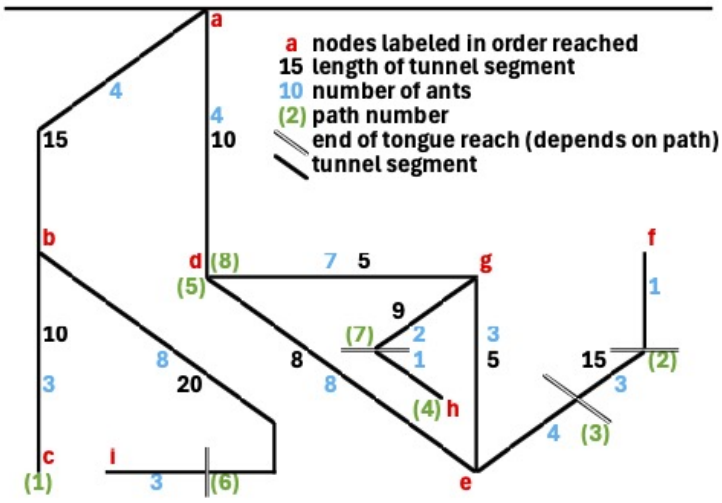
Sample Input 1	Sample Output 1
6 10 20 12 -5 15 25 0 6 15 -3 30 -7 0 0 0 0 0 0 0 0	15 24

Problem B
Ardi, the Hungry Aardvark



This is the ant farm specified in Sample 1

Ardi the aardvark arrived at the pet shop tired and hungry. Ardi spotted an ant farm on the next shelf, and began plotting how to satisfy herself after the shop closed. Ardi knew the length of her tongue was a standard 30 cm. She could visually measure the lengths of the tunnel sections and could count the number of ants in them. Her objective was to get in one good probe of her sticky tongue to catch the maximum number of ants. She realized that after that one probe, any remaining ants would be on alert and hastily dig their way beyond any possible future probe.



Schematic of the ant farm with details

Ardi figured that in some cases the optimal probe might be to follow a simple path through the tunnels that ends in a loop, like a lasso. She could then allow time for additional ants on any surrounded interior branches to "escape" directly to her encircling tongue. Thus she needs to calculate all possibilities, both those ending with a loop and those without.

For each path, she maintains a record with a sequence of L (for left) and R (for right) choices at each junction, with left and right being relative to the direction her tongue would be going. Upon entering a junction there are exactly two choices, leftmost and rightmost, even though both choices may be visually to the left or to the right. Ardi also records the length of each segment up to her full 30 cm. reach. Each possible path ends at either a dead end, at the full reach of her tongue, or at a junction completing a loop (note that her tongue is too sticky to slide through the same

segment or junction twice.) She does not keep a record of tunnel segments beyond her reach, other than to count the number of unreachable ants, which could be good to know if she can find a loop around them as described earlier.

Ardi can make a complete analysis of the ant farm since the sides are transparent. This is how we get the data about the possible paths. However, there is just one actual probe with the tongue. Remember that if Ardi completes a whole loop with her tongue, there may be inner branches into the interior of the loop. Any ants in that area are also trapped and will eventually reach the encircling tongue and be caught. They can't hide forever!

Input

The first line of the input contains a single integer indicating the number of paths Ardi traces through the ant farm. Each subsequent line details one of the paths, beginning with the number of segments in the path. Altogether, every reachable tunnel segment of a path is checked. Each segment consists of four parts: the starting junction (lower case letter starting with 'a'), the letter 'R' (for turn right) or 'L' (for turn left), the length (a positive integer) of the segment, and the number of reachable ants in the segment.

The final segment of a path will be followed by this (which for your convenience is formatted similarly): the ending junction letter (or termination point) which could indicate the closing of a loop if Ardi's tongue can reach that far, the letter 'X', the number of remaining unreachable ants (if any) in the final segment, then a 0 (zero).

Output

Print a single integer indicating the maximum number of ants Ardi can catch with a single probe. That total might include all those on segments within a loop.

