Inspirational Message
Input File: inspiring.txt

In *Inspirational Message*, you are trying to cheer up your friend. You must write an inspirational note to him to brighten his day. You will be given a certain number of words that you must print to the console in the specific format given below.

Input:

The first line will have a number specifying the number of lines below. The next lines will each contain one word (string). You do not need to check the validity of the input.

Output:

You will output the words in the specific format shown in the example output below. You must print out **everything** shown below the example output, including all of the dashes, slashes, etc. in the answer.

Example Input:

```
3
Amazing
Awesome
Fantastic
```

Example Output:

```
/----------\
\*!Amazing!*/
\*!Awesome!*/
\*!Fantastic!*/
\----------/
```

Small Car
Input File: None

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

Input:

None.

Output:

You will output the exact car is shown below.

Example Output:

```
    .-----.
 __//.{)|_\\._
/___         ___:
--(_)------(_)--
```

Triangle Creator
Input File: triangle.txt

You have a math exam next period, and your friend tells you one of the questions, but forgets the exact values. He tells you that one of the problems asks to draw x number of triangle of varying N heights. You are unaware of the number of triangles you will need to draw. Create a program that will print a triangle, given N, with '/' and '\' for the sides and '_' for the bottom, with N being the triangle's height in lines.

Input:

The first line represents x, or the number of triangles to draw. Each following line will contain N, or the height of the triangle.

Output:

Output x number of triangles, using '/' and '\' for the sides and '_' for the bottom, with the given height(s) of N, including a line to separate the printed triangles.

Example Input:

```
2
3
5
```

Example Output:

```
  /\
 /  \
/____\

    /\
   /  \
  /    \
 /      \
/________\
```

Sorting Distances
Input File: sorting.txt

In *Sorting Distances*, you are a taxi driver who is trying to find the different paths to your destination in descending order. However, all you have been given are a bunch of scrambled distances. Your job is to return the distances you have been given sorted from **greatest to least**.

Input:

The first line contains an integer N. The following N lines will contain one number each, representing a path's distance to the desired destination. All distances will be greater than 0, and no two distances will be the same.

Output:

You will need to output all of the distances sorted from greatest to least, with each distance on a different line.

Example Input:

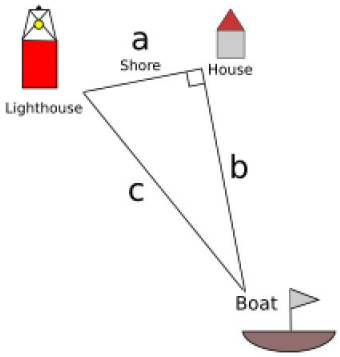
```
4
489
507
23
17001
```

Example Output:

```
17001
507
489
23
```

Distance to Shore
Input File: distance.txt

In *Distance to Shore*, you are in a boat a certain distance from shore. On the shore there is a house, and a little farther to the left is a lighthouse. The boat, the house, and the lighthouse form a right triangle. The following diagram illustrates the positions of the boat, the shore, and the lighthouse. You know the distances between two of the different objects, and must find the third using the Pythagorean Theorem.



Input:

The first line contains an integer N. The following N lines will contain two numbers, with an a, b, or c preceding each number. The letters represent one of the sides that is shown above - c is a hypotenuse, and a and b are legs. You can assume that c will be greater than a and b.

Output:

You will need to output the length of the side that is not given using the Pythagorean Theorem. Round the answer **down** to the nearest integer.

Example Input:

```
2
A 15.2 C 18.3
B 5 A 12
```

Example Output:

```
10
13
```

Word Spelling
Input File: spelling.txt

In *Word Spelling*, you are given a word, and multiple different letters following the word. You must determine if those letters can be arranged to spell the word shown. Not all the letters have to be used, and no letter can be used twice. Capitals are different from lowercase.

Input:

The first line contains an integer N. The following N lines will contain one word, followed by an integer K representing the number of characters following.

Output:

You will need to output “yes” or “no” depending on whether or not the list of letters can be used to spell the word.

Example Input:

```
4
apple 6 l y a p e l
John 3 J h n
bottle 10 e e t l b o o e a c
Poster 9 r t a n o P p s e
```

Example Output:

```
no
no
no
yes
```

Compound Interest
Input File: compound.txt

Suppose Michael has P dollars to invest in an account that pays r% interest compounded quarterly. How much money does Michael have after t years?

The formula for compounded interest is:

$A = P\,(1 + r/n)^{(nt)}$

Input:

The first line is the number of lines to follow. Each line will contain the initial investment into the bank account (P), the number of times the interest is compounded annually (n), the interest rate (r) (in decimals), followed by the time in years since the initial investment was deposited into the account (t) (in order).

Output:

Output the amount of money in the account after t years rounded **down** to the nearest integer, including a dollar sign ($) before the value.

Example Input:

```
2
1000 4 0.035 5
2500 2 0.05 10
```

Example Output:

```
$1190
$4096
```

Magic Square
Input File: magic.txt

A magic square is a 3x3 square where the numbers 1-9 fill up each tile, and each row, column, and diagonal adds up to 15. The table below shows a magic square.

2	9	4
7	5	3
6	1	8

As you can see, if you take any three numbers that make up a row, column, or diagonal, the sum is always 15. Unfortunately, the pattern shown above is the only way to make it perfectly. Your friend is trying to make these magic squares, but most of the time, a few of the rows don't perfectly add up to 15. Your task is to count up how many rows, columns, and diagonals that need to be fixed before it can be a magic square.

Input:

The first line contains an integer N, representing the number of sets of data. The next N sets of data each contain three lines with three numbers on each line.

Output:

Output the number of rows, columns, and diagonals that don't add up to 15 for each set of data.

Example Input:

```
2
5 6 4
7 8 9
3 1 2
4 1 7
3 9 6
8 5 2
```

Example Output:

```
2
3
```

Texas Hold'em
Input File: poker.txt

You are playing a game of Texas Hold'em poker against your friend. There are five cards on the flop and you and your friend both have two cards in hand. You know that you have a pair or a triple, but need to find what is the highest pair or triple you have in a hand of seven cards. In Texas Hold'em, 2 is the lowest card and ace is the highest. Find the highest pair you have in a hand of five cards, or the highest triple if it exists.

Input:

The first line represents the number of lines to follow. Each line contains the sequence of seven cards you have in your hand and on the flop, separated by a space.

A - Ace
K - King
Q - Queen
J - Jack

Output:

Output the highest pair from the seven cards. If a pair and a triple exists in your hand of seven cards, the triple supersedes the pair and is therefore the only value outputted.

