Philaland Coin

Problem Description

The problem solvers have found a new Island for coding and named it as Philaland.

These smart people were given a task to make purchase of items at the Island easier by distributing various coins with different value.

Manish has come up with a solution that if we make coins category starting from $1 till the maximum price of item present on Island, then we can purchase any item easily. He added following example to prove his point.

Lets suppose the maximum price of an item is 5$ then we can make coins of {$1, $2, $3, $4, $5} to purchase any item ranging from $1 till $5.

Now Manisha, being a keen observer suggested that we could actually minimize the number of coins required and gave following distribution {$1, $2, $3}. According to him any item can be purchased one time ranging from $1 to $5. Everyone was impressed with both of them.

Your task is to help Manisha come up with minimum number of denominations for any arbitrary max price in Philaland.

Constraints

1<=T<=100

1<=N<=5000

Input Format

First line contains an integer T denoting the number of test cases.

Next T lines contains an integer N denoting the maximum price of the item present on Philaland.

Output

For each test case print a single line denoting the minimum number of denominations of coins required.

Timeout

1

Test Case

Example 1

Input

2

10

5

Output

4

3

Explanation

For test case 1, N=10.

According to Manish {$1, $2, $3,... $10} must be distributed.

But as per Manisha only {$1, $2, $3, $4} coins are enough to purchase any item ranging from $1 to $10. Hence minimum is 4. Likewise denominations could also be {$1, $2, $3, $5}. Hence answer is still 4.

For test case 2, N=5.

According to Manish {$1, $2, $3, $4, $5} must be distributed.

But as per Manisha only {$1, $2, $3} coins are enough to purchase any item ranging from $1 to $5. Hence minimum is 3. Likewise denominations could also be {$1, $2, $4}. Hence answer is still 3.

## Prime Fibonnaci

### Problem Description

Given two numbers n1 and n2

1. Find prime numbers between n1 and n2, then

2. Make all possible unique combinations of numbers from the prime numbers list you found in step 1.

3. From this new list, again find all prime numbers.

4. Find smallest (a) and largest (b) number from the 2nd generated list, also count of this list.

5. Consider smallest and largest number as the 1st and 2nd number to generate Fibonacci series respectively till the count (number of primes in the 2nd list).

6. Print the last number of a Fibonacci series as an output

### Constraints

2 <= n1, n2 <= 100

n2 - n1 >= 35

### Input Format

One line containing two space separated integers n1 and n2.

### Output

Last number of a generated Fibonacci series.

### Timeout

1

### Test Case

Example 1

Input

2 40

Output

13158006689

Explanation

1st prime list = [2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37]

Combination of all the primes = [23, 25, 27, 211, 213, 217, 219, 223, 229, 231, 32, 35, 37, 311, 313, 319, 323, 329, 331, 337, 52, 53, 57, 511, 513, 517, 519, 523, 529, 531, 537, 72, 73, 75, 711, 713, 717, 719, 723, 729, 731, 737, 112, 113, 115, 117, 1113, 1117, 1119, 1123, 1129, 1131, 1137, 132, 133, 135, 137, 1311, 1317, 1319, 1323, 1329, 1331, 1337, 172, 173, 175, 177, 1711, 1713, 1719, 1723, 1729, 1731, 1737, 192, 193, 195, 197, 1911, 1913, 1917, 1923, 1929, 1931, 1937, 232, 233, 235, 237, 2311, 2313, 2317, 2319, 2329, 2331, 2337, 292, 293, 295, 297, 2911, 2913, 2917, 2919, 2923, 2931, 2937, 312, 315, 317, 3111, 3113, 3117, 3119, 3123, 3129, 3137, 372, 373, 375, 377, 3711, 3713, 3717, 3719, 3723, 3729, 3731]

