Codeforce questions of club in IIT Indore

A. MEX Operation

time limit per test

1 s.

memory limit per test

256 MB

input

standard input

output

standard output

A binary string is a string that consists of characters 00 and 11.

Let MEXMEX of a binary string be the smallest digit among 00, 11, or 22 that does not occur in the string. For example, MEXMEX of 001011001011 is 22, because 00 and 11 occur in the string at least once, MEXMEX of 11111111 is 00, because 00 and 22 do not occur in the string and 0<20<2.

A binary string ss is given. You should cut it into any number of substrings such that each character is in exactly one substring. It is possible to cut the string into a single substring — the whole string.

A string aa is a substring of a string bb if aa can be obtained from bb by deletion of several (possibly, zero or all) characters from the beginning and several (possibly, zero or all) characters from the end.

What is the **minimal** sum of MEXMEX of all substrings pieces can be?

**Input**

The input consists of multiple test cases. The first line contains a single integer tt (1≤t≤1041≤t≤104) — the number of test cases. Description of the test cases follows.

Each test case contains a single binary string ss (1≤|s|≤1051≤|s|≤105).

It's guaranteed that the sum of lengths of ss over all test cases does not exceed 105105.

**Output**

For each test case print a single integer — the minimal sum of MEXMEX of all substrings that it is possible to get by cutting ss optimally.

**Example**

**input**

**Copy**

6

01

1111

01100

101

0000

01010

**output**

**Copy**

1

0

2

1

1

2

**Note**

In the first test case the minimal sum is MEX(0)+MEX(1)=1+0=1MEX⁡(0)+MEX⁡(1)=1+0=1.

In the second test case the minimal sum is MEX(1111)=0MEX⁡(1111)=0.

In the third test case the minimal sum is MEX(01100)=2MEX⁡(01100)=2.

B. Find XOR

time limit per test

1 s.

memory limit per test

256 MB

input

standard input

output

standard output

*In order to celebrate Twice's 5th anniversary, Tzuyu and Sana decided to play a game.*

Tzuyu gave Sana two integers aa and bb and a really important quest.

In order to complete the quest, Sana has to output the smallest possible value of (a⊕xa⊕x) + (b⊕xb⊕x) for any given xx, where ⊕⊕ denotes the [bitwise XOR operation](http://en.wikipedia.org/wiki/Bitwise_operation#XOR).

**Input**

Each test contains multiple test cases. The first line contains the number of test cases tt (1≤t≤1041≤t≤104). Description of the test cases follows.

The only line of each test case contains two integers aa and bb (1≤a,b≤1091≤a,b≤109).

**Output**

For each testcase, output the smallest possible value of the given expression.

**Example**

**input**

**Copy**

6

6 12

4 9

59 832

28 14

4925 2912

1 1

**output**

**Copy**

10

13

891

18

6237

0

**Note**

For the first test case Sana can choose x=4x=4 and the value will be (6⊕46⊕4) + (12⊕412⊕4) = 2+82+8 = 1010. It can be shown that this is the smallest possible value.

C. Gregor and the Enemy

time limit per test

1 s.

memory limit per test

256 MB

input

standard input

output

standard output

There is a chessboard of size nn by nn. The square in the ii-th row from top and jj-th column from the left is labelled (i,j)(i,j).

Currently, Gregor has some pawns in the nn-th row. There are also enemy pawns in the 11-st row. On one turn, Gregor moves one of **his** pawns. A pawn can move one square up (from (i,j)(i,j) to (i−1,j)(i−1,j)) if there is no pawn in the destination square. Additionally, a pawn can move one square diagonally up (from (i,j)(i,j) to either (i−1,j−1)(i−1,j−1) or (i−1,j+1)(i−1,j+1)) if and only if there is an enemy pawn in that square. The enemy pawn is also removed.

Gregor wants to know what is the maximum number of his pawns that can reach row 11?

Note that only Gregor takes turns in this game, and **the enemy pawns never move**. Also, when Gregor's pawn reaches row 11, it is stuck and cannot make any further moves.

**Input**

The first line of the input contains one integer tt (1≤t≤2⋅1041≤t≤2⋅104) — the number of test cases. Then tt test cases follow.

Each test case consists of three lines. The first line contains a single integer nn (2≤n≤2⋅1052≤n≤2⋅105) — the size of the chessboard.

The second line consists of a string of binary digits of length nn, where a 11 in the ii-th position corresponds to an enemy pawn in the ii-th cell from the left, and 00 corresponds to an empty cell.

The third line consists of a string of binary digits of length nn, where a 11 in the ii-th position corresponds to a Gregor's pawn in the ii-th cell from the left, and 00 corresponds to an empty cell.

It is guaranteed that the sum of nn across all test cases is less than 2⋅1052⋅105.

**Output**

For each test case, print one integer: the **maximum** number of Gregor's pawns which can reach the 11-st row.

**Example**

**input**

**Copy**

4

3

000

111

4

1111

1111

3

010

010

5

11001

00000

**output**

**Copy**

3

4

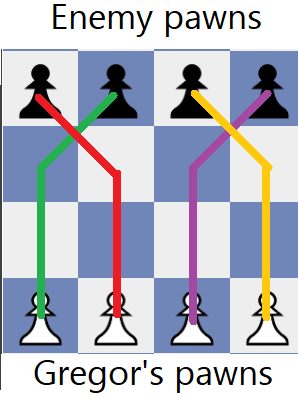
0

0

**Note**

In the first example, Gregor can simply advance all 33 of his pawns forward. Thus, the answer is 33.

In the second example, Gregor can guarantee that all 44 of his pawns reach the enemy row, by following the colored paths as demonstrated in the diagram below. Remember, only Gregor takes turns in this "game"!



In the third example, Gregor's only pawn is stuck behind the enemy pawn, and cannot reach the end.

In the fourth example, Gregor has no pawns, so the answer is clearly 00.

D. Checking Divisibility

time limit per test

1 s.

memory limit per test

256 MB

input

standard input

output

standard output

You are given an integer nn.

You can perform any of the following operations with this number an arbitrary (possibly, zero) number of times:

1. Replace nn with n2n2 if nn is divisible by 22;
2. Replace nn with 2n32n3 if nn is divisible by 33;
3. Replace nn with 4n54n5 if nn is divisible by 55.

For example, you can replace 3030 with 1515 using the first operation, with 2020 using the second operation or with 2424 using the third operation.

Your task is to find the minimum number of moves required to obtain 11 from nn or say that it is impossible to do it.

You have to answer qq independent queries.

**Input**

The first line of the input contains one integer qq (1≤q≤10001≤q≤1000) — the number of queries.

The next qq lines contain the queries. For each query you are given the integer number nn (1≤n≤10181≤n≤1018).

**Output**

Print the answer for each query on a new line. If it is impossible to obtain 11 from nn, print -1. Otherwise, print the minimum number of moves required to do it.

**Example**

**input**

**Copy**

7

1

10

25

30

14

27

1000000000000000000

**output**

**Copy**

0

4

6

6

-1

6

72

D. Checking Divisibility

time limit per test

1 s.

memory limit per test

256 MB

input

standard input

output

standard output

You are given an integer nn.

