IEEE Extreme 10.0

Questions October 22, 2016

Collected by Basit Qureshi qureshi@psu.edu.sa

Contents

Dog Walking

Your friend, Alice, is starting a dog walking business. She already has K dog walkers employed, and today there are N dogs that need to be walked. Each dog walker can walk multiple dogs at the same time, so the dogs will be arranged into K nonempty groups, and each group will then be walked by a single dog walker. However, smaller dogs can be aggressive towards larger dogs, and that makes it harder to walk them together.
More formally, if the smallest dog in a group has size a, and the largest dog in the group has size b, then the range of the group is defined as b-a. In particular, the range of a group consisting of a single dog is 0. The smaller the range of a group is, the easier it is to walk that particular group. Hence Alice would like to distribute the dogs among the dog walkers so that the sum of ranges of the groups is minimized. Also, since she doesn't want any of the dog walkers to feel left out, she makes sure each dog walker gets to walk at least one dog.
Given N, K and the sizes of the dogs, can you help Alice determine what is the minimum sum of ranges over the K groups if the dogs are arranged optimally?

Input Format

The first line of input contains t, $1 \leq t \leq 5$, which gives the number of test cases. Each test case starts with a line containing two integers N, the number of dogs, and K, the number of employees, separated by a single space. Then N lines follow, one for each dog, containing an integer $x$representing the size of the corresponding dog.

Constraints

$1 \leq K \leq N \leq 10^5$, $0 \leq x \leq 10^9$

Output Format

For each test case, you should output, on a line by itself, the minimum sum of ranges over the K groups if the dogs are arranged optimally.

Sample Input

```
2
4 2
3
5
1
1
5 4
```

```
30
40
20
41
50
```

Sample Output

```
2
1
```

Explanation

In the first test case there are four dogs: one of size 3, one of size 5, and two of size 1. There are two dog walkers, and we want to distribute the dogs among them. One optimal way to do this is to make one dog walker walk the dogs of size 3 and 5, and the other dog walker walk the two dogs of size 1. Then the first group has range 5-3=2, while the second group has range 1-1=0, giving a total of 2+0=2.

In the second test case there are dogs of size 30, 40, 20, 41 and 50, and four dog walkers. There are so many dog walkers that we can ask all but one of them to walk a single dog. We will make the last dog walker walk the dogs of size 40 and 41, which gives a range of 41-40=1. All other groups have range 0, so the total is 1.

Playing 20 Questions with an Unreliable Friend

To celebrate the 10th anniversary of Xtreme, your friend has arranged 10 balloons in a row in the next room, and has challenged you to guess the sequence of the colors in the row of balloons. The balloons can be red, blue, or green. Your friend will answer a series of yes/no questions about the colors of the balloons. Unfortunately, your friend will tell a certain number of lies when answering your questions.

The questions can be in one of the following forms:

1. You may ask if a particular balloon is a particular color, e.g.:
   - "Is the second balloon red?"
   - "Is the 10th balloon blue?"
2. You may ask about the count of balloons of a particular color, e.g.:
   - "Are there 3 red balloons?"
   - "Are there 0 blue balloons?"
3. The previous types of questions can be subquestions that are combined together into a larger question with *or*'s or *and*'s. When combined with an *or*, only one of the answers to the subquestions must be`yes` for the answer to the entire question to be `yes`, and when combined with *and*, all of the answers to the subquestions must be `yes` in order to the answer to the larger question to be `yes`.

- "Is the third balloon green *or* the fourth balloon red?"
- "Is the tenth balloon red *and* are there three red balloons *and* is the first balloon blue?"

Note that subquestions in a particular question will be combined either with *or*'s or *and*'s, but not both. You are not allowed to ask a question like "Is the tenth balloon red *or* are there three red balloons *and* is the first balloon blue?"
At the beginning of the game, your friend will tell you how many answers to your questions will be lies.*Your friend will be honest when telling you the number of lies he is about to tell*. Your task is to determine what colors each of the balloons could be, given the answers to your questions and the number of lies that were told.