2nd prime list=[193, 3137, 197, 2311, 3719, 73, 137, 331, 523, 1931, 719, 337, 211, 23, 1117, 223, 1123, 229, 37, 293, 2917, 1319, 1129, 233, 173, 3119, 113, 53, 373, 311, 313, 1913, 1723, 317]

smallest (a) = 23

largest (b) = 3719

Therefore, the last number of a Fibonacci series i.e. 34th Fibonacci number in the series that has 23 and 3719 as the first 2 numbers is 13158006689

Example 2

Input

30 70

Output

2027041

Explanation

1st prime list=[31, 37, 41, 43, 47, 53, 59, 61, 67]

2nd prime list generated form combination of 1st prime list = [3137, 5953, 5347, 6761, 3761, 4337, 6737, 6131, 3767, 4759, 4153, 3167, 4159, 6143]

smallest prime in 2nd list=3137  
largest prime in 2nd list=6761

Therefore, the last number of a Fibonacci series i.e. 14th Fibonacci number in the series that has 3137 and 6761 as the first 2 numbers is 2027041

## Collision Course

### Problem Description

On a busy road, multiple cars are passing by. A simulation is run to see what happens if brakes fail for all cars on the road. The only way for them to be safe is if they don't collide and pass by each other. The goal is to identify whether any of the given cars would collide or pass by each other safely around a Roundabout. Think of this as a reference point O ( Origin with coordinates (0,0) ), but instead of going around it, cars pass through it.

Considering that each car is moving in a straight line towards the origin with individual uniform speed. Cars will continue to travel in that same straight line even after crossing origin. Calculate the number of collisions that will happen in such a scenario.

Note : - Calculate collisions only at origin. Ignore the other collisions. Assume that each car continues on its respective path even after the collision without change of direction or speed for an infinite distance.

### Constraints

1<=C<=10^5

-10^9 <= x,y <= 10^9

0 < v < 10^9.

### Input Format

The first line contains an integer C, denoting the number of cars being considered that are passing by around the origin.

Next C lines contain 3 space delimited values, first two of them being for position coordinates (x,y) in 2D space and the third one for speed (v).

### Output

A single integer Q denoting the number of collisions at origin possible for given set of cars.

### Timeout

1

### Test Case

Example 1

Input

5

5 12 1

16 63 5

-10 24 2

7 24 2

-24 7 2

Output

4

Explanation

Let the 5 cars be A, B, C, D, and E respectively.

4 Collisions are as follows -

1) A & B.

2) A & C.

3) B & C.

4) D & E.

## Television Sets

### Problem Description

Dr. Vishnu is opening a new world class hospital in a small town designed to be the first preference of the patients in the city. Hospital has N rooms of two types - with TV and without TV, with daily rates of R1 and R2 respectively.

However, from his experience Dr. Vishnu knows that the number of patients is not constant throughout the year, instead it follows a pattern. The number of patients on any given day of the year is given by the following formula –

*(6-M)^2 + |D-15|* where

*M is the number of month (1 for jan, 2 for feb ...12 for dec) and*

*D is the date (1,2...31).*

All patients prefer *without TV* rooms as they are cheaper, but will opt for *with TV*rooms only if *without TV* rooms are not available. Hospital has a revenue target for the first year of operation. Given this target and the values of N, R1 and R2 you need to identify the number of TVs the hospital should buy so that it meets the revenue target. Assume the Hospital opens on 1st Jan and year is a non-leap year.

### Constraints

Hospital opens on 1st Jan in an ordinary year

5 <= Number of rooms <= 100

500 <= Room Rates <= 5000

0 <= Target revenue < 90000000

### Input Format

First line provides an integer N that denotes the number of rooms in the hospital

Second line provides two space-delimited integers that denote the rates of rooms with TV (R1) and without TV (R2) respectively

Third line provides the revenue target

### Output

Minimum number of TVs the hospital needs to buy to meet its revenue target. If it cannot achieve its target, print the total number of rooms in the hospital.