You can perform any of the following operations with this number an arbitrary (possibly, zero) number of times:

1. Replace nn with n2n2 if nn is divisible by 22;
2. Replace nn with 2n32n3 if nn is divisible by 33;
3. Replace nn with 4n54n5 if nn is divisible by 55.

For example, you can replace 3030 with 1515 using the first operation, with 2020 using the second operation or with 2424 using the third operation.

Your task is to find the minimum number of moves required to obtain 11 from nn or say that it is impossible to do it.

You have to answer qq independent queries.

**Input**

The first line of the input contains one integer qq (1≤q≤10001≤q≤1000) — the number of queries.

The next qq lines contain the queries. For each query you are given the integer number nn (1≤n≤10181≤n≤1018).

**Output**

Print the answer for each query on a new line. If it is impossible to obtain 11 from nn, print -1. Otherwise, print the minimum number of moves required to do it.

**Example**

**input**

**Copy**

7

1

10

25

30

14

27

1000000000000000000

**output**

**Copy**

0

4

6

6

-1

6

72

E. Metro Stations

time limit per test

1 s.

memory limit per test

256 MB

input

standard input

output

standard output

The circle line of the Roflanpolis subway has nn stations.

There are two parallel routes in the subway. The first one visits stations in order 1→2→…→n→1→2→…1→2→…→n→1→2→… (so the next stop after station xx is equal to (x+1)(x+1) if x<nx<n and 11 otherwise). The second route visits stations in order n→(n−1)→…→1→n→(n−1)→…n→(n−1)→…→1→n→(n−1)→… (so the next stop after station xx is equal to (x−1)(x−1) if x>1x>1 and nn otherwise). All trains depart their stations simultaneously, and it takes exactly 11 minute to arrive at the next station.

Two toads live in this city, their names are Daniel and Vlad.

Daniel is currently in a train of the **first** route at station aa and will exit the subway when his train reaches station xx.

Coincidentally, Vlad is currently in a train of the **second** route at station bb and he will exit the subway when his train reaches station yy.

Surprisingly, all numbers a,x,b,ya,x,b,y are distinct.

Toad Ilya asks you to check if Daniel and Vlad will ever be at the same station at the same time during their journey. In other words, check if there is a moment when their trains stop at the same station. Note that this includes the moments when Daniel or Vlad enter or leave the subway.

**Input**

The first line contains five space-separated integers nn, aa, xx, bb, yy (4≤n≤1004≤n≤100, 1≤a,x,b,y≤n1≤a,x,b,y≤n, all numbers among aa, xx, bb, yy are distinct) — the number of stations in Roflanpolis, Daniel's start station, Daniel's finish station, Vlad's start station and Vlad's finish station, respectively.

**Output**

Output "YES" if there is a time moment when Vlad and Daniel are at the same station, and "NO" otherwise. You can print each letter in any case (upper or lower).

**Examples**

**input**

**Copy**

5 1 4 3 2

**output**

**Copy**

YES

**input**

**Copy**

10 2 1 9 10

**output**

**Copy**

NO

**Note**

In the first example, Daniel and Vlad start at the stations (1,3)(1,3). One minute later they are at stations (2,2)(2,2). They are at the same station at this moment. Note that Vlad leaves the subway right after that.

Consider the second example, let's look at the stations Vlad and Daniel are at. They are:

* initially (2,9)(2,9),
* after 11 minute (3,8)(3,8),
* after 22 minutes (4,7)(4,7),
* after 33 minutes (5,6)(5,6),
* after 44 minutes (6,5)(6,5),
* after 55 minutes (7,4)(7,4),
* after 66 minutes (8,3)(8,3),
* after 77 minutes (9,2)(9,2),
* after 88 minutes (10,1)(10,1),
* after 99 minutes (1,10)(1,10).

After that, they both leave the subway because they are at their finish stations, so there is no moment when they both are at the same station.

F. Deletion Operation

time limit per test

2 s.

memory limit per test

256 MB

input

standard input

output

standard output

Polycarp has an array aa consisting of nn integers.

He wants to play a game with this array. The game consists of several moves. On the first move he chooses any element and deletes it (after the first move the array contains n−1n−1 elements). For each of the next moves he chooses any element with the only restriction: its parity should differ from the parity of the element deleted on the previous move. In other words, he alternates parities (even-odd-even-odd-... or odd-even-odd-even-...) of the removed elements. Polycarp stops if he can't make a move.

Formally:

* If it is the first move, he chooses any element and deletes it;
* If it is the second or any next move:
  + if the last deleted element was **odd**, Polycarp chooses any **even** element and deletes it;
  + if the last deleted element was **even**, Polycarp chooses any **odd** element and deletes it.
* If after some move Polycarp cannot make a move, the game ends.

Polycarp's goal is to **minimize** the sum of **non-deleted** elements of the array after end of the game. If Polycarp can delete the whole array, then the sum of **non-deleted** elements is zero.

Help Polycarp find this value.

**Input**

The first line of the input contains one integer nn (1≤n≤20001≤n≤2000) — the number of elements of aa.

The second line of the input contains nn integers a1,a2,…,ana1,a2,…,an (0≤ai≤1060≤ai≤106), where aiai is the ii-th element of aa.

**Output**

Print one integer — the **minimum** possible sum of **non-deleted** elements of the array after end of the game.

**Examples**

**input**

**Copy**

5

1 5 7 8 2

**output**

**Copy**

0

**input**

**Copy**

6

5 1 2 4 6 3

**output**

**Copy**

0

**input**

**Copy**

2

1000000 1000000

**output**

**Copy**

1000000

F. Deletion Operation

time limit per test

2 s.

memory limit per test

256 MB

input

standard input

output

standard output

Polycarp has an array aa consisting of nn integers.

He wants to play a game with this array. The game consists of several moves. On the first move he chooses any element and deletes it (after the first move the array contains n−1n−1 elements). For each of the next moves he chooses any element with the only restriction: its parity should differ from the parity of the element deleted on the previous move. In other words, he alternates parities (even-odd-even-odd-... or odd-even-odd-even-...) of the removed elements. Polycarp stops if he can't make a move.

Formally:

* If it is the first move, he chooses any element and deletes it;
* If it is the second or any next move:
  + if the last deleted element was **odd**, Polycarp chooses any **even** element and deletes it;
  + if the last deleted element was **even**, Polycarp chooses any **odd** element and deletes it.
* If after some move Polycarp cannot make a move, the game ends.

Polycarp's goal is to **minimize** the sum of **non-deleted** elements of the array after end of the game. If Polycarp can delete the whole array, then the sum of **non-deleted** elements is zero.

Help Polycarp find this value.

**Input**

The first line of the input contains one integer nn (1≤n≤20001≤n≤2000) — the number of elements of aa.

The second line of the input contains nn integers a1,a2,…,ana1,a2,…,an (0≤ai≤1060≤ai≤106), where aiai is the ii-th element of aa.

**Output**

Print one integer — the **minimum** possible sum of **non-deleted** elements of the array after end of the game.