Input Format

The input begins an integer t, $1 \le t \le 20$, which gives the number of testcases in the input.
There will be a blank line preceding each testcase. Each testcase begins a line containing two space-separated integers q and n, where q is the number of questions that you asked, and n is the number of lies that your friend told when answering your questions. Note: $1 \le q \le 20$, $0 \le n \le q$
The next 2 q lines represent the questions and answers. A question will be made up of between 1 and 10, inclusive, subquestions in one of the following forms:

```
color i c

count c j
```

The first type of subquestion is asking if the i^{th} balloon is the color c. i will be an integer, 1 ≤ i ≤ 10, and $c$will be one of the following characters: `r`, `g`, or `b`.
The second type of subquestion is asking if the number of balloons of color c is equal to j. c will again be one of the following characters: `r`, `g`, or `b`. j will be an integer, 0 ≤ j ≤ 10.
When there are multiple subquestions in a question, they will be separated by `or` or `and`.
The answer to each question will appear on the line immediately following the question, and it will be either `yes` or `no`.

Output Format

For each test case, you should output a single line containing ten space separated values, where the i^{th}value in the line corresponds to what you conclude about the color of the i^{th} balloon. Each of the values will be one of the following strings:

- `r`, if you know that the balloon is red.
- `g`, if you know that the balloon is green.
- `b`, if you know that the balloon is blue.
- `rg`, if you know that the balloon must be either red or green.
- `rb`, if you know that the balloon must be either red or blue.
- `gb`, if you know that the balloon must be either blue or green.
- `rgb`, if you know that the balloon could be any of the possible colors.

Note that there should not be a space after the last value in the line.

Sample Input

```
3

2 2
color 1 b
yes
```

```
color 2 r
no

3 1
count r 4 and count g 7
yes
color 1 b and color 2 r and color 3 b
yes
color 1 g or color 4 g
yes

2 0
count r 1
yes
color 6 b or color 1 r
yes
```

Sample Output

```
rg r rgb rgb rgb rgb rgb rgb rgb rgb
b r b g rgb rgb rgb rgb rgb rgb
rgb rgb rgb rgb rgb gb rgb rgb rgb rgb
```

Explanation

First Testcase

In the first testcase, you ask two questions, and your friend lies about both of the answers.

Your first question is "Is the first balloon blue?" Since your friend lied when he said "yes", you know that it must be red or green.

Your second question is "Is the second balloon red?" Since your friend lied when he said "no", you know, in fact, that it must be red.

Second Testcase

For the second test case, you ask three questions and your friend lies in one of the answers.

In the first question, you ask "Are there 4 red balloons and 7 green balloons?" Your friend answers "yes", but since there are only 10 balloons, this must be a lie. You can then conclude that the remaining answers are truthful.

In your second question, you ask "Is the first balloon blue, the second balloon red, and the third balloon blue?" Since your friend is telling the truth when he answers "yes", you now know the colors of the first three balloons.

In your final question, you ask "Is the first balloon green or the fourth balloon green?" Your friend truthfully answers "yes". Thus you can conclude that one of the following must be true:

1. Both the first and the fourth balloons are green.

2. The first balloon is green, but the fourth is not.

3. The fourth balloon is green, but the first one is not.

However, since you already know that the first balloon is blue, you know that it is the third case that must be true, so you conclude that the fourth balloon is green.

Third Testcase

For the final testcase, your friend did not lie in any of the answers. You know that:

- There is one red balloon.

- Either the sixth balloon is blue or the first balloon is red, or both.

Note that if the first balloon is red, then no other balloons can be red, because of the first answer. If the first balloon is not red, then the sixth balloon must be blue, because of the second answer. Therefore, there is no scenario in which the sixth balloon can be red.

Inti Sets

In order to motivate his Peruvian students, a teacher includes words in the Quechua language in his math class.

Today, he defined a curious set for a given positive integer N. He called this set, an *Inti set*, and defined it as the set of all positive integer numbers that have the number *1* as their single common positive divisor with number N.
The math class about Inti sets was amazing. After class, the students try to challenge to teacher. They each ask questions like this: "Could you tell me the sum of all numbers, between A and B (inclusive), that are in the Inti set of N?"

Since the teacher is tired and he's sure that you are the best in class, he wants to know if you can help him.