Sample Input 1

```
8
2   a R 15 4   b R 10 3   c X  0 0
3   a L 10 4   d R  8 8   e R 15 7   f X  1 0
4   a L 10 4   d L  5 7   g L  5 3   e L 15 4   f X  4 0
3   a L 10 4   d L  5 7   g R  9 3   h X  0 0
4   a L 10 4   d L  5 7   g L  5 3   e R  8 8   d X  0 0
2   a R 15 4   b L 20 8   i X  3 0
4   a L 10 4   d R  8 8   e L  5 3   g L  9 2   h X  1 0
4   a L 10 4   d R  8 8   e L  5 3   g R  5 7   d X  0 0
```

Sample Output 1

```
25
```

CCSC
Consortium for Computing Sciences in Colleges
Central Plains Region

Problem C
Pinewood Derby

Snapshot of a Pinewood Derby heat

A pinewood derby is a competition in which grade school kids race with cars they make out of blocks of pine. There are rules and protocols to try to make the competitions reasonably fair. However, one ongoing point of contention is the mechanism for scoring and combining the results of the various heats.

For one particular competition, the judges are going to measure the time in hundredths of seconds that it takes cars to reach the bottom of the track. They will have each car run once on each track of the course in separate heats in order to compensate for differences among the tracks.

The judges want to know if using the average times on all the tracks gives the same results as assigning points based on the ranks in each of the heats. The points awarded would essentially be the opposite of the ranks, thus starting at 0 for the last place car in the heat. For example, if a course has 4 tracks so that 4 cars are racing together, the first place finisher would get 3 points for that heat, 2^{nd} place would get 2 points, 3^{rd} place would get 1 point and 4^{th} place would get no points.

Input

The first line will contain two integers: the number of tracks, t, and the number or pinewood derby cars, c, which is also the number of heats, with $50 >= c > t >= 2$.

After this will come the descriptions of the c heats, one line per heat. A heat includes the car details for each of the t tracks in order by track. First is the (unique) name for the car of 1-8 lower case letters, which will be consistent across the heats. Following the name will be

Track Number

Heat Number	1				2				3			
	Name	Time	Rank	Points	Name	Time	Rank	Points	Name	Time	Rank	Points
1	ace	50.2	2	1	bat	49.5	1	2	cat	63.2	3	0
2	bat	45.7	1	2	cat	52.1	2	1	dog	56.4	3	0
3	cat	55.1	2	1	dog	62.1	3	0	elf	52.8	1	2
4	dog	60.6	3	0	elf	58.3	2	1	fig	49.5	1	2
5	elf	47.8	1	2	fig	53.9	2	1	ace	58.9	3	0
6	fig	52.3	1	2	ace	52.3	1	2	bat	52.6	2	1

Name	Average Time	Place by Time	Point Total	Place by Points
ace	53.80	4	3	2
bat	49.27	1	5	1
cat	56.80	5	2	3
dog	59.70	6	0	4
elf	52.97	3	5	1
fig	51.90	2	5	1

Rank by Time	Rank by Points
bat	bat
fig	elf
elf	fig
ace	ace
cat	cat
dog	dog

Example with 6 cars on 3 tracks (thus 6 heats with each car on each track once, though not all cars have a chance to compete against each other)

the time it took that car to make it down the track for that heat, measured in hundredths of a seconds.

Output

Print one line per car in the rank order that the average time would place them. In case of a tie, order the ties alphabetically by name. For each of these lines, first print the rank number that the car would have received by using the point method of scoring, followed by a space and then the name of the car. If the cars tie by points, assign them the same rank so that the next car would get the next numerical rank (eg. 2 cars tying for first by points would both get 1^{st}, the next car would be listed as 2^{nd}, etc.)

Sample Input 1	Sample Output 1
2 3 candy 54.60 kite 45.20 kite 60.00 cat 56.90 cat 64.20 candy 60.50	1 kite 1 candy 1 cat

Problem D
Row, Row, Row your Boat

Row, row, row your boat,
Gently down the stream.
Merrily, merrily, merrily, merrily,
Life is but a dream.

Row, row, row your boat,
Gently down the stream.
If you see a crocodile,
Don't forget to scream.