**Examples**

**input**

**Copy**

5

1 5 7 8 2

**output**

**Copy**

0

**input**

**Copy**

6

5 1 2 4 6 3

**output**

**Copy**

0

**input**

**Copy**

2

1000000 1000000

**output**

**Copy**

1000000

G. Hitagi's Sequence

time limit per test

1 s.

memory limit per test

256 MB

input

standard input

output

standard output

A few years ago, Hitagi encountered a giant crab, who stole the whole of her body weight. Ever since, she tried to avoid contact with others, for fear that this secret might be noticed.

To get rid of the oddity and recover her weight, a special integer sequence is needed. Hitagi's sequence has been broken for a long time, but now Kaiki provides an opportunity.

Hitagi's sequence aa has a length of nn. Lost elements in it are denoted by zeros. Kaiki provides another sequence bb, whose length kk equals the number of lost elements in aa (i.e. the number of zeros). Hitagi is to replace each zero in aa with an element from bb so that **each element in**bb**should be used exactly once**. Hitagi knows, however, that, **apart from**00**, no integer occurs in**aa**and**bb**more than once in total.**

If the resulting sequence is **not** an increasing sequence, then it has the power to recover Hitagi from the oddity. You are to determine whether this is possible, or Kaiki's sequence is just another fake. In other words, you should detect whether it is possible to replace each zero in aa with an integer from bb so that each integer from bb is used exactly once, and the resulting sequence is **not** increasing.

**Input**

The first line of input contains two space-separated positive integers nn (2≤n≤1002≤n≤100) and kk (1≤k≤n1≤k≤n) — the lengths of sequence aa and bb respectively.

The second line contains nn space-separated integers a1,a2,…,ana1,a2,…,an (0≤ai≤2000≤ai≤200) — Hitagi's broken sequence with exactly kk zero elements.

The third line contains kk space-separated integers b1,b2,…,bkb1,b2,…,bk (1≤bi≤2001≤bi≤200) — the elements to fill into Hitagi's sequence.

Input guarantees that apart from 00, no integer occurs in aa and bb more than once in total.

**Output**

Output "Yes" if it's possible to replace zeros in aa with elements in bb and make the resulting sequence not increasing, and "No" otherwise.

**Examples**

**input**

**Copy**

4 2  
11 0 0 14  
5 4

**output**

**Copy**

Yes

**input**

**Copy**

6 1  
2 3 0 8 9 10  
5

**output**

**Copy**

No

**input**

**Copy**

4 1  
8 94 0 4  
89

**output**

**Copy**

Yes

**input**

**Copy**

7 7  
0 0 0 0 0 0 0  
1 2 3 4 5 6 7

**output**

**Copy**

Yes

**Note**

In the first sample:

* Sequence aa is 11,0,0,1411,0,0,14.
* Two of the elements are lost, and the candidates in bb are 55 and 44.
* There are two possible resulting sequences: 11,5,4,1411,5,4,14 and 11,4,5,1411,4,5,14, both of which fulfill the requirements. Thus the answer is "Yes".

In the second sample, the only possible resulting sequence is 2,3,5,8,9,102,3,5,8,9,10, which is an increasing sequence and therefore invalid.

H. Gravity Shift

time limit per test

1 s.

memory limit per test

256 MB

input

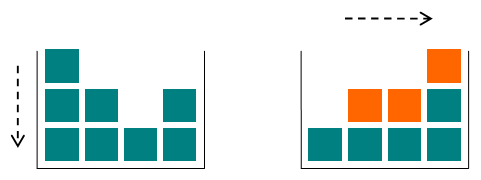
standard input

output

standard output

Little Chris is bored during his physics lessons (too easy), so he has built a toy box to keep himself occupied. The box is special, since it has the ability to change gravity.

There are nn columns of toy cubes in the box arranged in a line. The ii-th column contains aiai cubes. At first, the gravity in the box is pulling the cubes downwards. When Chris switches the gravity, it begins to pull all the cubes to the right side of the box. The figure shows the initial and final configurations of the cubes in the box: the cubes that have changed their position are highlighted with orange.



Given the initial configuration of the toy cubes in the box, find the amounts of cubes in each of the nn columns after the gravity switch!

**Input**

The first line of input contains an integer nn (1≤n≤1001≤n≤100), the number of the columns in the box. The next line contains nn space-separated integer numbers. The ii-th number aiai (1≤ai≤1001≤ai≤100) denotes the number of cubes in the ii-th column.

**Output**

Output nn integer numbers separated by spaces, where the ii-th number is the amount of cubes in the ii-th column after the gravity switch.

**Examples**

**input**

**Copy**

4  
3 2 1 2

**output**

**Copy**

1 2 2 3

**input**

**Copy**

3  
2 3 8

**output**

**Copy**

2 3 8

**Note**

The first example case is shown on the figure. The top cube of the first column falls to the top of the last column; the top cube of the second column falls to the top of the third column; the middle cube of the first column falls to the top of the second column.

In the second example case the gravity switch does not change the heights of the columns.

I. Painted Circles

time limit per test

2 s.

memory limit per test

256 MB

input

standard input

output

standard output

One day, as Sherlock Holmes was tracking down one very important criminal, he found a wonderful painting on the wall. This wall could be represented as a plane. The painting had several concentric circles that divided the wall into several parts. Some parts were painted red and all the other were painted blue. Besides, any two neighboring parts were painted different colors, that is, the red and the blue color were alternating, i. e. followed one after the other. The outer area of the wall (the area that lied outside all circles) was painted blue. Help Sherlock Holmes determine the total area of red parts of the wall.

Let us remind you that two circles are called concentric if their centers coincide. Several circles are called concentric if any two of them are concentric.

**Input**

The first line contains the single integer nn (1≤n≤1001≤n≤100). The second line contains nn space-separated integers riri (1≤ri≤10001≤ri≤1000) — the circles' radii. It is guaranteed that all circles are different.

**Output**

Print the single real number — total area of the part of the wall that is painted red. The answer is accepted if absolute or relative error doesn't exceed 10−410−4.

**Examples**

**input**

**Copy**

1  
1

**output**

**Copy**

3.1415926536

**input**

**Copy**

3  
1 4 2

**output**

**Copy**

40.8407044967

**Note**

In the first sample the picture is just one circle of radius 11. Inner part of the circle is painted red. The area of the red part equals π×12=ππ×12=π.

In the second sample there are three circles of radii 11, 44 and 22. Outside part of the second circle is painted blue. Part between the second and the third circles is painted red. Part between the first and the third is painted blue. And, finally, the inner part of the first circle is painted red. Overall there are two red parts: the ring between the second and the third circles and the inner part of the first circle. Total area of the red parts is equal (π×42−π×22)+π×12=π×12+π=13π(π×42−π×22)+π×12=π×12+π=13π

J. Network Problem

time limit per test

2 s.

memory limit per test

256 MB

input

standard input

output

standard output

Valeric and Valerko missed the last Euro football game, so they decided to watch the game's key moments on the Net. They want to start watching as soon as possible but the connection speed is too low. If they turn on the video right now, it will "hang up" as the size of data to watch per second will be more than the size of downloaded data per second.