Input Format
The first line of input contains an integer Q, $1 \le Q \le 20$, representing the number of students. Each of the next Q lines contain three space-separated integers N, A and B, which represent a query.
Constraints
$1 \le A \le B \le N \le 10^{12}$
Output Format
The output is exactly Q lines, one per student query. For each query you need to find the sum of all numbers between A and B, that are in the Inti set of N, and print the sum modulo 1000000007.
Sample Input
```
2
12 5 10
5 1 4
```
Sample Output
```
12
10
```
Explanation
In the sample input, $Q = 2$, so you have to answer two questions:
In the first question $N = 12$, $A = 5$ and $B = 10$. So you have to find the sum of all numbers between 5 and 10, that are in the Inti set of 12.

Inti set (12) = { 1, 5, 7, 11, 13, ... }

2 and 4 are not in the Inti set (12) because 12 and these numbers are also divisible by 2.

3 and 9 are not in the Inti set (12) because 12 and these numbers are also divisible by 3.

The numbers in the Inti set, which are in the query's range, are 5 and 7, so answer is (5 + 7) MOD 1000000007 = 12

In the second question, the numbers in the Inti set of 5 between 1 and 4 are: 1, 2, 3, 4; so the answer is (1 + 2 + 3 + 4) MOD 1000000007 = 10

Painter's Dilemma

Bob just got his first job as a house painter. Today, on his first day on the job, he has to paint a number of walls.

For those of you that have done some house painting before, you know that this is no easy task. Each wall has to be painted in multiple rounds, possibly with a different color of paint each time, and there needs to be sufficient time of waiting between consecutive rounds for the paint to dry.

To make optimal use of their time, house painters usually paint walls while waiting for other walls to dry, and hence interleave the rounds of painting different walls. But this also brings up another challenge for the painters: different walls may require different colors of paint, so they might have to replace the color on one of their paint brushes before painting that wall. But this requires washing the paint brush, waiting for it to dry, and then applying the new paint to the brush, all of which takes precious time.

The more experienced house painters circumvent the issue by bringing a lot of paint brushes. But Bob is not that fortunate, and only has two paint brushes!

Given a sequence of colors c_1, c_2, ..., c_N that Bob needs, in the order that he needs them, can you help him determine the minimum number of times he needs to change the color of one of his brushes? Both of his brushes will have no color to begin with.

Bob may ask you to compute this number for a few different scenarios, but not many. After all, he only needs to do this until he gets his first paycheck, at which point all his effort will have been worth the trouble, and he can go buy more paint brushes.

Input Format
The first line of input contains t, $1 \le t \le 5$, which gives the number of scenarios. Each scenario consists of two lines. The first line contains an integer N, the length of the sequence of colors Bob needs. The second line contains a sequence of N integers c_1, c_2, ..., c_N, representing the sequence of colors that Bob needs, in the order that he needs them. Each distinct color is represented with a distinct integer.
Constraints
$1 \le N \le 500$, $1 \le c_i \le 20$
Output Format

For each scenario, you should output, on a line by itself, the minimum number of times Bob needs to change the color of one of his brushes.

Sample Input

```
2
5
7 7 2 11 7
10
9 1 7 6 9 9 8 7 6 7
```

Sample Output

```
3
6
```

Explanation

In the first scenario, Bob needs to paint using the colors 7, 7, 2, 11, and 7, in that order. He could start by applying color 7 to the first brush. Then he can use the first brush for the first two times. The third time he needs the color 2. He could apply that color to his second brush, and thus use his second brush for the third time. Next he needs the color 11, so he might apply this color to the first brush, and use the first brush this time. Finally, he needs the color 7 just as before. But the first brush no longer has this color, so we need to reapply it. Just as an example, he could apply 7 to the second brush, and then use the second brush. In total, he had to change the color of one of his brushes 4 times.

However, Bob can be smarter about the way he changes colors. For example, considering the same sequence as before, he could start by applying color 7 to the first brush, and use the first brush for the first two times. Then he could use the second brush twice, first by applying the color 2, and then by applying the color 11. This leaves the first brush with paint 7, which he can use for the last time. This leaves him with only 3 color changes in total.

Food Truck

This problem was created and sponsored by Nokia. If they opted in, top finishers will be contacted about possible careers at Nokia.

Madhu has a food-truck called "The Yummy Goods" that goes to a different business hotspot every day at lunch! Madhu wants to perform location-based advertising to folks in the offices near her halt. To do this she uses the GPS location as a longitude and a latitude at the stop and decides on a radius (r) value. She wants to broadcast advertisement SMSes, to customers within this radius, advertising her food-truck.