### Timeout

1

### Test Case

Example 1

Input

20

1500 1000

7000000

Output

14

Explanation

Using the formula, number of patients on 1st Jan will be 39, on 2nd Jan will be 38 and so on. Considering there are only twenty rooms and rates of both type of rooms are 1500 and 1000 respectively, we will need 14 TV sets to get revenue of 7119500. With 13 TV sets Total revenue will be less than 7000000

Example 2

Input

10

1000 1500

10000000

Output

10

Explanation

In the above example, the target will not be achieved, even by equipping all the rooms with TV. Hence, the answer is 10 i.e. total number of rooms in the hospital.

## Lazy Student

### Problem Description

There is a test of Algorithms. Teacher provides a question bank consisting of N questions and guarantees all the questions in the test will be from this question bank. Due to lack of time and his laziness, Codu could only practice M questions. There are T questions in a question paper selected randomly. Passing criteria is solving at least 1 of the T problems. Codu can't solve the question he didn't practice. What is the probability that Codu will pass the test?

### Constraints

0 < T <= 10000

0 < N, T <= 1000

0 <= M <= 1000

M,T <= N

### Input Format

First line contains single integer T denoting the number of test cases.

First line of each test case contains 3 integers separated by space denoting N, T, and M.

### Output

For each test case, print a single integer.

If probability is p/q where p & q are co-prime, print (p\*mulInv(q)) modulo 1000000007, where mulInv(x) is multiplicative inverse of x under modulo 1000000007.

### Timeout

1

### Test Case

Example 1

Input

1

4 2 1

Output

500000004

Explanation

The probability is ½. So output is 500000004.

## Lifeguard Prob

### Problem Description

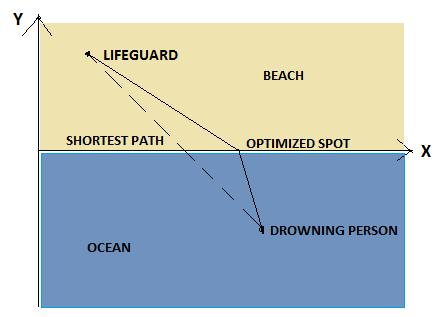
A life guard is sitting on a beach on a lookout for potential emergencies.

He suddenly notices a person who is drowning and springs to action.

He runs up to the sea with a speed f\*V km/hr, then he swims straight to the person at the rate V km/hr (both in straight lines and where f is a multiplying factor as humans run much faster than they can swim).

He wants to minimize the time taken to get to that person.

See the below image for better understanding :



Since the lifeguard runs faster, it will save some more time to run a longer distance rather than going straight and thus swimming a long distance.

However, this comes with the trade-off that running longer can actually mean going a longer distance thus taking more time.

Thus, it can be logically inferred that, there must exist a spot on the Beach-Sea Interface where, if the lifeguard directly runs to from his starting location, and then swims directly to the drowning person, it'll take the least time.

Given the starting location , the location of the drowning person and the multiplying factor f, find the optimized spot for fastest time.

Assumptions/Problem Explanation:

1. Consider that everything is in a two dimensional (2D) plane.

The x axis represents the Beach-Sea interface, positive Y axis is towards land and negative Y-axis towards sea (See image above).

2. The Y-axis along with origin is at some arbitrary location to the left of both the lifeguard and the drowning person. Since the origin point remains the same for both of them and the staring locations are given relative to the origin , its actual location does not matter. The only thing to note is, the origin and Y axis is to the left of both of them, so beach is always in 1st quadrant and sea in 4th. Thus, the positions of lifeguard and the drowning person are given as their (x,y) co-ordinates. (7,5) means the person is 7 units on the axis and 5 units on the positive y axis, and hence on the beach. Similarly, (7,-5) means the person is 7 units on the axis and 5 units on the negative Y axis, and hence in sea.

3. The lifeguard both runs and swims in perfectly straight lines.

4. With regards to everything explained above, your task is to find a point on the Beach-Sea Interface (X -axis) (x\_optimized,0) to where if the lifeguard runs directly from his starting position and then swims directly from the point to the drowning person, it'll take the least amount of time.