Example Input:

```
2
10 10 9 9 A J K
Q 2 J Q J Q A
```

Example Output:

```
10 10
Q Q Q
```

Valid DNA
Input File: dna.txt

DNA is made of of nitrogen bases that form a long sequence. There are four nitrogen bases: A, T, C, and G. Since DNA is made of two strands, each of these sequences is paired with another strand that contains the exact opposite pattern. An A on one strand always pairs up with a T on the other and a C on one strand always pairs up with a G on the other. Therefore, if one sequence is ATCGC, the other sequence will be TAGCG. Your task is to determine whether the given DNA sequences follow this rule or not.

Input:

The first line contains an integer N, representing the number of sets of DNA. The following N sets of data each contain two lines, with one ten-letter sequence of A's, T's, C's, and G's on each line.

Output:

For each set of data, determine whether the two sequences of nitrogen bases are valid to be paired together (A must go with T, C must go with G). Either output "Valid" or "Invalid."

Example Input:

```
3
ACTACGCATC
TGATGCGTAG
GCATGACTAG
CGTAGACATC
GTCATCGATC
CAGTAGCTAG
```

Example Output:

```
Valid
Invalid
Valid
```

Frog
Input File: frog.txt

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

Input:

None.

Output:

You will output the frog exactly as shown below.

Example Output:

```
  @..@
 (----)
/ >__< \
^^ ~~ ^^
```

Bookcase
Input File: bookcase.txt

In *Bookcase*, a student has been assigned to organize a row of books. The books are to be arranged alphabetically, but they are currently all scrambled. You must check the book titles and organize them alphabetically. A “space” counts as the earliest character, so “`This Test`” should be arranged before “`ThisTest`”.

Input:

The first line contains an integer N. There are N following books, each book being denoted by its title. No two books will have the same title. None of the book titles will contain an `int` (like `1984`).

Output:

You must output the new arrangement of the books in alphabetical order. Each book should be on a new line.

Example Input:

```
7
The Hunger Games
Ready Player One
Animal Farm
A Short History of Nearly Everything
Dracula
A Thousand Splendid Suns
Dune
```

Example Output:

```
A Short History of Nearly Everything
A Thousand Splendid Suns
Animal Farm
Dracula
Dune
Ready Player One
The Hunger Games
```

Line Graph
Input File: line.txt

You are currently learning how to graph lines in Algebra class using the formula y = mx + b. Given the m, x, and b values, your job is to find the y value.

Input:

The first line contains an integer N. The following N lines contain three space-separated integers representing m, x, and b, respectively.

Output:

Output the y value on a separate line for each test case. Your answer will be an integer.

Example Input:

```
3
2 3 4
-10 0 8
5 -5 -6
```

Example Output:

```
10
8
-31
```

Chopping Trees
Input File: chopping.txt

You, a lumberjack, are cutting down a small forest. However, the sun is going down, and you must return home once dark falls. Using the rules given below concerning the chopping time, determine the amount of time it takes for you to cut down your small forest (this amount of time will always be an integer; no rounding necessary).

There are three main components of tree-chopping time in this problem:

1. The amount of time it takes to make one swing (between 1-20 seconds)
2. The number of swings to cut down a tree (between 1-20 swings)
3. The clean up time for each chopped tree (between 1-20 seconds)

Given the number of trees in your small forest (between 1 and 250) and values for the three components above, find the amount of time it takes to completely chop down and clean up the small forest.

Input:

The first line contains an integer N. Each of the following N lines will each contain four space-separated integers, the first being the amount of time it takes to make one swing, the second being the number of swings to cut down a tree, the third being the clean up time for each chopped tree, and the fourth being the number of trees in the forest.

Output:

You will output the amount of time, in seconds, it takes for you to cut down your small forest for each test case. Your result should be a non-rounded integer.

Example Input:

```
3
8 3 2 15
13 14 5 130
3 7 9 76
```

Example Output:

```
390
24310
2280
```

Bunny Island
Input File: bunny.txt

Ōkunoshima, an island in Japan, is home to thousands of bunnies. Due to a lack of predators, the bunny population has grown rapidly, and can be modeled by the Fibonacci Equation. This equation is shown below:

$$F_n = F_{n-1} + F_{n-2}$$

Your job is to find the new number of bunnies based on two starting numbers (F_{n-1} and F_{n-2}) and a period of time. Given that the bunny population is modeled so that *n* increases by **one every month**, use the given data to find the new bunny population in different scenarios.

Input:

The first line contains an integer N. The following N lines contain three space-separated integers representing F_{n-2} ($1 < F_{n-2} < 100$), F_{n-1} ($1 < F_{n-1} < 100$), and a time period in months ($1 < \text{months} < 25$).

Output:

You will output the new bunny population for each of the different scenarios (each answer gets a new line).

Example Input:

```
4
5 8 20
2 3 1
13 21 5
1 2 16
```

Example Output:

```
121393
5
233
4181
```

Hexagons
Input File: None.

Hexagons are the best polygons! Please output the pattern shown below (For convenience of reading, small spaces were inserted between underscores)

The hexagon overlords can see computer screens (egads!), so please ensure the pattern exactly matches the one shown below.

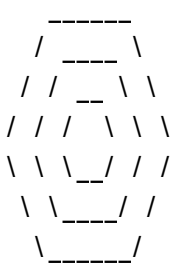
Input:

None.

Output:

You will output the pattern exactly as shown below.

Sample Output:



Anti-RPS
Input File: anti-rps.txt

As a prank, you decide to teach your friend the rules of *Rock, Paper, Scissors*, except that instead of the standard methods of deciding victory, you give them the reversed rules:

Paper beats Scissors
Rock beats Paper
Scissors beats Rock

Given a series of objects, please return the objects that beat them under the above ruleset.

Input:

A string consisting of `R` (rock), `P` (paper), or `S` (scissors).

Output:

A string consisting of the items that would beat the input string.

Sample Input:

```
RPSSPRRRRPS
```

Sample Output:

```
SRPPRSSSSRP
```

Tournament
Input File: tournament.txt

You are organizing a head to head meta-foosball competition. It has the format of a double round robin tournament, in which every entrant plays everyone else twice.

Since each match costs $5 to set up, large tournaments could get costly. Let the total cost of all the rounds be *C*. You will charge the lowest whole-dollar amount to each entrant such that the total amount collected from the entrants is at least half of *C*. Your sponsor will then pay the rest of the set up costs.

Input:

An integer, *n*, the number of lines.
For the next *n* lines:
An integer, a possible number of competitors.

Output:

The amount, in dollars, the sponsor will pay for each number of competitors; one amount per line.

Sample Input:

```
3
5
10
100
```

Sample Output:

```
50
220
24700
```

Artifacts
Input File: artifacts.txt

You are being chased by a horde of hangry raptors! You run through several rooms containing valuable artifacts you want to take. However, since there are raptors chasing you, you can only take one of the two artifacts per room. Each artifact has a value associated with it. You would like to maximize the total value of the artifacts you take.

Input:

An integer *c*, the total number of rooms.
A line with 2**c* space-separated integers, stating the values of the artifacts.