She needs your help to generate the list of phone numbers of such folks. She has access to a big file of telecom data, which among other details, contains the phone number, longitude, and latitude of active cell-phone users in the city at that moment.

In order to calculate the distance between her stops and her subscribers, she wants you to use the most recent location available for each subscriber. To calculate the distance, you should use the Haversine formula:

$d = 2 \times r \times \arcsin(\text{sqrt}(\sin^2((lat1 - lat2)/2) + \cos(lat1) \times \cos(lat2) \times \sin^2((long1 - long2)/2)))$
where d is the distance between two points on the surface of the earth, in km's
r is the radius of the earth (6378.137 km for this problem)
$lat1$, $long1$ are the latitude and longitude, respectively, of point 1
$lat2$, $long2$ are the latitude and longitude, respectively, of point 2

Input Format

The first line contains Madhu's latitude and longitude in degrees, separated by a comma.

The second line contains the radius r in kms, within which she wants to broadcast her advertisement.

The third line is a header for the data in the subsequent lines.

The remaining lines have rows of telecom data of active cellphone users. Each line contains the following comma-separated fields:

- A time stamp in `MM/DD/YYYY hh:mm` format. `MM`, is a two-digit month, e.g. `01` for January, `DD` is a two-digit day of month (`01` through `31`), `YYYY` is a four-digit year, `hh` is the two digits of hour (`00` through`23`), and `mm` is the two digits of minute (`00` through `59`)

- The latitude of the subscriber, in degrees
- The longitude of the subscriber, in degrees
- The subscriber's phone number, as a 10-digit number

Notes:

- Some subscribers may appear multiple times. You should use the most recent entry to determine the location of a subscriber. If a subscriber appears multiple times, the date/time stamps will differ.
- None of the field values will contain commas.

Constraints

In order to eliminate rounding and approximation errors, no subscribers will be at a distance d from Madhu, such that $0.99 \times r \leq d \leq 1.01 \times r$
$1 \leq r \leq 100$

There will be at most 50,000 lines in the subscriber list.

Output Format

A comma separated list of phone numbers for subscribers within a radius r of the stop, sorted in ascending order.

Sample Input

```
18.9778972,72.8321983

1.0

Date&Time,Latitude,Longitude,PhoneNumber

10/21/2016 13:34,18.912875,72.822318,9020320100

10/21/2016 10:35,18.9582233,72.8275845,9020320024

10/21/2016 15:20,18.95169982,72.83525604,9020320047

10/21/2016 15:23,18.9513048,72.8343388,9020357980

10/21/2016 15:23,18.9513048,72.8343388,9020357962

10/21/2016 15:28,18.9548652,72.8332443,9020320027

10/21/2016 14:03,18.9179784,72.8279306,9020357972
```

```
10/21/2016 14:03,18.9179784,72.8279306,9020357959

10/21/2016 09:52,18.97523123,72.83494895,9020320007

10/21/2016 09:44,18.9715932,72.8383992,9020357607

10/21/2016 09:44,18.9715932,72.8383992,9020357593

10/21/2016 09:44,18.9715932,72.8383992,9020357584

10/21/2016 14:57,18.93438826,72.82704499,9020320011

10/21/2016 09:56,18.97596514,72.8327072,9020320045

10/21/2016 08:33,18.9811929,72.8353202,9020320084

10/21/2016 13:27,18.9159265,72.8245989,9020357896

10/21/2016 13:09,18.9077347,72.8076201,9020320094

10/21/2016 10:52,18.97523003,72.83494865,9020320007
```

Sample Output

```
9020320007,9020320045,9020320084,9020357584,9020357593,9020357607
```

Explanation

We can calculate the distance between the location "18.9778972, 72.8321983" and each of the subscribers whose details are provided. Only the 6 phone numbers, listed in the *Sample Output* set, have a distance to the location of the food-truck that is less than 1.0 km.

Memory Management

Dr. X develops various highly optimized code for applications ranging from smart watches to autonomous vehicles using Wintel Operating Systems and Processors. For its next generation of Operating Systems, Wintel decided to change the page replacement algorithm from First-In-First-Out (FIFO) to Least Recently Used (LRU) in order to improve the performance. In an OS, page replacement algorithms decide which memory pages to page out (swap out, write to disk) when a page of memory needs to be allocated.