Row, row, row your boat,
Gently down the river.
If you see a polar bear,
Don't forget to shiver.

Row, row, row your boat,
Gently down the creek.
If you see a little mouse,
Don't forget to squeak.

Row, row, row your boat,
Gently to the shore.
If you see a lion,
Don't forget to roar.

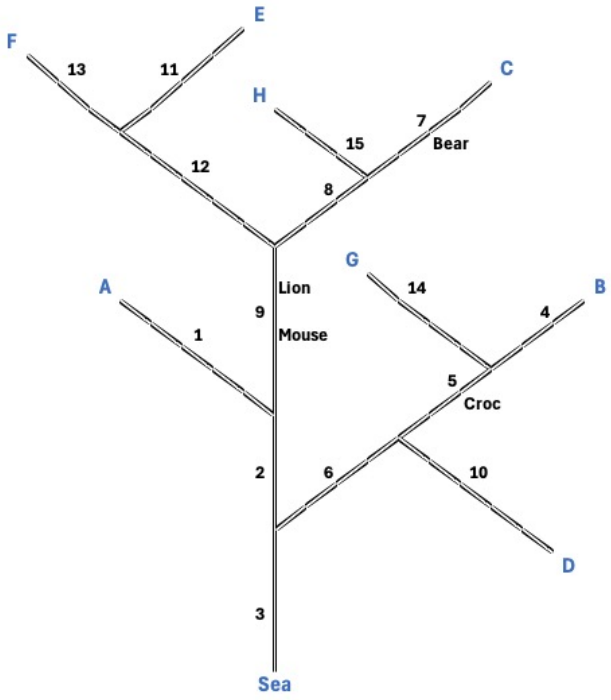


Illustration of Sample 1 River System. Routes are lettered in blue. Segments are numbered. Critters may be abbreviated.

Some friends wish to take a leisurely ride floating down the river all the way to the sea. They want you to help them choose the starting dock that maximizes their time. This may involve a whole river system with multiple branches and subbranches that merge on the way to the sea. They may encounter interesting critters along the way which may induce the boaters to respond in various ways. You will be provided with the details of the river system, including branch lengths, locations of possible starting docks, average speed of each stream segment, and occurrences of any critters. For each critter you will be provided with its type, and what you may need to say or do as you pass it. A segment is defined as the portion of the stream between two consecutive confluence points where streams merge or where a potential starting dock is located or the mouth where the river meets the sea.

Input

The input will detail the routes from feasible docks to the sea. The routes will be presented as a series of segments from the dock to subsequent confluences with other streams and finally to the sea.

Input will begin with a line containing two positive integers, each less than 100, indicating the numbers of routes (docks) and the number of segments. Both the routes (and their entry docks) and the segments will be numbered in order of presentation in the input, which is also the order each route would be traversed, at least up to where a route description (after the first route) ends as it merges into a previously presented route. (This is done to reduce redundancy in the input.) At that point a back link will be provided to the remainder of the segments for that route.

The first segment description will be for route 1, segment 1. Each segment will be described as follows: The first line contains 1) its length (in km), 2) its speed (in meters per minute), 3) a link to the next segment which could be a 1 indicating the following segment or a back link (integer greater than 1) to an already described segment of an earlier route, or a 0 to indicate the sea, and finally 4) the number of critters on that segment. If there are any critters, they will be detailed on subsequent pairs of lines by two strings each: its critter type (such as crocodile, polar_bear, mouse, or lion) and the response to it (scream, shiver, squeak, or roar). The following annotates the Sample 1 input for the routes and segments:

```
8 15          (8 routes, 15 segments)
30 100 1 0    (route 1 segment 1 - 30 km, 100 m/min, next segment, no critters)
15 125 1 0    (route 1 segment 2 - 15 km, 125 m/min, next segment, no critters)
20 100 0 0    (route 1 segment 3 - 20 km, 100 m/min, reaches the sea, no critters)
18 150 1 0    (route 2 segment 4 - 18 km, 150 m/min, next segment, no critters)
15 150 1 1    (route 2 segment 5 - 15 km, 150 m/min, next segment, 1 critter)
crocodile
scream
20 100 3 0    (route 2 segment 6 - 20 km, 100 m/min, merge with segment 3, no critters)
25 105 1 1    (route 3 segment 7 - 25 km, 105 m/min, next segment, 1 critter)
polar_bear
shiver
10 90 1 0     (route 3 segment 8 - 10 km, 90 m/min, next segment, no critters)
25 90 2 2     (route 3 segment 9 - 25 km, 90 m/min, merge with segment 2, 2 critters)
lion
roar
mouse
squeak
30 75 6 0     (route 4 segment 10, 30 km, 75 m/min, merge with segment 6, no critters)
20 100 1 0    (route 5 segment 11, 20 km, 100 m/min, next segment, no critters)
30 100 9 0    (route 5 segment 12, 30 km, 100 m/min, merge with segment 9, no critters)
```