The guys want to watch the whole video without any pauses, so they have to wait some **integer** number of seconds for a part of the video to download. After this number of seconds passes, they can start watching. Waiting for the whole video to download isn't necessary as the video can download after the guys started to watch.

Let's suppose that video's length is cc seconds and Valeric and Valerko wait tt seconds before the watching. Then for any moment of time t0t0, t≤t0≤c+tt≤t0≤c+t, the following condition must fulfill: the size of data received in t0t0 seconds is not less than the size of data needed to watch t0−tt0−t seconds of the video.

Of course, the guys want to wait as little as possible, so your task is to find the minimum integer number of seconds to wait before turning the video on. The guys must watch the video without pauses.

**Input**

The first line contains three space-separated integers aa, bb and cc (1≤a,b,c≤1000,a>b)(1≤a,b,c≤1000,a>b). The first number (aa) denotes the size of data needed to watch one second of the video. The second number (bb) denotes the size of data Valeric and Valerko can download from the Net per second. The third number (cc) denotes the video's length in seconds.

**Output**

Print a single number — the minimum integer number of seconds that Valeric and Valerko must wait to watch football without pauses.

**Examples**

**input**

**Copy**

4 1 1

**output**

**Copy**

3

**input**

**Copy**

10 3 2

**output**

**Copy**

5

**input**

**Copy**

13 12 1

**output**

**Copy**

1

**Note**

In the first sample video's length is 1 second and it is necessary 4 units of data for watching 1 second of video, so guys should download 4 ⋅⋅ 1 = 4 units of data to watch the whole video. The most optimal way is to wait 3 seconds till 3 units of data will be downloaded and then start watching. While guys will be watching video 1 second, one unit of data will be downloaded and Valerik and Valerko will have 4 units of data by the end of watching. Also every moment till the end of video guys will have more data then necessary for watching.

In the second sample guys need 2 ⋅⋅ 10 = 20 units of data, so they have to wait 5 seconds and after that they will have 20 units before the second second ends. However, if guys wait 4 seconds, they will be able to watch first second of video without pauses, but they will download 18 units of data by the end of second second and it is less then necessary.

K. Different Divisors

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

Positive integer xx is called *divisor* of positive integer yy, if yy is divisible by xx without remainder. For example, 11 is a divisor of 77 and 33 is not divisor of 88.

We gave you an integer dd and asked you to find **the smallest** positive integer aa, such that

* aa has at least 44 divisors;
* difference between any two divisors of aa is at least dd.

**Input**

The first line contains a single integer tt (1≤t≤30001≤t≤3000) — the number of test cases.

The first line of each test case contains a single integer dd (1≤d≤100001≤d≤10000).

**Output**

For each test case print one integer aa — the answer for this test case.

**Example**

**input**

**Copy**

2

1

2

**output**

**Copy**

6

15

**Note**

In the first test case, integer 66 have following divisors: [1,2,3,6][1,2,3,6]. There are 44 of them and the difference between any two of them is at least 11. There is no smaller integer with at least 44 divisors.

In the second test case, integer 1515 have following divisors: [1,3,5,15][1,3,5,15]. There are 44 of them and the difference between any two of them is at least 22.

The answer 1212 is INVALID because divisors are [1,2,3,4,6,12][1,2,3,4,6,12]. And the difference between, for example, divisors 22 and 33 is less than d=2d=2.

L. Dormitories

time limit per test

4 s.

memory limit per test

256 MB

input

standard input

output

standard output

There are nn dormitories in Berland State University, they are numbered with integers from 11 to nn. Each dormitory consists of rooms, there are aiai rooms in ii-th dormitory. The rooms in ii-th dormitory are numbered from 11 to aiai.

A postman delivers letters. Sometimes there is no specific dormitory and room number in it on an envelope. Instead of it only a room number among all rooms of all nn dormitories is written on an envelope. In this case, assume that all the rooms are numbered from 11 to a1+a2+⋯+ana1+a2+⋯+an and the rooms of the first dormitory go first, the rooms of the second dormitory go after them and so on.

For example, in case n=2n=2, a1=3a1=3 and a2=5a2=5 an envelope can have any integer from 11 to 88 written on it. If the number 77 is written on an envelope, it means that the letter should be delivered to the room number 44 of the second dormitory.

For each of mm letters by the room number among all nn dormitories, determine the particular dormitory and the room number in a dormitory where this letter should be delivered.

**Input**

The first line contains two integers nn and mm (1≤n,m≤2⋅105)(1≤n,m≤2⋅105) — the number of dormitories and the number of letters.

The second line contains a sequence a1,a2,…,ana1,a2,…,an (1≤ai≤1010)(1≤ai≤1010), where aiai equals to the number of rooms in the ii-th dormitory. The third line contains a sequence b1,b2,…,bmb1,b2,…,bm (1≤bj≤a1+a2+⋯+an)(1≤bj≤a1+a2+⋯+an), where bjbj equals to the room number (among all rooms of all dormitories) for the jj-th letter. All bjbj are given in **increasing** order.

**Output**

Print mm lines. For each letter print two integers ff and kk — the dormitory number ff (1≤f≤n)(1≤f≤n) and the room number kk in this dormitory (1≤k≤af)(1≤k≤af) to deliver the letter.

**Examples**

**input**

**Copy**

3 6  
10 15 12  
1 9 12 23 26 37

**output**

**Copy**

1 1  
1 9  
2 2  
2 13  
3 1  
3 12

**input**

**Copy**

2 3  
5 10000000000  
5 6 9999999999

**output**

**Copy**

1 5  
2 1  
2 9999999994

**Note**

In the first example letters should be delivered in the following order:

* the first letter in room 11 of the first dormitory
* the second letter in room 99 of the first dormitory
* the third letter in room 22 of the second dormitory
* the fourth letter in room 1313 of the second dormitory
* the fifth letter in room 11 of the third dormitory
* the sixth letter in room 1212 of the third dormitory

M. Optimizing Program

time limit per test

2 s.

memory limit per test

256 MB

input

standard input

output

standard output

Adilbek was assigned to a special project. For Adilbek it means that he has nn days to run a special program and provide its results. But there is a problem: the program needs to run for dd days to calculate the results.

Fortunately, Adilbek can optimize the program. If he spends xx (xx is a non-negative integer) days optimizing the program, he will make the program run in ⌈dx+1⌉⌈dx+1⌉ days (⌈a⌉⌈a⌉ is the ceiling function: ⌈2.4⌉=3⌈2.4⌉=3, ⌈2⌉=2⌈2⌉=2). The program cannot be run and optimized simultaneously, so the total number of days he will spend is equal to x+⌈dx+1⌉x+⌈dx+1⌉.

Will Adilbek be able to provide the generated results in no more than nn days?

**Input**

The first line contains a single integer TT (1≤T≤501≤T≤50) — the number of test cases.

The next TT lines contain test cases – one per line. Each line contains two integers nn and dd (1≤n≤1091≤n≤109, 1≤d≤1091≤d≤109) — the number of days before the deadline and the number of days the program runs.

**Output**

Print TT answers — one per test case. For each test case print YES (case insensitive) if Adilbek can fit in nn days or NO (case insensitive) otherwise.