5. All calculations must be done upto 6 decimal points accuracy and the output must be upto 6 decimal points as well.

### Constraints

0 <= x\_l < 100 (Integer)

0 <= y\_l < 100 (Integer)

0 <= x\_w < 100 (Integer)

-500 < y\_w < 0 (Integer)

1 < f <= 15 (Integer)

### Input Format

The input shall consist of 3 parameters :

1. Starting position of the lifeguard in terms of his coordinates (x\_l,y\_l).

2. Position of the drowning person (x\_w,y\_w)

3. The multiplying factor f.

These parameters would be given in the following order in 5 different lines:

x\_l

y\_l

x\_w

y\_w

f

### Output

Output must be a single number, x\_optimized, as described above. The output must be having accuracy to 6 decimal places. That is, 1 should be represented as 1.000000

### Timeout

1

### Test Case

Example 1

Input

1

1

1

-1

1.2

Output

1.000000

Swayamvar

Problem Description

A ceremony where a Bride chooses her Groom from an array of eligible bachelors is called Swayamvar. But this is a Swayamvar with difference. An array of Bride-to-be will choose from an array of Groom-to-be.

The arrangement at this Swayamvar is as follows

· Brides-to-be are organized such that the most eligible bachelorette will get first chance to choose her Groom. Only then, the next most eligible bachelorette will get a chance to choose her Groom

· If the initial most eligible bachelorette does not get a Groom of her choice, none of the Brides-to-be have any chance at all to get married. So unless a senior bachelorette is out of the “queue”, the junior bachelorette does not get a chance to select her Groom-to-be

· Inital state of Grooms-to-be is such that most eligible bachelor is at the head of the “queue”. The next most eligible bachelor is next in the queue. So on and so forth.

· Now everything hinges on the choice of the bachelorette. The most eligible bachelorette will now meet the most eligible bachelor.

· If bachelorette selects the bachelor, both, the bachelorette and the bachelor are now Bride and Groom respectively and will no longer be a part of the Swayamvar activity. Now, the next most eligible bachelorette will get a chance to choose her Groom. Her first option is the next most eligible bachelor (relative to initial state)

· If the most eligible bachelorette passes the most eligible bachelor, the most eligible bachelor now moves to the end of the queue. The next most eligible bachelor is now considered by the most eligible bachelorette. Selection or Passing over will have the same consequences as explained earlier.

· If most eligible bachelorette reaches the end of bachelor queue, then the Swayamvar is over. Nobody can get married.

· Given a mix of Selection or Passing over, different pairs will get formed.

The selection criteria is as follows

1. Each person either drinks rum or mojito. Bride will choose groom only if he drinks the same drink as her.

Note : There are equal number of brides and grooms for the swayamvar.

Tyrion as the hand of the king wants to know how many pairs will be left unmatched. Can you help Tyrion?

Constraints

1<= N <= 10^4

Input Format

First line contains one integer N, which denotes the number of brides and grooms taking part in the swayamvar. Second line contains a string in lowercase, of length N containing initial state of brides-to-be. Third line contains a string in lowercase, of length N containing initial state of grooms-to-be. Each string contains only lowercase 'r' and 'm' stating person at that index drinks "rum"(for 'r') or mojito(for 'm').

Output

Output a single integer denoting the number of pairs left unmatched.

Timeout

1

Explanation

Example 1

Input

4

rrmm mrmr

Output

0

Explanation

The bride at first place will only marry groom who drinks rum. So the groom at first place will join the end of the queue. Updated groom's queue is "rmrm".

Now the bride at first place will marry the groom at first place. Updated bride's queue is "rmm" and groom's queue is "mrm".

The process continues and at last there are no pairs left. So answer is 0.

Example 2

Input

4 rmrm mmmr

Output

2

Explanation

Following the above process 2 pairs will be left unmatched. Remember that bride will not move until she gets a groom of her choice.

Digit Pairs

Problem Description

Given N three-digit numbers, your task is to find bit score of all N numbers and then print the number of pairs possible based on these calculated bit score.