15 130 12 0 (route 6 segment 13, 15 km, 130 m/min, merge with segment 12, no critters)
30 100 5 0 (route 7 segment 14, 30 km, 100 m/min, merge with segment 5, no critters)
15 120 8 0 (route 8 segment 15, 15 km, 120 m/min, merge with segment 8, no critters)

Output

Determine the route that maximizes the time of the trip down the river. First print the number of the starting dock. As you pass critters along that route, print what you have been told to do, such as "shiver" or "squeak", each on a separate line. Finally, print the time of the trip in hours, minutes, and seconds, formatted with colons ':' and no spaces, and two digits for each of minutes and seconds. Round down to the nearest second.

Sample Input 1

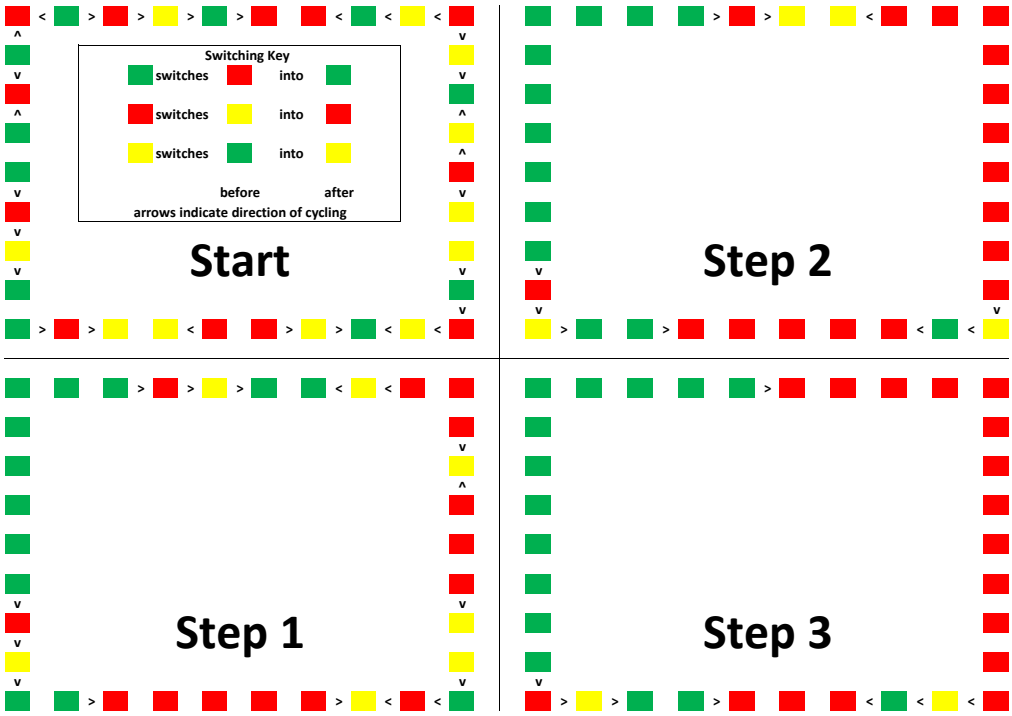
```
8 15
30  100   1   0
15  125   1   0
20  100   0   0
18  150   1   0
15  150   1   1
crocodile
scream
20  100   3   0
25  105   1   1
polar_bear
shiver
10   90   1   0
25   90   2   2
lion
roar
mouse
squeak
30   75   6   0
20  100   1   0
30  100   9   0
15  130  12   0
30  100   5   0
15  120   8   0
```

Sample Output 1

```
5
roar
sqeak
18:17:47
```

This page is intentionally left blank.