Output:

The largest possible total value of taken artifacts.

Sample Input:

```
4
5 10 17 18 16 16 30 1
```

Sample Output:

```
74
```

Cups and Swaps
Input File: cupsandswaps.txt

Have you ever seen the street performance in which a person places a coin under one of several cups, shuffles them, and invites an audience member to guess the location of the coin? Well, given the cup you want to track, you want to determine after a series of shuffles where the coin ends up.

There are *c* locations for the cups, numbered 1, 2, ... , *c*.

Input:
Three space separated integers, *c*, *t*, and *n*, the number of locations ($2 < c < 16$), the current location number of the cup to track (*t*), and the number of cup swaps ($0 < n \leq 50$).
For the next n lines:
Two space separated integers, *a* and *b*, specifying the locations of the two cups swapped.

Output:
The location of the tracked coin.

Sample Input:

```
8 1 10
1 2
3 2
3 4
5 4
5 6
7 6
1 2
2 1
8 6
7 8
```

Sample Output:

```
8
```

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.

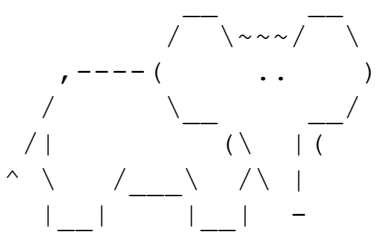
Input:

None.

Output:

Output the elephant exactly as shown below.

Example Output:



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
```

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
```

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
```

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
```

Find Prime
Input File: prime.txt

You are just learning what prime numbers are. In order to help you learn, you want to create a program to print out the first N prime numbers. However, this comes with a twist. To showcase your coding skills, you want to square every number between the integers K1 and K2.

Input:

An integer L, followed by L lines. Each following line will contain three positive integers, N, K1, and K2.

Output:

Print the first N prime numbers. If the prime number is between the values of K1 and K2 (inclusive), square the prime number before printing it.

Example Input:

```
3
8 5 15
3 1 5
26 80 82
```

Example Output:

```
2 3 25 49 121 169 17 19
4 9 25
2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73 79 83 89 97 101
```

Counting Time
Input File: time.txt

All of the clocks you own have broken, so the only way for you to know how much time has passed is by counting each second individually. Given how many seconds have passed by, calculate that time in terms of days, hours, minutes, and seconds to make it easier to get a sense of how long it has been.

Input:

The first line contains an integer N. The following N lines each contain an integer representing the amount of time that has passed in seconds.

Output:

Convert the time passed in seconds to be in terms of days, hours, minutes, and seconds, in the following format: # days # hrs # min # sec.

Example Input:

```
3
122
6734
5917380
```

Example Output:

```
0 days 0 hrs 2 min 2 sec
0 days 1 hrs 52 min 14 sec
68 days 11 hrs 43 min 0 sec
```

Decoder
Input File: decoder.txt

You are sent a message by your company, however, your company's message has to be decoded. The message is all numbers, and each number corresponds to a letter like follows: a = 1, b = 2, c = 3, and so on. Also, a "*SPACE"* = 27, and a period = 28. You must run the message through your program, decode the message, and print the decoded message to the console.

Input:

You are given a random message, where each number is separated by spaces, and the number 27 represents a space in your decoder. The different lines have no meaning. You can assume that the input is valid.

Output:

You should output the decoded message. You do not need to worry about correct capitalization.

Example Input:

```
8 5 12 12 15 27 16 5 18 19 15 14 28 27 27 20 8 9 19 27 9 19 27 20 8 5 27 13 5 19 19 1
7 5 27 9 27 23 1 14 20 27 20 15 27 19 8 1 18 5 27 23 9 20 8 27 25 15 21 28
```

Example Output:

```
hello person.  this is the message i want to share with you.
```

Playground
Input File: playground.txt

Your local elementary school wants to build a new playground. However, they have a limited amount of money, and can only afford a certain amount of fencing needed to surround the playground. Given the price for each foot of fencing and the school's budget, calculate the maximum area of the playground. Note that the playground must be in the shape of a rectangle (four sides, with 90° angles), meaning no circular or triangular playgrounds.

Input:

The first line contains an integer N. The following N lines each contain two positive integers, the first representing the price per feet for fencing and the second representing the school budget.

Output:

Output the maximum area of the playground that can be surrounded by fencing. Your answer should be in square feet and rounded to the nearest integer.

Example Input:

```
3
5 1000
8 100
6 600
```

Example Output:

```
2500
10
625
```

Best Football Team
Input File: football.txt

In your state, there are several school football teams. You want to find which team has the best record. Given the results of the games, list the football teams in the order they appear, followed by their win loss record. Assume there are no ties.

Input:

An integer N, followed by N lines. Each following line represents a game and contains two team names. The first name is the winner of the game and the second name is the loser.

Output:

In the order that the names appear, print their team name, followed by the number of games they won, a forward slash, and the number of games they lost.

Example Input:

```
7
Thunderstrikes Gorillas
Termites Earthquakes
Waves Emperors
Earthquakes Gorillas
Thunderstrikes Waves
Termites Emperors
Thunderstrikes Termites
```

Example Output:

```
Thunderstrikes 3/0
Gorillas 0/2
Termites 2/1
Earthquakes 1/1
Waves 1/1
Emperors 0/2
```

Multiplying Ducks
Input File: ducks.txt

Ducks are the main species that live on Duck Island. The island contains a variety of food for the ducks and no predators, so the ducks grow in population very fast. Every 10 days, the duck population suddenly doubles. Every 24 days, one fourth of the duck population passes away from old age (if these two events occur on the same day, assume the population doubles before any ducks pass away). Given the initial population on January 1st, find the population of ducks on a later date in the same year. Assume that it is not a leap year, meaning February has 28 days, January, March, May, July, August, October, and December have 31 days, and the rest have 30 days.

Input:

The first line contains an integer N. The following N lines each contains an integer P, the initial population, a space, and then a date in the format of mm/dd.

Output:

Output the duck population on the given date.

Example Input:

```
3
10 02/01
55 06/17
1 12/31
```

Example Output:

```
60
641520
918330048
```

Changing a 2D Array
Input File: array2d.txt

You are given a file with a printed 2D array. You must change the array in the following manner:

- If the number is a multiple of 3, you must square that number
- If the number is a multiple of 5, you must take the nth root of the number, where n equals the row number (the first line describing the number of columns/rows does not count as a row number, and the rows start at 1, not 0)
- If the number is a multiple of 3 AND 5, you must divide the number by 5

All your numbers must be rounded to the nearest integer. 0 is **not** considered a multiple of 3 or 5 or both.

Input:

The first line has a number that represents the number of columns, followed by a space, followed by a second number that represents the number of rows. Then the printed array is shown, where each row has integers separated by spaces, and the columns on separate lines.

Output:

You will output the 2D array modified using the changes described above.