1. Rule for calculating bit score from three digit number:

From the 3-digit number,

· extract largest digit and multiply by 11 then

· extract smallest digit multiply by 7 then

· add both the result for getting bit pairs.

Note: - Bit score should be of 2-digits, if above results in a 3-digit bit score, simply ignore most significant digit.

Consider following examples:

Say, number is 286

Largest digit is 8 and smallest digit is 2

So, 8\*11+2\*7 =102 so ignore most significant bit , So bit score = 02.

Say, Number is 123

Largest digit is 3 and smallest digit is 1

So, 3\*11+7\*1=40, so bit score is 40.

2. Rules for making pairs from above calculated bit scores

Condition for making pairs are

· Both bit scores should be in either odd position or even position to be eligible to form a pair.

· Pairs can be only made if most significant digit are same and at most two pair can be made for a given significant digit.

Constraints

N<=500

Input Format

First line contains an integer N, denoting the count of numbers.

Second line contains N 3-digit integers delimited by space

Output

One integer value denoting the number of bit pairs.

Timeout

1

Explanation

Example 1

Input

8 234 567 321 345 123 110 767 111

Output

3

Explanation

After getting the most and least significant digits of the numbers and applying the formula given in Rule 1 we get the bit scores of the numbers as:

58 12 40 76 40 11 19 18

No. of pair possible are 3:

40 appears twice at odd-indices 3 and 5 respectively. Hence, this is one pair.

12, 11, 18 are at even-indices. Hence, two pairs are possible from these three-bit scores.

Hence total pairs possible is 3

Dole Out Cadbury

Problem Description

You are a teacher in reputed school. During Celebration Day you were assigned a task to distribute Cadbury such that maximum children get the chocolate. You have a box full of Cadbury with different width and height. You can only distribute largest square shape Cadbury. So if you have a Cadbury of length 10 and width 5, then you need to break Cadbury in 5X5 square and distribute to first child and then remaining 5X5 to next in queue

Constraints

0<P<Q<1501

0<R<S<1501

Input Format

First line contains an integer P that denotes minimum length of Cadbury in the box

Second line contains an integer Q that denotes maximum length of Cadbury in the box

Third line contains an integer R that denotes minimum width of Cadbury in the box

Fourth line contains an integer S that denotes maximum width of Cadbury in the box

Output

Print total number of children who will get chocolate.

Timeout

1

Explanation

Example 1

Input

5

7

3

4

Output

24

Explanation

Length is in between 5 to 7 and width is in between 3 to 4.

So we have 5X3,5X4,6X3,6X4,7X3,7X4 type of Cadbury in the box.

If we take 5X3 :

First, we can give 3X3 square Cadbury to 1st child .Then we are left with 3X2. Now largest square is 2X2 which will be given to next child. Next, we are left with two 1X1 part of Cadbury which will be given to another two children.

And so on

Petrol Pump

Problem Description

A big group of students, starting a long journey on different set of vehicles need to fill petrol in their vehicles.

As group leader you are required to minimize the time they spend at the petrol pump to start the journey at the earliest. You will be given the quantity of petrol (in litres) that can be filled in each vehicle. There are two petrol vending machines at the petrol pump. You need to arrange the vehicles in such a way that they take shortest possible time to fill all the vehicles and provide the time taken in seconds as output. Machine vends petrol @ 1litre/second.

Assume that there is no time lost between switching vehicles to start filling petrol.

Constraints

1<= Number of vehicles < 50.

0 <= Quantity of petrol required in any vehicle <= 200

Input Format

First line will provide the quantity of petrol (separated by space) that can be filled in each vehicle.

Output

Shortest possible time to fill petrol in all the vehicles.