Problem E
Running Lights



Lights were installed around a mirror and controlled with some custom programming to make them appear to run in a cycle, either clockwise or counterclockwise. Unfortunately, the programming was screwed up.

The lights were indeed programmed as follows: Red changes to green if there is at least one green next to it. Green changes to yellow if there is at least one yellow next to it. Yellow changes to red if there is at least one red next to it. Under any other circumstance, the color of the light stays the same.

For some configurations of lights, the pattern does eventually appear to move in a cycle, sometimes with clusters of the same color. Otherwise, we can show that the system eventually settles on a single color, though initially some sections may appear to go in different directions or may expand or shrink.

Your task, given an initial configuration, is to determine if the pattern eventually runs in a repeating cycle,

and if so, whether it is clockwise or counterclockwise. If it does not result in a cycle, determine what the remaining color will be.

Input

The one input line contains a string of color designations for the (clockwise) order of lights as R (red), G (green) and Y (yellow).

Output

If eventually only one color remains, print that color as Red, Yellow, or Green. If all colors remain indefinitely, print what direction the lights appear to run as Clockwise or Counterclockwise.

Sample Input 1	Sample Output 1
RGRYGRRGYRYGYRYYGRYGYRRYYRGGYRGGRG	Green

Sample Input 2	Sample Output 2
RRYGRYRYYGGG	Clockwise

Problem F
Subset Sum

$\{\{\}, \{1\}, \{2\}, \{3\}, \{4\}, \{1,2\}, \{1,3\}, \{1,4\}, \{2,3\}, \{2,4\}, \{3,4\}, \{1,2,3\}, \{1,2,4\}, \{1,3,4\}, \{2,3,4\}\}$ - This is the set of all proper subsets of the numbers 1, 2, 3, and 4 (which doesn't include the complete set $\{1,2,3,4\}$). Adding the 7 instances of each of the 4 numbers yields a sum of 70.


Is this a subset or just a set of subs?

Jack has been working with an arbitrary set of numbers and notices that he can make many different proper subsets with them. He starts to wonder what the sum of sums of all of these subsets will be. After a while he realizes that if he knew the sum of all the numbers over all these subsets, it may be useful to his continued analysis. He has asked you to compute this sum for him when you are given the list of (unique) numbers that he is considering.

To put this into slightly different terms, you will be given a set of numbers (so there will be no repeats due to the definition of a set). You are to find all proper subsets of this set (a proper subset is any collection of items from the set other than selecting all of them). The answer you need to print out is the sum of the numbers in each of these subsets, all added together.

Input

On the first line will be an integer N, with $1 < N < 100$.
On each of the next N lines, there will be an integer. The collection of these will not include duplicates. Each of these integers will be between -10,000 and 10,000 inclusive.

Output

Print on a line by itself the sum of all the occurrences of the numbers in all of the proper subsets of the given numbers when taken modulo 91919. (Print out the remainder when the sum is divided by 91919).

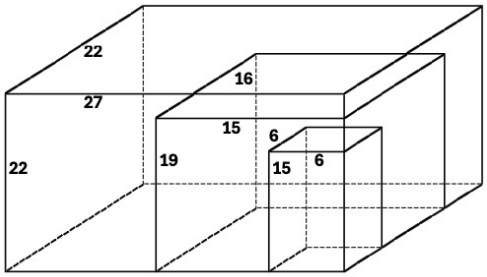
Sample Input 1	Sample Output 1
1 7	0

Sample Input 2	Sample Output 2
4 1 2 3 4	70

CCSC
Consortium for Computing Sciences in Colleges
Central Plains Region

Problem G
Will it Float?