**Example**

**input**

**Copy**

3

1 1

4 5

5 11

**output**

**Copy**

YES

YES

NO

**Note**

In the first test case, Adilbek decides not to optimize the program at all, since d≤nd≤n.

In the second test case, Adilbek can spend 11 day optimizing the program and it will run ⌈52⌉=3⌈52⌉=3 days. In total, he will spend 44 days and will fit in the limit.

In the third test case, it's impossible to fit in the limit. For example, if Adilbek will optimize the program 22 days, it'll still work ⌈112+1⌉=4⌈112+1⌉=4 days.

N. Jumping Game

time limit per test

2 s.

memory limit per test

256 MB

input

standard input

output

standard output

There is a frog staying to the left of the string s=s1s2…sns=s1s2…sn consisting of nn characters (to be more precise, the frog initially stays at the cell 00). Each character of ss is either 'L' or 'R'. It means that if the frog is staying at the ii-th cell and the ii-th character is 'L', the frog can jump only to the left. If the frog is staying at the ii-th cell and the ii-th character is 'R', the frog can jump only to the right. **The frog can jump only to the right from the cell**00.

**Note that the frog can jump into the same cell twice and can perform as many jumps as it needs**.

The frog wants to reach the n+1n+1-th cell. The frog chooses some **positive integer** value dd **before the first jump** (and cannot change it later) and jumps by no more than dd cells at once. I.e. if the ii-th character is 'L' then the frog can jump to any cell in a range [max(0,i−d);i−1][max(0,i−d);i−1], and if the ii-th character is 'R' then the frog can jump to any cell in a range [i+1;min(n+1;i+d)][i+1;min(n+1;i+d)].

The frog doesn't want to jump far, so your task is to find the minimum possible value of dd such that the frog can reach the cell n+1n+1 from the cell 00 if it can jump by no more than dd cells at once. **It is guaranteed that it is always possible to reach**n+1n+1**from**00.

You have to answer tt independent test cases.

**Input**

The first line of the input contains one integer tt (1≤t≤1041≤t≤104) — the number of test cases.

The next tt lines describe test cases. The ii-th test case is described as a string ss consisting of at least 11 and at most 2⋅1052⋅105 characters 'L' and 'R'.

It is guaranteed that the sum of lengths of strings over all test cases does not exceed 2⋅1052⋅105 (∑|s|≤2⋅105∑|s|≤2⋅105).

**Output**

For each test case, print the answer — the minimum possible value of dd such that the frog can reach the cell n+1n+1 from the cell 00 if it jumps by no more than dd at once.

**Example**

**input**

**Copy**

6

LRLRRLL

L

LLR

RRRR

LLLLLL

R

**output**

**Copy**

3

2

3

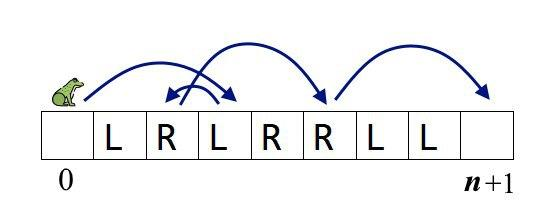
1

7

1

**Note**

The picture describing the first test case of the example and one of the possible answers:



In the second test case of the example, the frog can only jump directly from 00 to n+1n+1.

In the third test case of the example, the frog can choose d=3d=3, jump to the cell 33 from the cell 00 and then to the cell 44 from the cell 33.

In the fourth test case of the example, the frog can choose d=1d=1 and jump 55 times to the right.

In the fifth test case of the example, the frog can only jump directly from 00 to n+1n+1.

In the sixth test case of the example, the frog can choose d=1d=1 and jump 22 times to the right.

O. Word Game

time limit per test

1 s.

memory limit per test

256 MB

input

standard input

output

standard output

PolandBall is playing a game with EnemyBall. The rules are simple. Players have to say words in turns. You cannot say a word which was already said. PolandBall starts. The Ball which can't say a new word loses.

You're given two lists of words familiar to PolandBall and EnemyBall. Can you determine who wins the game, if both play optimally?

**Input**

The first input line contains two integers nn and mm (1⩽n,m⩽1031⩽n,m⩽103) — number of words PolandBall and EnemyBall know, respectively.

Then nn strings follow, one per line — words familiar to PolandBall.

Then mm strings follow, one per line — words familiar to EnemyBall.

Note that one Ball **cannot** know a word more than once (strings are unique), but some words **can** be known by both players.

Each word is non-empty and consists of no more than 500500 lowercase English alphabet letters.

**Output**

In a single line of print the answer — "YES" if PolandBall wins and "NO" otherwise. Both Balls play optimally.

**Examples**

**input**

**Copy**

5 1  
polandball  
is  
a  
cool  
character  
nope

**output**

**Copy**

YES

**input**

**Copy**

2 2  
kremowka  
wadowicka  
kremowka  
wiedenska

**output**

**Copy**

YES

**input**

**Copy**

1 2  
a  
a  
b

**output**

**Copy**

NO

**Note**

In the first example PolandBall knows much more words and wins effortlessly.

In the second example if PolandBall says kremowka first, then EnemyBall cannot use that word anymore. EnemyBall can only say wiedenska. PolandBall says wadowicka and wins.

P. Red and Blue Badges

time limit per test

1 s.

memory limit per test

512 MB

input

standard input

output

standard output

There are bb boys and gg girls participating in Olympiad of Metropolises. There will be a board games tournament in the evening and nn participants have accepted the invitation. The organizers do not know how many boys and girls are among them.

Organizers are preparing red badges for girls and blue ones for boys.

Vasya prepared n+1n+1 decks of badges. The ii-th (where ii is from 00 to nn, inclusive) deck contains ii blue badges and n−in−i red ones. The total number of badges in any deck is exactly nn.

Determine the **minimum** number of decks among these n+1n+1 that Vasya should take, so that there will be a suitable deck no matter how many girls and boys there will be among the participants of the tournament.

**Input**

The first line contains an integer bb (1≤b≤3001≤b≤300), the number of boys.

The second line contains an integer gg (1≤g≤3001≤g≤300), the number of girls.

The third line contains an integer nn (1≤n≤b+g1≤n≤b+g), the number of the board games tournament participants.

**Output**

Output the only integer, the **minimum** number of badge decks that Vasya could take.

**Examples**

**input**

**Copy**

5

6

3

**output**

**Copy**

4

**input**

**Copy**

5

3

5

**output**

**Copy**

4

**Note**

In the first example, each of 4 decks should be taken: (0 blue, 3 red), (1 blue, 2 red), (2 blue, 1 red), (3 blue, 0 red).

In the second example, 4 decks should be taken: (2 blue, 3 red), (3 blue, 2 red), (4 blue, 1 red), (5 blue, 0 red). Piles (0 blue, 5 red) and (1 blue, 4 red) can not be used.

Q. Minimum Number

time limit per test

2 s.

memory limit per test

256 MB

input

standard input

output

standard output

You are given two binary strings xx and yy, which are binary representations of some two integers (let's denote these integers as f(x)f(x) and f(y)f(y)). You can choose any integer k≥0k≥0, calculate the expression sk=f(x)+f(y)⋅2ksk=f(x)+f(y)⋅2k and write the binary representation of sksk in **reverse order** (let's denote it as revkrevk). For example, let x=1010x=1010 and y=11y=11; you've chosen k=1k=1 and, since 21=10221=102, so sk=10102+112⋅102=100002sk=10102+112⋅102=100002 and revk=00001revk=00001.