Example Input:

```
3 4
5 71 2
8 9 15
2 20 1645
57 40 105
```

Example Output:

```
5 71 2
8 81 3
2 3 12
3249 3 21
```

Alien Message
Input File: alien.txt

You have just received a secret message from an alien planet. However, the message you received is encrypted and currently makes no sense. Fortunately, the aliens have also sent you the instructions to decode the message. Your job is to use the key and instructions provided to decrypt the text, and print out the correct message.

Input:

The first line contains an integer N, which represents the number of messages you must decrypt. The next line will contain two integers separated by a space, indicating the number of rows and columns, respectively, in the matrix message below. This is followed by a matrix of that size. This pattern will repeat N times (representing the number of messages you receive).

Output:

In order to decode the message, you must first divide each element in every row by 7, then subtract **every other element** in every **column** by 19 (first element you subtract, second you do nothing, third element you subtract, etc.). Then, you convert all the numbers to chars, getting the equivalent ASCII character (i.e. 65 corresponds to 'A'), and print out every **column** all on the same line. You do this for each message, and have a new line for each new message.

Example Input:

```
1
7 7
742 357 357 910 854 854 945
707 812 588 700 798 224 735
889 910 840 840 812 602 945
693 224 679 224 763 777 735
910 945 896 693 896 896 910
763 728 805 798 735 784 770
840 840 602 910 903 840 364
```

Example Output:

```
Welcome to the TeamsCode Programming Competition!
```

Pattern Sequence
Input File: pattern.txt

You have created a survey that asks 20 participants to rate their level of happiness in school from 0 to 10. Your goals is to find the longest recurring pattern of values from the sequence of 20 values given by the participants. Your requirements for a recurring pattern is that the pattern of numbers occurs at least twice in sequence of 20 values. Given the sequence of 20 numbers, find how many numbers is in the longest recurring pattern.

Input:

The first line represents the number of lines to follow. Each line contains the sequence of 20 values, ranging from 0 to 10, including a space separating each value.

Output:

Output the longest recurring pattern followed by an integer that represents the length of the longest recurring pattern on the next line in the input of 20 values, with a space separating each value.

Example Input:

```
2
4 5 2 4 5 4 3 9 10 3 2 6 2 7 6 9 4 5 2 4
10 10 10 3 2 1 9 1 2 0 9 2 10 10 10 3 2 1 9 6
```

Example Output:

```
4 5 2 4
4
10 10 10 3 2 1 9
7
```

New Palindrome
Input File: palindrome.txt

You've been given a sentence by your friend, and he has a special task for you. You must determine the minimum number of characters you must remove in order to make that sentence a palindrome. The palindrome is case-sensitive ('S' is not equal to 's'), and spaces do **not** count as characters.

Input:

The first line contains an integer N, which represents the number of sentences you are given (each line is a separate sentence and a new problem). The next N lines each contain one sentence.

Output:

You must output the minimum number of characters you must remove in order to make the sentence a palindrome. You must do this N times (once for every new sentence), and each new answer should be on the next line.

Example Input:

```
3
Redivide
please do not step on the pets there.
The owl Ate my metal worm
```

Example Output:

```
1
15
7
```

Digit Search
Input File: digit.txt

You are given a string consisting of several random letters and digits. However, you are only interested in the digits and want to find the highest digit (up to `9`) that appears in the string that is preceded by all smaller digits. These digits must appear in order, from smallest (`0`) to greatest, but do not have to adjacent to each other. For example, in the string `j012ty43a4p5q7s68`, the correct highest digit would be 6 because it is preceded by 0, 1, 2, 3, 4, and 5 in order. 7 does not count because it isn't preceded by a 6, and therefore 8 also does not count. For each input string, find the highest digit.

Input:

The first line contains an integer N. The following N lines each contain a single string.

Output:

Output the highest digit according to the rules listed above. If there isn't a highest digit, print `None`.

Example Input:

```
3
001om525hi43Woe7ih5f
Qwa01qe2a34Gl56jj87qw89ef
ABC98765432123456789XYZ
```

Example Output:

```
3
9
None
```

Lost Phone
Input File: lostphone.txt

You were going about your daily routine when you suddenly realize you have lost your phone. Using an app on your computer you know that your phone is a distance `d` away, but unfortunately, the app is terrible and does not tell you the actual location of the phone. You know of multiple places you could have lost your phone, and your job is to determine which of these places match the distance `d` which is mentioned by your app. All coordinates given during this problem are based on the Cartesian coordinate system.

Input:

First, you will be given the distance `d`, representing the distance between you and your phone. On the next line, you will be given your position in terms of `(x, y)` coordinates. After that, you will be given a new line containing an integer N, representing the number of following places you could have lost your phone. These next N lines will each contain the name of the location along with its position in `(x, y)` coordinates. `x` and `y` are both single-digit integers (`1` through `9`).

Output:

You will output the name of the location at which you lost your phone, using the same capitalization.

Example Input:

```
5
(5, 6)
4
Library (3, 9)
School (1, 2)
Home (9, 3)
Football Stadium (9, 2)
```

Example Output:

```
Home
```

Intersecting Lines
Input File: intersect.txt

You've now advanced to Algebra II and would like to automate simple Algebra I tasks like finding the point of intersection between two points. Your goal now is to create a program that takes in two equations in the form y = mx + b and prints out the x and y values of the intersection.

Input:

The first line contains an integer N. The following N lines contain two space-separated numbers representing m_1, b_1, m_2, and b_2, respectively.

Output:

Output x and y separated by a space for each test case. Round x and y to the nearest integer. If the lines are parallel and do not intersect, print `None`. If there are an infinite amount of points of intersection (both lines are the same), print `Same Line`.

Example Input:

```
4
2 3 -5 -11
0 3 1 6
2 10 2 -7
17 -1 13 -13
```

Example Output:

```
-2 -1
-3 3
None
-3 -52
```

Roman Numerals
Input File: roman.txt

Your task is to create a program that can convert numbers into Roman numerals. For example, if the input given is 7, your program will produce VII, and if the input is 99, your program will produce XCIX. The table below gives the pairings for different values:

I = 1	V = 5	X = 10	L = 50	C = 100	D = 500	M = 1000

Therefore, you can see that to represent 124 in Roman numerals, you would need one C and two X's to form 120. Then, since the remaining 4 is one less than 5, it is represented as IV, signifying V - I = 5 - 1 = 4. Added together, 124 is represented as CXXIV.

Other special cases include 9 = IX, 40 = XL, 90 = XC, 400 = CD, and so on. In addition, the number 4000 is represented in Roman numerals as MMMM.

Input:

The first line contains an integer N. The following N lines each contain an integer between 1 and 4999.

Output:

Output the Roman numeral for each given input. Note: Use capital i rather than lowercase L to represent 1 in Roman numerals.

