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