For given xx and yy, you need to choose such kk that revkrevk is **lexicographically minimal** (read notes if you don't know what does "lexicographically" means).

It's guaranteed that, with given constraints, kk exists and is finite.

**Input**

The first line contains a single integer TT (1≤T≤1001≤T≤100) — the number of queries.

Next 2T2T lines contain a description of queries: two lines per query. The first line contains one binary string xx, consisting of no more than 105105 characters. Each character is either 0 or 1.

The second line contains one binary string yy, consisting of no more than 105105 characters. Each character is either 0 or 1.

It's guaranteed, that 1≤f(y)≤f(x)1≤f(y)≤f(x) (where f(x)f(x) is the integer represented by xx, and f(y)f(y) is the integer represented by yy), both representations don't have any leading zeroes, the total length of xx over all queries doesn't exceed 105105, and the total length of yy over all queries doesn't exceed 105105.

**Output**

Print TT integers (one per query). For each query print such kk that revkrevk is lexicographically minimal.

**Example**

**input**

**Copy**

4

1010

11

10001

110

1

1

1010101010101

11110000

**output**

**Copy**

1

3

0

0

**Note**

The first query was described in the legend.

In the second query, it's optimal to choose k=3k=3. The 23=1000223=10002 so s3=100012+1102⋅10002=10001+110000=1000001s3=100012+1102⋅10002=10001+110000=1000001 and rev3=1000001rev3=1000001. For example, if k=0k=0, then s0=10111s0=10111 and rev0=11101rev0=11101, but rev3=1000001rev3=1000001 is lexicographically smaller than rev0=11101rev0=11101.

In the third query s0=10s0=10 and rev0=01rev0=01. For example, s2=101s2=101 and rev2=101rev2=101. And 0101 is lexicographically smaller than 101101.

The quote from Wikipedia: "To determine which of two strings of characters comes when arranging in *lexicographical order*, their first letters are compared. If they differ, then the string whose first letter comes earlier in the alphabet comes before the other string. If the first letters are the same, then the second letters are compared, and so on. If a position is reached where one string has no more letters to compare while the other does, then the first (shorter) string is deemed to come first in alphabetical order."

R. Bracket Sequence

time limit per test

1 s.

memory limit per test

512 MB

input

standard input

output

standard output

Petya's friends made him a birthday present — a bracket sequence. Petya was quite disappointed with his gift, because he dreamed of correct bracket sequence, yet he told his friends nothing about his dreams and decided to fix present himself.

To make everything right, Petya is going to move at most one bracket from its original place in the sequence to any other position. Reversing the bracket (e.g. turning "(" into ")" or vice versa) isn't allowed.

We remind that bracket sequence ss is called correct if:

* ss is empty;
* ss is equal to "(tt)", where tt is correct bracket sequence;
* ss is equal to t1t2t1t2, i.e. concatenation of t1t1 and t2t2, where t1t1 and t2t2 are correct bracket sequences.

For example, "(()())", "()" are correct, while ")(" and "())" are not. Help Petya to fix his birthday present and understand whether he can move one bracket so that the sequence becomes correct.

**Input**

First of line of input contains a single number nn (1≤n≤2000001≤n≤200000) — length of the sequence which Petya received for his birthday.

Second line of the input contains bracket sequence of length nn, containing symbols "(" and ")".

**Output**

Print "Yes" if Petya can make his sequence correct moving at most one bracket. Otherwise print "No".

**Examples**

**input**

**Copy**

2

)(

**output**

**Copy**

Yes

**input**

**Copy**

3

(()

**output**

**Copy**

No

**input**

**Copy**

2

()

**output**

**Copy**

Yes

**input**

**Copy**

10

)))))(((((

**output**

**Copy**

No

**Note**

In the first example, Petya can move first bracket to the end, thus turning the sequence into "()", which is correct bracket sequence.

In the second example, there is no way to move at most one bracket so that the sequence becomes correct.

In the third example, the sequence is already correct and there's no need to move brackets.

S. Game with string

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

Two people are playing a game with a string ss, consisting of lowercase latin letters.

On a player's turn, he should choose two consecutive equal letters in the string and delete them.

For example, if the string is equal to "xaax" than there is only one possible turn: delete "aa", so the string will become "xx". A player not able to make a turn loses.

Your task is to determine which player will win if both play optimally.

**Input**

The only line contains the string ss, consisting of lowercase latin letters (1≤|s|≤1000001≤|s|≤100000), where |s||s| means the length of a string ss.

**Output**

If the first player wins, print "Yes". If the second player wins, print "No".

**Examples**

**input**

**Copy**

abacaba

**output**

**Copy**

No

**input**

**Copy**

iiq

**output**

**Copy**

Yes

**input**

**Copy**

abba

**output**

**Copy**

No

**Note**

In the first example the first player is unable to make a turn, so he loses.

In the second example first player turns the string into "q", then second player is unable to move, so he loses.

T. Chat Order

time limit per test

3 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

Polycarp is a big lover of killing time in social networks. A page with a chatlist in his favourite network is made so that when a message is sent to some friend, his friend's chat rises to the very top of the page. The relative order of the other chats doesn't change. If there was no chat with this friend before, then a new chat is simply inserted to the top of the list.

Assuming that the chat list is initially empty, given the sequence of Polycaprus' messages make a list of chats after all of his messages are processed. Assume that no friend wrote any message to Polycarpus.

**Input**

The first line contains integer *n* (1 ≤ *n* ≤ 200 000) — the number of Polycarpus' messages. Next *n* lines enlist the message recipients in the order in which the messages were sent. The name of each participant is a non-empty sequence of lowercase English letters of length at most 10.

**Output**

Print all the recipients to who Polycarp talked to in the order of chats with them, from top to bottom.

**Examples**

**input**

**Copy**

4  
alex  
ivan  
roman  
ivan

**output**

**Copy**

ivan  
roman  
alex

**input**

**Copy**

8  
alina  
maria  
ekaterina  
darya  
darya  
ekaterina  
maria  
alina

**output**

**Copy**

alina  
maria  
ekaterina  
darya

**Note**

In the first test case Polycarpus first writes to friend by name "alex", and the list looks as follows:

1. alex

Then Polycarpus writes to friend by name "ivan" and the list looks as follows:

1. ivan
2. alex

Polycarpus writes the third message to friend by name "roman" and the list looks as follows:

1. roman
2. ivan
3. alex

Polycarpus writes the fourth message to friend by name "ivan", to who he has already sent a message, so the list of chats changes as follows:

1. ivan
2. roman
3. alex

U. Make Them Equal

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

You are given a sequence a1,a2,…,ana1,a2,…,an consisting of nn integers.

You can choose any non-negative integer DD (i.e. D≥0D≥0), and for each aiai you can:

* add DD (only once), i. e. perform ai:=ai+Dai:=ai+D, or
* subtract DD (only once), i. e. perform ai:=ai−Dai:=ai−D, or
* leave the value of aiai unchanged.