In a FIFO page replacement, the OS keeps track of all the pages in memory in a queue, with the most recent arrival at the back, and the oldest arrival in front. When a page needs to be replaced, the page at the front of the queue (the oldest page) is selected.

In a LRU page replacement, the OS maintains a list containing all the pages in memory. At the back of this list is the least recently used page and at the front is the most recently used page. Whenever a new page is accessed, the OS checks if the page is present in the list or not. If the page is listed, the page is moved to the front of the list. If the page is not listed, then the OS would evict the least recently used page from the list and add the new page to the front of the list.

Dr. X is excited by the prospect of improved performance for LRU page replacement so that he can advertise the improved performance of his applications. But Dr. X is aware of the fact that not all of his applications will get a performance boost because of the change. So he has hired you to develop a program to determine if an application can be advertised or not. Your program will monitor the various data accesses by the application and should determine the number of times that the pages will have to be replaced. If the number of page replacements with the LRU page replacement algorithm is less than with the FIFO page replacement algorithm, then Dr. X will be able to advertise the application.

Input Format

The input begins with an integer t, $1 \leq t \leq 20$, which gives the number of test cases in the input.
Each test case begins with a line containing three space-separated integers p, s and n, where p is the number of pages present in the OS for a program, s is the size of each page and n is the number of memory accesses by the application.
The next n lines will contain the various addresses accessed by the application.

Note: In order to solve this challenge, you will need to identify the page that corresponds to the addresses listed. The page number is given by the formula: floor([address]/s).

Constraints

p and s will always be a power of 2
$1 \le p \le 128$, $128 \le s \le 4096$, $1 \le n \le 600$
The memory addresses range from 0 to 2^{31}-1, inclusive

Output Format

For each test case, you should output a single line with three space-separated values:

- The first value should be either `yes` or `no`: If the application can be advertised, then the output should be `yes`. If not, the output should be `no`.
- The second value should be the number of page replacements under the FIFO page replacement approach.
- The last value should be the number of page replacements under the LRU page replacement approach.

Sample Input

```
2
4 1024 5
0
1024
2048
3076
4096
2 128 7
0
255
127
256
60
1024
120
```

Sample Output

```
no 1 1
yes 3 2
```

Explanation

First Test case

For the first test case, there are only four pages which can be allocated and the size of each page is 1024 bytes. The addresses 0, 1024, 2048, 3076 and 4096 correspond to the 0th, 1st, 2nd, 3rd and 4th page.

The page replacements are shown with asterisks (*) in the following tables.

FIFO Page Replacement:

Cycle	0	1	2	3	4	5
Page Requested	-	0	1	2	3	4
OS Page 0	-	0	0	0	0	4*
OS Page 1	-	-	1	1	1	1
OS Page 2	-	-	-	2	2	2
OS Page 3	-	-	-	-	3	3

LRU Page Replacement:

Cycle	0	1	2	3	4	5
Page Requested	-	0	1	2	3	4
OS Page 0	-	0	0	0	0	4*
OS Page 1	-	-	1	1	1	1
OS Page 2	-	-	-	2	2	2
OS Page 3	-	-	-	-	3	3

Both LRU and FIFO page replacement algorithm have 1 page replacement. Hence, there will be no performance improvement, and Dr. X cannot advertise this application.

Second Test case

For the second test case, there are only two pages which can be allocated and the size of each page is 128 bytes. The addresses 0, 255, 127, 256, 60, 1024 and 120 correspond to the 0th, 1st, 0th, 2nd, 0th, 8th and 0th page. (Note that the floor(127/128), floor(60/128), and floor(120/128) are all equal to 0. Similarly, the floor(255/128) = 1).

The page replacements are shown with asterisks (*) in the following tables.

FIFO Page Replacement:

Cycle	0	1	2	3	4	5	6	7
New Data	-	0	1	0	2	0	8	0
OS Page 0	-	0	0	0	2*	2	8*	8
OS Page 1	-	-	1	1	1	0*	0	0

LRU Page Replacement:

```
    Cycle                       0       1       2       3       4       5       6
7
    New Data                    -       0       1       0       2       0       8
0
    OS Page 0                   -       0       0       0       0       0       0
0
    OS Page 1                   -       -       1       1       2*      2       8*
8
```

In case of FIFO page replacement, there are 3 page replacements. In case of LRU page replacement, there are 2 page replacements. Hence, there will be improved performance in this case, and Dr. X can advertise this application.