Timeout

1

Explanation

Example 1

Input

1 2 3 4 5 10 11 3 6 16

Output

31

Explanation

First Petrol vending machine will cater to vehicles taking - 16, 6, 4, 3, 2 litres of petrol (Total 31 sec)

Second machine will cater to vehicles taking - 11, 10, 5, 3, 1 litres of petrol (Total 30 sec)

Example 2

Input

25 30 35 20 90 110 45 70 80 12 30 35 85

Output

335

Explanation

First Petrol vending machine will cater to vehicles taking - 80, 45, 35, 30, 25, 12, 85, 20 litres of petrol.

Second machine will cater to vehicles taking - 90, 70, 35, 30, 110 litres of petrol. Since second machine will take more time, total time to fill petrol in all vehicles will be 335 seconds.

Grooving Monkeys

Problem Description

N monkeys are invited to a party where they start dancing. They dance in a circular formation, very similar to a Gujarati Garba or a Drum Circle. The dance requires the monkeys to constantly change positions after every 1 second.

The change of position is not random & you, in the audience, observe a pattern. Monkeys are very disciplined & follow a specific pattern while dancing.

Consider N = 6, and an array monkeys = {3,6,5,4,1,2}.

This array (1-indexed) is the dancing pattern. The value at monkeys[i], indicates the new of position of the monkey who is standing at the ith position.

Given N & the array monkeys[ ], find the time after which all monkeys are in the initial positions for the 1st time.

Constraints

1<=t<=10 (test cases)

1<=N<=10000 (Number of monkeys)

Input Format

First line contains single integer t, denoting the number of test cases.

Each test case is as follows -

Integer N denoting the number of monkeys.

Next line contains N integer denoting the dancing pattern array, monkeys[].

Output

t lines,

Each line must contain a single integer T, where T is the minimum number of seconds after which all the monkeys are in their initial position.

Timeout

1

Explanation

Example 1

Input

1

6

3 6 5 4 1 2

Output

6

Explanation

Consider N = 6, and an array monkeys = {3,6,5,4,1,2}.

Suppose monkeys are a,b,c,d,e,f, & Initial position (at t = 0) -> a,b,c,d,e,f

At t = 1 -> e,f,a,d,c,b

a will move to 3rd position, b will move to 6th position, c will move to 5th position, d will move to 4th position, e will move to 1st position and f will move to 2nd position. Thus from a,b,c,d,e,f at t =0, we get e,f,a,d,c,b at t =1. Recursively applying same transpositions, we get following positions for different values of t.

At t = 2 -> c,b,e,d,a,f

At t = 3 -> a,f,c,d,e,b

At t = 4 -> e,b,a,d,c,f

At t = 5 -> c,f,e,d,a,b

At t = 6 -> a,b,c,d,e,f

Since at t = 6, we got the original position, therefore the answer is 6.

Death Battle

Problem Description

In a crossover fantasy universe, Houin Kyoma is up in a battle against a powerful monster Nomu that can kill him in a single blow. However being a brilliant scientist Kyoma found a way to pause time for exactly M seconds. Each second, Kyoma attacks Nomu with certain power, which will reduce his health points by that exact power. Initially Nomu has H Health Points. Nomu dies when his Health Points reach 0. Normally Kyoma performs Normal Attack with power A. Besides from Kyoma’s brilliance, luck plays a major role in events of this universe. Kyoma’s Luck L is defined as probability of performing a super attack. A super attack increases power of Normal Attack by C. Given this information calculate and print the probability that Kyoma kills Nomu and survives. If Kyoma dies print “RIP”.

Constraints

0 < T <= 50

1 <= A, H, C, L1, L2 <= 1000

1 <= M <= 20.

L1<=L2

Input Format

First line is integer T denoting number of test cases.

Each test case consist of single line with space separated numbers A H L1 L2 M C. Where luck L is defined as L1/L2. Other numbers are, as described above.

Output

Print probability that Kyoma kills Nomu in form P1/P2 where P1<=P2 and gcd(P1,P2)=1. If impossible, print “RIP” without quotes.

Timeout

1

Explanation

Example 1

Input

2

10 33 7 10 3 2

10 999 7 10 3 2

Output

98/125

RIP