It is possible that after an operation the value aiai becomes negative.

Your goal is to choose such **minimum non-negative integer** DD and perform changes in such a way, that all aiai are equal (i.e. a1=a2=⋯=ana1=a2=⋯=an).

Print the required DD or, if it is impossible to choose such value DD, print -1.

For example, for array [2,8][2,8] the value D=3D=3 is minimum possible because you can obtain the array [5,5][5,5] if you will add DD to 22 and subtract DD from 88. And for array [1,4,7,7][1,4,7,7] the value D=3D=3 is also minimum possible. You can add it to 11 and subtract it from 77 and obtain the array [4,4,4,4][4,4,4,4].

**Input**

The first line of the input contains one integer nn (1≤n≤1001≤n≤100) — the number of elements in aa.

The second line of the input contains nn integers a1,a2,…,ana1,a2,…,an (1≤ai≤1001≤ai≤100) — the sequence aa.

**Output**

Print one integer — the **minimum non-negative integer** value DD such that if you add this value to some aiai, subtract this value from some aiai and leave some aiai without changes, all obtained values become equal.

If it is impossible to choose such value DD, print -1.

**Examples**

**input**

**Copy**

6

1 4 4 7 4 1

**output**

**Copy**

3

**input**

**Copy**

5

2 2 5 2 5

**output**

**Copy**

3

**input**

**Copy**

4

1 3 3 7

**output**

**Copy**

-1

**input**

**Copy**

2

2 8

**output**

**Copy**

3

V. Pursuit

time limit per test

2 seconds

memory limit per test

512 megabytes

input

standard input

output

standard output

You and your friend Ilya are participating in an individual programming contest consisting of multiple stages. A contestant can get between 00 and 100100 points, inclusive, for each stage, independently of other contestants.

Points received by contestants in different stages are used for forming overall contest results. Suppose that kk stages of the contest are completed. For each contestant, k−⌊k4⌋k−⌊k4⌋ stages with the highest scores are selected, and these scores are added up. This sum is the overall result of the contestant. (Here ⌊t⌋⌊t⌋ denotes rounding tt down.)

For example, suppose 99 stages are completed, and your scores are 50,30,50,50,100,10,30,100,5050,30,50,50,100,10,30,100,50. First, 77 stages with the highest scores are chosen — for example, all stages except for the 22-nd and the 66-th can be chosen. Then your overall result is equal to 50+50+50+100+30+100+50=43050+50+50+100+30+100+50=430.

As of now, nn stages are completed, and you know the points you and Ilya got for these stages. However, it is unknown how many more stages will be held. You wonder what the smallest number of additional stages is, after which your result might become greater than or equal to Ilya's result, at least in theory. Find this number!

**Input**

Each test contains multiple test cases. The first line contains the number of test cases tt (1≤t≤10001≤t≤1000). Description of the test cases follows.

The first line of each test case contains a single integer nn (1≤n≤1051≤n≤105) — the number of completed stages.

The second line contains nn integers a1,a2,…,ana1,a2,…,an (0≤ai≤1000≤ai≤100) — your points for the completed stages.

The third line contains nn integers b1,b2,…,bnb1,b2,…,bn (0≤bi≤1000≤bi≤100) — Ilya's points for the completed stages.

It is guaranteed that the sum of nn over all test cases does not exceed 105105.

**Output**

For each test case print a single integer — the smallest number of additional stages required for your result to be able to become greater than or equal to Ilya's result.

If your result is already not less than Ilya's result, print 00.

**Example**

**input**

**Copy**

5

1

100

0

1

0

100

4

20 30 40 50

100 100 100 100

4

10 20 30 40

100 100 100 100

7

7 59 62 52 27 31 55

33 35 50 98 83 80 64

**output**

**Copy**

0

1

3

4

2

**Note**

In the first test case, you have scored 100100 points for the first stage, while Ilya has scored 00. Thus, your overall result (100100) is already not less than Ilya's result (00).

In the second test case, you have scored 00 points for the first stage, while Ilya has scored 100100. A single stage with an opposite result is enough for both your and Ilya's overall scores to become equal to 100100.

In the third test case, your overall result is 30+40+50=12030+40+50=120, while Ilya's result is 100+100+100=300100+100+100=300. After three additional stages your result might become equal to 420420, while Ilya's result might become equal to 400400.

In the fourth test case, your overall result after four additional stages might become equal to 470470, while Ilya's result might become equal to 400400. Three stages are not enough.

W. Verse For Santa

time limit per test

1 second

memory limit per test

256 megabytes

input

standard input

output

standard output

New Year is coming! Vasya has prepared a New Year's verse and wants to recite it in front of Santa Claus.

Vasya's verse contains nn parts. It takes aiai seconds to recite the ii-th part. Vasya can't change the order of parts in the verse: firstly he recites the part which takes a1a1 seconds, secondly — the part which takes a2a2 seconds, and so on. After reciting the verse, Vasya will get the number of presents equal to the number of parts he fully recited.

Vasya can skip at most one part of the verse while reciting it (if he skips more than one part, then Santa will definitely notice it).

Santa will listen to Vasya's verse for no more than ss seconds. For example, if s=10s=10, a=[100,9,1,1]a=[100,9,1,1], and Vasya skips the first part of verse, then he gets two presents.

Note that it is possible to recite the whole verse (if there is enough time).

Determine which part Vasya needs to skip to obtain the maximum possible number of gifts. If Vasya shouldn't skip anything, print 0. If there are multiple answers, print any of them.

You have to process tt test cases.

**Input**

The first line contains one integer tt (1≤t≤1001≤t≤100) — the number of test cases.

The first line of each test case contains two integers nn and ss (1≤n≤105,1≤s≤1091≤n≤105,1≤s≤109) — the number of parts in the verse and the maximum number of seconds Santa will listen to Vasya, respectively.

The second line of each test case contains nn integers a1,a2,…,ana1,a2,…,an (1≤ai≤1091≤ai≤109) — the time it takes to recite each part of the verse.

It is guaranteed that the sum of nn over all test cases does not exceed 105105.

**Output**

For each test case print one integer — the number of the part that Vasya needs to skip to obtain the maximum number of gifts. If Vasya shouldn't skip any parts, print 0.

**Example**

**input**

**Copy**

3

7 11

2 9 1 3 18 1 4

4 35

11 9 10 7

1 8

5

**output**

**Copy**

2

1

0

**Note**

In the first test case if Vasya skips the second part then he gets three gifts.

In the second test case no matter what part of the verse Vasya skips.

In the third test case Vasya can recite the whole verse.

X. Swords

time limit per test

2 seconds

memory limit per test

256 megabytes

input

standard input

output

standard output

There were nn types of swords in the theater basement which had been used during the plays. Moreover there were **exactly** xx swords of each type. yy people have broken into the theater basement and each of them has taken exactly zz swords of some **single type**. Note that different people might have taken different types of swords. Note that the values x,yx,y and zz are unknown for you.

The next morning the director of the theater discovers the loss. He counts all swords — exactly aiai swords of the ii-th type are left untouched.