Example Input:

```
4
46
19
392
88
```

Example Output:

```
XLVI
XIX
CCCXCII
LXXXVIII
```

Reverse Engineering
Input File: reverse.txt

You are a construction worker who is collecting materials. You have been given an image of the latest shipment; however, the categories of materials have been mixed up. Your job is to determine the shape, size, and substance type of the various materials. There are a few key pieces to this problem:

- There are three different shapes that the materials can have: **linear**, **triangular**, and **rectangular**.

- Area is calculated using `width * height` for linear and rectangular figures and `0.5 * width * height` for triangular figures, **not** by counting the enclosed spaces.

- The linear shapes will always have a height of `1`.

- Triangular shapes have heights represented by the number of lines and widths represented by the number of characters on the last line.

- The top and bottom lines of the triangle will contain an odd number of characters (excluding spaces), and will always resemble an isosceles triangle similar to the structure below:

```
   X
  X X
 X   X
XXXXXXX
```

- The rectangular shapes can have equal or different heights and widths.

- The minimum width is `2`, the maximum width `10`, the minimum height is `1`, the maximum height is `10`.

- There are three different types of materials, symbolized by their corresponding symbols:
    - **Wood:** `W`
    - **Metal:** `M`
    - **Clay:** `C`

Input:

The first line contains an integer N. There are N following “materials,” each material separated by a new line. The size of the material is symbolized by the width/height of the characters, and the type of material is symbolized by the characters (`W`, `M`, `C`) outlining the shape.

Output:

You will output N lines of descriptions, each line containing the shape, the area, and the substance of the material, in that order. Each piece of data (shape, area, and substance) will be separated by a semicolon and a space. The final area will always be an integer.

Example Input:

```
4
WWWWW
W   W
W   W
WWWWW

   M
  M M
 M   M
MMMMMMM

CCCCCCCCCC

MM
MM
```

Example Output:

```
Rectangular; 20; Wood
Triangular; 14; Metal
Linear; 10; Clay
Rectangular; 4; Metal
```

Chess
Input File: chess.txt

You want to play chess with a friend who doesn't know how the pieces move. You want to teach them how the queen moves by placing a white queen on a standard 8-by-8 square chessboard with a number of white pieces on it.

A queen can move to any square on a row, column, or diagonal it's on, as long as the move does not place it on the square of another piece and does not require it to jump over any piece.

Given a board of pieces, you would like to know to how many squares the queen can move.

Input:

An integer, *n*, which is the number of boards.
n boards, which are sets of 8 lines of 8 characters, "0" representing an empty square, "X" representing an occupied square, and "Q" representing the queen, with an empty line separating different boards.

Output:

For each board, print the number of squares the queen can move to.

Example Input:

```
1
00000000
00000000
00X00000
000QX000
00000000
00000000
000X0000
00000000
```

Example Output:

```
18
```

Outlier
Input File: outlier.txt

You have noticed that several problems on your previous tests took longer than expected. The instructions state that the time required for each problem should be close to a nondecreasing arithmetic sequence. What they don't state is how the fourth problem ended up taking more time than the last problem….

You talk to your classmates, who also have had this issue. They have called problems that take more time than they should *outliers*. You would like to identify these outliers.

Input:

Two space separated integers, n and x, specifying number of tests ($0 < n < 10$) and number of problems on each test ($3 < x < 10$).

For the next n lines, there are x space-separated integers, each integer specifying the time it took to complete the problem.

Output:

For each test, print the problem number (one more than the number of problems preceding it) of the outlier.

Sample Input:

```
4 5
1 2 15 4 5
2 2 2 3 2
1 4 3 4 5
10 9 10 11 12
```

Sample Output:

```
3
4
2
1
```

Street Crossing
Input File: streetcrossing.txt

You want to get to Costco to eat free samples of hummus, but there is a busy road to cross.

The road has several lanes, with cars intersecting your straight-line path every now and then.

Each lane takes two seconds to traverse, and you cannot stop in the middle of the street.
Assume that each car passes instantaneously.
You would like to determine the shortest amount of time to wait before crossing the road to avoid getting hit by a car.

Input:

An integer, *n*, the number of data sets.
For each data set:
An integer, *x,* stating the number of lanes.
Lanes in order of crossing: For the next x lines:
Two space-separated integers, *a* and *b*. Cars in the lane will pass every *a* ($2 < a < 60$) seconds, the first car is *b* ($0 \leq b < a$) seconds away from you.

Output:

The minimum time in seconds needed to wait to cross the road, and -1 if it is never possible to cross the road.

Sample Input:

```
1
4
5 2
7 2
9 2
11 2
```

Sample Output:

```
3
```

Trophies
Input File: trophies.txt

You have too many trophies from all the various competitions you won in the past, and you want to store trophies on your trophy stand into a box to create space on the stand. All your trophies are rectangular prisms with the same dimensions, with bases of 1 cm by *l* cm, and you would like to determine the number of ways to fit your trophies in a rectangular box with dimensions *l* cm by *s* cm.

Input:

An integer, *n*, the number of lines.
For the next *n* lines:
Two space-separated integers, *l*, and *s*, the length ($2 < l < 10$) of each trophy (the width and height are 1 cm), and box side length ($1 < s < 100$).

Output:

For each line, print the number of possible ways to arrange the trophies in the box. Two arrangements are different if the final layout of the trophies are different; the order the trophies are placed in the box does not differentiate between arrangements.

Example Input:

```
1
2 4
```

Example Output:

```
5
```

Password
Input File: password.txt

Congratulations! You've forgotten your password for the 71st time!

You only remember that your password consists of a series of strings and was a certain length... oh well. Time to make the next password "bubbleSortIsTheBest1234567890"...

You would like to know how many passwords exist such that each password consists of only strings from a given *allowed* list and is a specified length.

Input:

An integer, *n*, the number of data sets:
For each data set:
Two space-separated integers, *a* and *l*, specifying the length of the allowed list ($0 < a < 6$) and the length of the password ($0 < l < 10$).
a space-separated strings, each a string in the allowed list.

Output:

The number of possible distinct passwords.