You are provided a set of rectangular boxes of negligible weight and thickness, some of which can be nested inside others. Upon being nested, any remaining space in a box can either be filled with water or left empty before being closed. Find the subset of boxes that when nested and with the remaining spaces being optimally either filled with water or left empty, results in the outermost box floating closest to halfway deep when placed in a pool of water. Two boxes can be nested only if they can be oriented such that all the dimensions of one are strictly smaller than that of the other. Allow any level of nesting, but only one box per level (so you don't have to deal with Tetris-like packing). The boxes should be rectilinearly aligned.



Sample nesting of boxes. Note that a fourth box 20x20x30 could be an option for the outermost box, but all four would not be mutually nest-able.

Input

The first line provides the number (no more than 10) of boxes. Each subsequent line details the 3 dimensions of one of the boxes (length, width, and height in arbitrary order) and correct to two decimal positions. Consider the boxes to be labelled with consecutive upper-case letters starting with 'A' as they appear in the input. They are not necessarily presented in order of size, and are likely not even totally sortable by size.

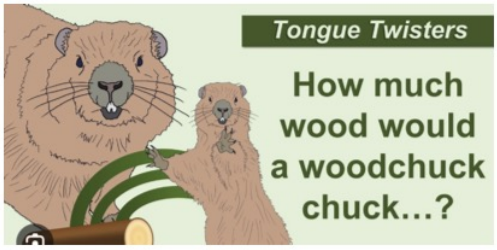
Output

Format as demonstrated. List the boxes in the order nested, one per line (outermost first) along with an indication of any remaining space being either full or empty of water. Leave one space between the box label and the indicating "empty" or "full" message. On the line following the list of boxes, Indicate the percentage optimally closest to 50% of the outermost box that would be underwater when placed in the pool (rounded to the nearest per cent.) In case of a tie, choose the nesting whose label sequence comes lexicographically first, eg. BAC beats BC. If there is still a tie (because of differences in being filled), choose the nesting that floats highest. No solution will be exactly 50%.

Sample Input 1	Sample Output 1
3 15 19 16 27 22 22 15 6 6	B full A empty 65%

Problem H
Woodchucks Chucking Wood

How much wood could a woodchuck chuck
If a woodchuck could chuck wood?
As much wood as a woodchuck could chuck,
If a woodchuck could chuck wood.



Here's a common example of a tongue twister.

In this and similar verses the tongue twisting arises from the repetition of words, slight variations of spelling, sounding, and splitting of words, and the rhythm of the verse. Given various verses, the challenge in identifying which are tongue twisters comes in identifying and counting such characteristics.

For this problem all you need to do is to take two lines of text and determine their similarity by finding a non-empty substring in each of these two strings such that they minimize the value of the following expression:

> "The number of unique letters over the two substrings divided by the sum of the lengths of the two strings"

Upper and lower case letters are considered the same letter. We will only be considering the alphabetic characters (a-z,A-Z). Let's use the first two lines of the verse at the top as an example (followed by the letter count):

```
howmuchwoodcouldawoodchuckchuck
ifawoodchuckcouldchuckwood
00000000001111111111222222222 3
01234567890123456789012345678 90
```

If we used a single letter such as 'a' from each, our ratio would be 1/2 (1 unique letter 'a' - sum of lengths 2). If we used the word "wood" from each, our ratio would be 3/8 (3 unique letters w,o,d - sum of length 8), slightly better. We can do even better if we use "uch wood could a woodchuck chuck" from the first line and "a woodchuck could chuck wood" from the second, giving us a ratio of 9/51 = 3/17 (9 unique letters w,o,d,c,u,l,a,h,k - 27 and 24 letter lengths for each substring respectively for a sum of 51 in all).

CCSC
Consortium for Computing Sciences in Colleges
Central Plains Region

Input

Each of the two lines will be no longer than 80 characters in length (including any space characters that may be present.) Each line will contain at least one letter character. Only upper and lower case letters and the space character will appear on either line - any punctuation will have been stripped out.

Output

On a single line print the fraction that you have computed. Print the numerator, then a space, the '/' character, another space, and then the denominator. Do reduce to lowest terms.

Sample Input 1	Sample Output 1
How much wood could a woodchuck chuck If a woodchuck could chuck wood	3 / 17