The director has no clue about the initial number of swords of each type in the basement, the number of people who have broken into the basement and how many swords each of them have taken.

For example, if n=3n=3, a=[3,12,6]a=[3,12,6] then one of the possible situations is x=12x=12, y=5y=5 and z=3z=3. Then the first three people took swords of the first type and the other two people took swords of the third type. Note that you don't know values x,yx,y and zz beforehand but know values of nn and aa.

Thus he seeks for your help. Determine the **minimum** number of people yy, which could have broken into the theater basement, and the number of swords zz each of them has taken.

**Input**

The first line of the input contains one integer nn (2≤n≤2⋅105)(2≤n≤2⋅105) — the number of types of swords.

The second line of the input contains the sequence a1,a2,…,ana1,a2,…,an (0≤ai≤109)(0≤ai≤109), where aiai equals to the number of swords of the ii-th type, which have remained in the basement after the theft. It is guaranteed that there exists at least one such pair of indices (j,k)(j,k) that aj≠akaj≠ak.

**Output**

Print two integers yy and zz — the minimum number of people which could have broken into the basement and the number of swords each of them has taken.

**Examples**

**input**

**Copy**

3

3 12 6

**output**

**Copy**

5 3

**input**

**Copy**

2

2 9

**output**

**Copy**

1 7

**input**

**Copy**

7

2 1000000000 4 6 8 4 2

**output**

**Copy**

2999999987 2

**input**

**Copy**

6

13 52 0 13 26 52

**output**

**Copy**

12 13

**Note**

In the first example the minimum value of yy equals to 55, i.e. the minimum number of people who could have broken into the basement, is 55. Each of them has taken 33 swords: three of them have taken 33 swords of the first type, and two others have taken 33 swords of the third type.

In the second example the minimum value of yy is 11, i.e. the minimum number of people who could have broken into the basement, equals to 11. He has taken 77 swords of the first type.

Y. Substring Game in the Lesson

time limit per test

2 seconds

memory limit per test

256 mebibytes

input

*standard input*

output

*standard output*

Mike and Ann are sitting in the classroom. The lesson is boring, so they decided to play an interesting game. Fortunately, all they need to play this game is a string ss and a number kk (0≤k<|s|0≤k<|s|).

At the beginning of the game, players are given a substring of ss with left border ll and right border rr, both equal to kk (i.e. initially l=r=kl=r=k). Then players start to make moves one by one, according to the following rules:

* A player chooses l′l′ and r′r′ so that l′≤ll′≤l, r′≥rr′≥r and s[l′,r′]s[l′,r′] is lexicographically less than s[l,r]s[l,r]. Then the player changes ll and rr in this way: l:=l′l:=l′, r:=r′r:=r′.
* Ann moves first.
* The player, that can't make a move loses.

Recall that a substring s[l,r]s[l,r] (l≤rl≤r) of a string ss is a continuous segment of letters from s that starts at position ll and ends at position rr. For example, "ehn" is a substring (s[3,5]s[3,5]) of "aaaehnsvz" and "ahz" is not.

Mike and Ann were playing so enthusiastically that they did not notice the teacher approached them. Surprisingly, the teacher didn't scold them, instead of that he said, that he can figure out the winner of the game before it starts, even if he knows only ss and kk.

Unfortunately, Mike and Ann are not so keen in the game theory, so they ask you to write a program, that takes ss and determines the winner for all possible kk.

**Input**

The first line of the input contains a single string ss (1≤|s|≤5⋅1051≤|s|≤5⋅105) consisting of lowercase English letters.

**Output**

Print |s||s| lines.

In the line ii write the name of the winner (print Mike or Ann) in the game with string ss and k=ik=i, if both play optimally

**Examples**

**input**

**Copy**

abba

**output**

**Copy**

Mike

Ann

Ann

Mike

**input**

**Copy**

cba

**output**

**Copy**

Mike

Mike

Mike

Z. Polygon

time limit per test

2 seconds

memory limit per test

256 megabytes

input

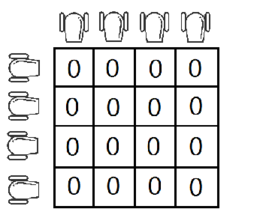
standard input

output

standard output

Polygon is not only the best platform for developing problems but also a square matrix with side nn, initially filled with the character 0.

On the polygon, military training was held. The soldiers placed a cannon above each cell in the first row and a cannon to the left of each cell in the first column. Thus, exactly 2n2n cannons were placed.

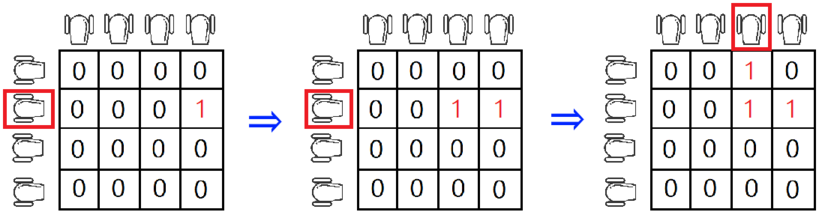
Initial polygon for n=4n=4.

Cannons shoot character 1. At any moment of time, no more than one cannon is shooting. When a 1 flies out of a cannon, it flies forward (in the direction of the shot) until it collides with a polygon border or another 1. After that, it takes the cell in which it was before the collision and remains there. Take a look at the examples for better understanding.

More formally:

* if a cannon stands in the row ii, to the left of the first column, and shoots with a 1, then the 1 starts its flight from the cell (i,1i,1) and ends in some cell (i,ji,j);
* if a cannon stands in the column jj, above the first row, and shoots with a 1, then the 1 starts its flight from the cell (1,j1,j) and ends in some cell (i,ji,j).

For example, consider the following sequence of shots:



1. Shoot the cannon in the row 22.                         2. Shoot the cannon in the row 22.                         3. Shoot the cannon in column 33.

You have a report from the military training on your desk. This report is a square matrix with side length nn consisting of 0 and 1. You wonder if the training actually happened. In other words, is there a sequence of shots such that, after the training, you get the given matrix?

Each cannon can make an arbitrary number of shots. Before the training, each cell of the polygon contains 0.

**Input**

The first line contains an integer tt (1≤t≤10001≤t≤1000) — the number of test cases. Then tt test cases follow.

Each test case starts with a line containing an integer nn (1≤n≤501≤n≤50) — the size of the polygon.

This is followed by nn lines of length nn, consisting of 0 and 1 — the polygon matrix after the training.

The total area of the matrices in all test cases in one test does not exceed 105105.

**Output**

For each test case print:

* YES if there is a sequence of shots leading to a given matrix;
* NO if such a sequence does not exist.

The letters in the words YES and NO can be printed in any case.

**Example**

**input**

**Copy**

5

4

0010

0011

0000

0000

2

10

01

2

00

00

4

0101

1111

0101

0111

4

0100

1110

0101

0111

**output**

**Copy**

YES

NO

YES

YES

NO

**Note**

The first test case was explained in the statement.

The answer to the second test case is NO, since a 1 in a cell (1,11,1) flying out of any cannon would continue its flight further.