Example Input:

```
2
3 9
a pass xyz
4 4
a b c d
```

Example Output:

```
40
256
```

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
```

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.
```

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
```

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
```

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
```

Underground Maze
Input File: maze.txt

You are trapped in an underground labyrinth and need to get above ground fast. You must find your way through an 8x8 maze.

Input:

A number N, representing the number of mazes to follow, followed by mazes with an ‘S’ representing the starting position, an ‘X’ representing the ending position, a ‘#’ representing a wall, and a ‘.’ at every place you are allowed to move to.

Output:

The maze with a space replacing each ‘.’ you must pass in order to get from start to finish. The ‘S’ should be replaced with a space and the ‘X’ should be replaced with the ‘S’.

Example Input:	Example Output:
2 ######## #S.....# ######.# ##X###.# ##.#...# ##.###.# #......# ######## ######## #.....S# ###.#### #...#..# #.##...# #.#..#.# #....#X# ########	######## # # ###### # ##S### # ## #.. # ## ### # #. # ######## ######## #.. # ### #### # #..# # ## # # #. # # # #S# ########

Best Bootcamps
camps.txt

In order to keep learning how to code, you are trying to find different coding bootcamps to attend. To keep track of which camp best, you've created your own scale (the LP scale) to measure how much you're learning. Initially, some camps are better than others, and therefore offer more LP points. However, you plan on attending more than just one camp, and you've also found that going to different camps in a certain order can help boost (or lower) the amount of learning you take in. For example, taking the bootcamp A class and then the bootcamp B class will give you the LP points of A, while giving you a bonus of a certain number of LP points for following up with B. Similarly, if you follow up camp B with another camp C, you might gain or lose another amount of points, depending on whether camp C builds upon or subtracts from B. Your task is to find the best order of classes to take to maximize your LP points.

Input:

The first line contains two integers. The first integer N represents the number of bootcamps there are, and the second integer represents the number of camps you have the budget to attend (assume you will attend as many camps as your budget allows. The next line contains the name of N bootcamps followed by the LP points they offer you for choosing them as your initial bootcamp. The next line contains another integer L. Each of the following L lines contains the name of two bootcamps, followed by an integer representing how many extra points you gain (or lose) from following up with the second bootcamp after taking the first bootcamp.

Output:

Output the bootcamps you attend, in order, followed by the maximum amount of LP points you can gain.

Example Input:

```
5 4
CodeNow 50 SmartCoder 75 BasicOOP 55 BigHack 65 CodingTeam 80
7
BasicOOP SmartCoder 35
CodeNow CodingTeam 15
BigHack BasicOOP -10
SmartCoder BigHack 20
CodingTeam BasicOOP -5
BigHack CodeNow 10
SmartCoder CodingTeam 5
```

Example Output:

```
CodingTeam BasicOOP SmartCoder BigHack 130
```

Road Race
Input File: roadrace.txt

You are competing with your friend to see who can travel across the United States and make it through all the major cities the fastest. Your starting position is in Seattle and your ultimate target is Miami. However, there are no direct paths to Miami; you must travel through other cities on your way there. You must print the cities that you travel through which allows you to take the shortest route from Seattle to Miami.

Input:

An integer N, followed by N lines. Each of the following lines contains a city name, followed by a space and another city name, with an integer afterwards representing the distance between the two cities.

Output:

Print the cities (in order) that you must travel through that allows you to take the shortest route to Miami. Include Seattle at the beginning and Miami at the end.

Example Input:

```
9
Boise Detroit 13
Seattle Sacramento 10
Phoenix Dallas 18
Detroit Sacramento 15
Seattle Portland 5
Miami Detroit 14
Dallas Miami 8
Phoenix Seattle 12
Dallas Boise 7
```

Example Output:

```
Seattle Phoenix Dallas Miami 38
```

Closest ZIP Codes
Input File: zipcode.txt

You have just graduated from college and don't know where you want to live. One of your friends from high school suggests that you buy a house nearby because the prices are very cheap. The problem is, however, that you have a strict policy that you can only move to places where the ZIP code contains the same digits as your current address. Given your current ZIP code, rearrange the numbers so that the new number is as close to your friend's ZIP code as possible (determined by the difference between the two numbers).

Input:

The first line contains an integer N. The following lines each contain two five-digit numbers, the first of which is your current ZIP code and the second is your friend's.

Output:

Output your new rearranged ZIP code that's closest to your friend's ZIP code, followed by the difference of the two numbers.

Example Input:

```
3
45467 75885
98040 56470
29399 30106
```

Example Output:

```
75644 241
49800 6670
29993 113
```

Hiking
Input File: hiking.txt

You are going on a hike on Mount Rainier. Unfortunately, you have an old backpack that only can carry 20 pounds worth of equipment. Your job is to decide which items to bring along to maximize the value of things you are bringing in your backpack.

Input:

An integer N, representing the number of following cases. For each case, there will be five lines, each representing an object. Each line will contain two numbers: the first being the weight of the object and the second being the value of the object.

Output:

The highest total value of all the objects you can put in your backpack without going over the 20 pound weight limit.

Example Input:

```
2
5 6
10 11
3 2
7 4
4 5

12 2
5 12
7 7
4 2
8 9
```

Example Output:

```
22
28
```

Sudoku
Input File: sudoku.txt

You are given a 9x9 sudoku grid. The grid is almost completely solved. Your task is to find the remaining squares that are unsolved to finish the puzzle. To make it easier, there will be at most two empty squares in each of the smaller 3x3 grids.

Input:

The first line contains an integer, which gives you how many numbers are missing. The following nine rows each contain nine numbers separated by a space. The numbers will be between 1 and 9. Blank spaces will be marked with a '?'.

Output:

Output the missing numbers in order of appearance (left to right, then top to bottom).

Example Input:

```
11
2 9 ? 7 4 3 8 6 1
4 ? 1 8 6 5 9 ? 7
8 7 6 1 9 2 5 4 3
3 8 7 4 5 9 2 1 6
6 1 2 3 ? 7 4 ? 5
? 4 9 2 ? 6 7 3 8
? ? 3 5 2 4 1 8 9
9 2 8 6 7 1 ? 5 4
1 5 4 9 3 ? 6 7 2
```

Example Output:

```
5 3 2 8 9 5 1 7 6 3 8
```

Longest Common Subsequence
Input File: longestcs.txt

You are given two strings, both of which are a random jumble of letters. The two strings share a few common letters here and there. Your tasks is to find the longest common subsequence in both strings. This subsequence does not need to be continuous, but must be in order relative to the other common characters.

Input:

An integer N, followed by N lines. Each of the following N lines contains two strings, separated by a space.

Output:

The longest common substring of the two strings. If there are multiple substrings of the same longest length, print the first occurring one.

Example Input:

```
3
sadkspoqe kdkjjsceoe
abcdef fedcba
apldfpaple gijapebpjkele
```

Example Output:

```
dksoe
a (could also be b, c, d, e, or f)
apple
```

Full Bookshelf
Input File: bookshelf.txt

You have a bookshelf in your room that you use to hold all your books. However, as you get older and start to read books with more and more pages, your bookshelf doesn't seem to have enough room anymore. Given a bookshelf with a certain number of rows, each of which a certain width, find out if it is possible to keep all of your books on the bookshelf, and if not, the least amount of books that you must get rid of in order to fit the rest.

Input:

The first row contains an integer N, followed by N sets of input. In each set of input, the first line contains two integers, the first of which is the number of rows your bookshelf has and the second is the width of each row. The next line contains an integer K, and the last line contains K integers, each one representing the width of one of your books.

Output:

For each input set, if you are able to fit all of the books on the bookshelf, output '0'. Otherwise, output the least amount of books that you must get rid of.

Example Input:

```
3
3 20
8
7 8 6 7 9 8 7 6
2 30
11
5 6 4 1 7 12 6 2 9 5 3
4 10
9
15 11 2 6 9 1 13 5 3
```

Example Output:

```
1
0
3
```

Binary Code
Input File: binary.txt

You are passed a string of 0s and 1s. However, this binary string passed to you is incomplete. Given the correct string, your job is to determine whether it is possible to modify the incomplete binary string to make it equal to the correct version using only two modification techniques: adding the characters “10” to the end of the string or moving the last digit of the string to the beginning. For example, you could change “011” to “01110” by adding “10” or change “011” to “101” by moving the last digit to the beginning.

Input:

An integer N, followed by N lines. Each following lines contains two strings: the first is the incomplete string and the second is the complete string.

Output:

Print “Possible” if you can use a combination of adding “10” to the end of the incomplete string and moving the last digit of the incomplete string to the beginning in order to make it match the complete string. Print “Impossible” otherwise.

Example Input:

```
3
1011 1001110
111 011011101
1100 01100110
```

Example Output:

```
Impossible
Possible
Possible
```

Base Palindrome
Input File: palindrome2.txt

You have been given a list of numbers and your boss wants you to determine which of these numbers are palindromes. However, there's a catch - your boss wants you to check if these numbers are palindromes in multiple different bases. Given a list of numbers in base-10, determine a list of in which bases each number is a palindrome (from bases 2-20).

Input:

The first row contains an integer N, followed by N sets of input. In each input, the line contains one positive integer in base-10 format.

Output:

For each line, you must print the different bases in which the number is a palindrome (from bases 2-20). The format will be the different bases separated by commas, with a new line for every new input you are testing. If there is no palindrome for any base, simply print "None". The integer input value will always be less than one billion.

Example Input:

```
5
10
86
717
87
829100
```

Example Output:

```
3, 4, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
6
2, 10, 14
None
16
```

Swap
Input File: swap.txt

The new hit board game *Swap!* has just hit the market. In the game, you are given a rectangular board filled with `X` pieces and `O` pieces. This board, which has unique placement of the `X` and `O` pieces, corresponds to an image of a board with a **different** arrangement of `X` and `O` pieces.

Your job is to make your board match the board in the image in the **fewest possible steps**. However, you can only move the pieces by **swapping** a piece with an adjacent piece (horizontal or vertical adjacent swaps, no diagonal swap). Given your board and the image board, find the fewest number of steps it takes to make the boards match (you can guarantee there is a solution; i.e., each board has the same ratio of `X` to `O` pieces).

Input:

The first line contains two space-separated integers representing the width, w, and height, h, respectively, of the two boards (`1 < width, height < 20`). The next h lines contain your board, followed by another set of h lines containing the image board.

Output:

You will output an integer representing the fewest possible steps to make your board match the image board.

Example Input:

```
5 5
XOOXO
XOXXX
XXOOO
OOOOO
XOXOX
XOXOO
XOXOX
OOOOO
OOOXX
XXOXX
```

Example Output:

```
11
```

Fast Travel
Input File: fasttravel.txt

You are a hiker walking through the wilderness. You are running out of food and supplies and thus must reach the cabin as quickly as you can. There are multiple different paths; however, the various paths are very different (**dirt path**, **river**, **mountainous trail**, and **swamp**). You travel at different speeds in each of these path types; the corresponding speeds are listed below:

- **Dirt Path:** 5 meters/second
- **River:** 1 m/s
- **Mountainous Trail:** 2 m/s
- **Swamp:** 2.5 m/s

Your job is to find the quickest path (the smallest amount of time it takes to travel to the cabin).

Input:

The first line contains an integer N. Each of the following N test cases contain the width, w, and height, h, of the maze on the first line and the maze itself on the following h lines.

You will be given a rectangular maze (`1 < width, height < 20`) in ASCII characters. The maze will be made up of the following symbols:

- `#` – A border to the maze. Impenetrable. The maze border will be surrounded by this material, not including the starting and ending points.
- `P` – The start to the maze (your starting position).
- `C` – The position of the cabin (the end destination).
- `D` – A dirt path.
- `R` – A river.
- `M` – A mountainous trail.
- `S` – A swamp.

Each of the different path types (`D`, `R`, `M`, and `S`) is 10 meters long per character (so `DDD` is 30 meters of dirt path).

Output:

Output the time it takes to travel to each cabin in seconds, calculated by how many `D`, `R`, `M`, and `S` positions you must cross.

Example Input:

```
2
10 7
##C#######
#RRD###S##
#DRR#SSSS#
#SRS#SDRM#
#RMMMSSDD#
#DRMMDSDD#
#######P##
7 5
###C###
#SRD#S#
#S##RM#
#MRDMD#
####P##
```

Example Output:

```
48
42
```

24 Game
Input File: 24game.txt

24 Game is a classic math game where, given four positive integers, you must use addition, subtraction, multiplication, division, and/or parentheses to produce 24 from all four numbers. For example, given 2, 3, 4, and 5, it is possible to produce 24 by doing `2 * (3 + 4 + 5)`.Your job is to write a program that determines whether it is possible to create 24 from four input numbers.

Input:

The first line contains an integer N. The following N lines each contain four space-separated, positive integers.

Output:

If it is possible to form 24 from the numbers using addition, subtraction, multiplication, division, and parentheses, print `Possible`. If not, print `Not Possible`.

Example Input:

```
4
2 5 7 3
7 13 19 2
101 50 7 33
24 10 53 5
```

Example Output:

```
Possible
Not Possible
Not Possible
Possible
```

14. Geometric Sequence
Input File: geometric.txt

A geometric sequence is a sequence where each progressive term is found by multiplying the previous term by a constant value. A geometric sequence could be 1, 2, 4, 8, ..., where the common ratio between is `2`, or it could be 120, 60, 30, 15, ..., where the common ratio is `0.5`. For this problem, you can assume that the common ratio is a positive integer. Your task is to find the length of the longest geometric sequence from a given sorted array of integers. The terms of the sequence do not need to be adjacent to each other.

Input:

The first line contains an integer N. The following N lines can each contain anywhere between 1 and 20 integers, each separated by a space. These integers are guaranteed to be sorted from least to greatest.

Output:

For each test case, output an integer representing the length of the longest geometric sequence that can be formed from the given integers.

Example Input:

```
3
1 2 3 4 5 6 7 8 9 10
2 3 4 5 8 9 10 16 20 27 40 81 120 243
3 5 7 11 23 31 101
```

Example Output:

```
4
5
1
```

Factorization
Input File: factor.txt

Factoring is always a hassle. As a result, you have decided to implement your own program that can find all the real zeros of a linear or quadratic function. This program takes input in the form $c_1x^{p_1} + c_2x^{p_2} + ...$, where c and p are integers and p is between 0 and 2. If any real zeros exist, your job is to print them out.

Input:

The first line contains an integer N. The following N lines each contain an expression in the form shown above.

Output:

Output the zeros for each test case in sorted order. For integer values, do not include trailing zeros, and for decimal values, round down to three decimal places if any are longer than that. If none exist, print `None`.

Example Input:

```
3
1x^1 + 3x^0 - 2x^1
4x^2 + 9x^1 + 5x^0
19x^2 - 6x^0 - 11x^2 - 8x^2
```

Example Output:

```
3
-1.25 -1
None
```

Sequence
Input File: sequence.txt

As part of your job as a worker at a transmissions company, you receive several sequences. Your task is to determine if a sequence received is made of alternating prime numbers and perfect squares. Each valid sequence can start with either a prime or a perfect square, and has at least one of each.

Input:

An integer *x* stating the number of sequences
For each of the next x lines:
A sequence.

Output:

For each sequence, output "True" if the sequence consists of alternating prime numbers and perfect squares and has at least one of each, and "False" if not.

Example Input:

```
3
2424225
64322529
100816449
```

Example Output:

```
True
True
False
```

Tower
Input File: tower.txt

You are attempting to build a tower with several three-dimensional blocks in your collection of rectangular prisms. You want to know the tallest tower you can build, given that each brick must be equal or less in both volume and height than the brick immediately below it. You can choose to not use bricks if desired, and bricks may be oriented in any fashion.

Input:

An integer *x* stating the number of blocks.
For the next *x* lines:
Three space-separated integers, stating the dimensions of the block.

Output:

The height of the tallest constructable tower following the above rules.

Example Input:

```
6
1 1 1
2 2 2
4 10 4
8 6 3
5 6 7
20 20 20
```

Example Output:

```
41
```

Traffic Lights
Input File: trafficlights.txt

You are trying to drive across downtown Landport, from the northwest corner to the southeast corner. Of course, you have terrible traffic light luck, as when you started driving, all the traffic lights turned red. At each intersection, there is a traffic light that alternates between red for a number of minutes between 0 and 9, and green for 1 minute. If a light is red, you cannot pass through it. If a light is green, you can pass through it. Note that the direction a vehicle enters an intersection does not affect its passability. If moving between intersections takes 0.9 minutes, what is the shortest amount of time it would take to reach your destination?

Input:

An integer, n, the number of data sets.
For each data set:
An integer x ($0 \leq x \leq 9$), the size of the x by x grid. There are x^2 intersections.
For the next x lines:
x numbers indicating the length of time each red light takes.

Output:

For each data set, the fewest number of minutes needed to reach the destination, rounded to the nearest tenth.

Example Input:

```
1
4
5290
9681
0751
1080
```

Example Output:

```
10.8
```

Mirrors and Lasers
Input file: mirrorslasers.txt

You arrive at the corner (0, 0) of a two dimensional room, holding a laser pointer. There are mirrors covering the walls, and the sign outside the room says to aim the laser pointer to attempt to hit the sensor in the opposite corner of the room (l, w) in the fewest reflections possible. There are several double sided mirrors in the room, each with non reflective edges (beams are not affected by edges of mirrors). You are thinking of several angles at which to aim the laser pointer and would like to know how many reflections each path would take.

Input:

Four space separated integers, *l, w, m,* and *n,* specifying the length ($1 < l < 100$) and width ($1 < w < 100$) of the room, number of mirrors ($0 < m < 15$), and number of angles to test ($0 < n < 10$).
For the next n lines:
Four space-separated integers, *x1, y1, x2, y2,* specifying the location of the endpoints [(*x1, y1*), (*x2, y2*)] of the double sided mirror.
For the next *m* lines:
Two space separated integers, x and y, specifying the coordinate the laser is initially pointed towards.
All coordinates in the input are guaranteed to be non negative integers.

Output:

For each angle, print the number of reflections the laser will take to reach the sensor, or -1 if the laser does not hit the sensor.

Example Input:

```
10 10 1 3
0 5 4 5
1 1
1 2
8 10
```

Example Output:

```
0
2
-1
```

Battery
Input File: battery.txt

Someone forgot to charge the laptop batteries overnight, and, even worse, most of the chargers are missing! You would like to charge the many laptops as quickly as possible in preparation for the coming day. Each remaining charger delivers energy at a different rate, and each laptop has a different percent battery. Once a laptop is plugged in, it cannot be unplugged until fully charged. Assume an instantaneous switch between laptops after a finished charge, and that a laptop can only be charged with one charger at a time.

Input:

An integer, *n*, the number of data sets.
For each data set:
Two space-separated integers, *c* and *l*, specifying number of chargers ($0 < c < 3$) and number of laptops ($0 < l < 500$).
c space-separated integers, each integer signifying the number of minutes it takes for the charger to charge a laptop battery one percent.
l space-separated integers, each integer (between 0 and 100 inclusive) signifying the current battery percentage of the laptop.

Output:

The shortest amount of time, in minutes, to fully charge all laptops.

Example Input:

```
1
2 6
2 3
92 94 93 95 91 89
```

Example Output:

```
56
```

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
```

Flattened Cube
Input File: cube.txt

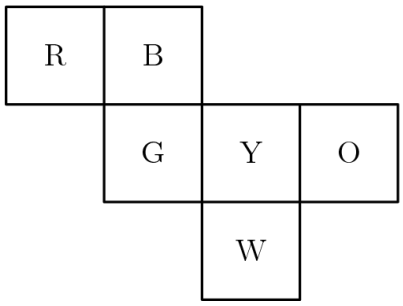
It is possible to make a 3D cube by folding a 2D diagram, such as the one below. Your job for cube is to determine which side is opposite of a given side.

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
```

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
```

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:

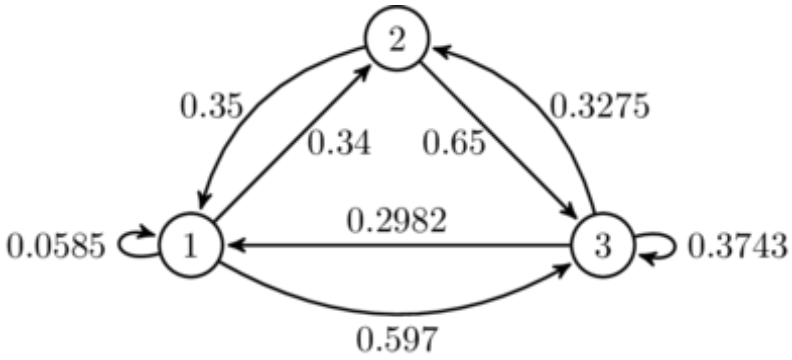
```
3
1 1 1
144
2 6 4
24
6 5 2
122
```

Example Output:

```
6
Target
123
```

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
```