Pirates

Fierce pirates have just arrived in the archipelago. They are searching for a hidden treasure and only have a map to help them. The map looks like a matrix with N rows and M columns with every cell containing one of the two symbols: 'O' or '~'. A land cell is represented by the symbol 'O' and a sea cell by '~'. Two land cells are part of the same island if there is a way from one to the other walking only on land cells, and from a cell one can walk in all 8 directions. The pirates are in cell $(x1, y1)$ and the treasure is in cell $(x2, y2)$, both of which are sea cells. To get to the treasure, the pirates may need to cross some islands. The police only watch over the land cells, and thus you have to help the pirates find a path to the treasure crossing a minimal number of islands. You need to help the pirates Q times.

Note:

- The top left cell is designated as (1,1), and the bottom right cell is (N, M).
- Crossing an island occurs whenever the pirates go from a sea cell to a land cell. If the pirates cross the same island multiple times, it should be counted that many times.

Input Format

The first line of input contains three space-separated integers, N, M and Q.
The next N lines contain the description of the map.
The last Q lines contain the queries, in the form of four space-separated integers, $x1$, $y1$, $x2$, and $y2$.

Constraints

$1 \le N, M \le 1000$
$1 \le Q \le 10^5$
$1 \le x1, x2 \le N$
$1 \le y1, y2 \le M$

Output Format

For each query, you should output, on a line by itself, the minimum number of islands that must be traversed when travelling from $(x1, y1)$ to $(x2, y2)$.

Sample Input

```
4  12  2
00000~~00000
0~~00~00~~~0
```

```
00~00~~0~0~0
000000~00000
2 2 3 11
4 7 3 9
```

Sample Output

```
2
1
```

Explanation

The map for this test case consists of two islands.

In the first query, the pirates begin in the water body that is surrounded by the island to the left, and end in the water body surrounded by the island to the right. They must cross both these islands to get from start to finish.

For the second query, they begin in the body of water between the two islands, and end in the body surrounded by the island to the right. They must cross the island to the right to reach the treasure.

Counting Molecules

Your task is to count the number of molecules in a cup of soda which contains distilled water, carbon dioxide, and glucose. You have a machine that counts the number of atoms of carbon, hydrogen, andoxygen in a given sample.

Input Format

The input consists of a single line with three space separated integers: c, h, and o

where

c is the count of carbon atoms
h is the count of hydrogen atoms
o is the count of oxygen atoms

Constraints

$0 \leq c, h, o < 10^{10}$

Output Format

If the number of atoms **is consistent** with a mixture containing **only water, carbon dioxide, and glucose molecules**, the output should consist of a single line containing three space separated integers: the number of water molecules, the number of carbon dioxide molecules, and the number of glucose molecules.
If the number of atoms **is not consistent** with a mixture containing **only water, carbon dioxide, and glucose molecules**, the output should consist of a line containing the word `Error`

Sample Input

```
10 0 20
```

Sample Output

```
0 10 0
```

Explanation

The input indicates that there are 10 carbon atoms and 20 oxygen atoms. The only way that this could occur would be if there were 0 water molecules, 10 carbon dioxide molecules, and 0 glucose molecules.

Note that there are additional sample inputs available if you click on the `Run Code` button.

Checkers Challenge

Watch the following YouTube video clip. Your task is to compute the number of possible ways the white player can win from an opening state of a single white piece in a game of Turkish Draughts. For more information on the game, you can view the Wikipedia page.

For this challenge, we will use the following variation on the official rules:

1. The black pieces can be arbitrary placed, and will not necessarily be located at places reachable in a legal game
2. A single white piece is a king if, and only if, it is placed in or reaches the top most line. Once a piece is a king it remains a king throughout.
3. A white piece can capture by jumping over a single black piece to the left, right or upwards, landing in the adjacent square
4. A white king can capture by jumping left, right, upwards or backwards and can skip arbitrary number of blank squares before and after the black piece
5. After capturing a black piece, the white piece (or king) must turn 90 degrees or keep moving in the same direction (no 180 degree turns are allowed).
6. We ask for the number of different ways the white player can win a single move. White wins by capturing all black pieces.

Input Format

Each input begins with an integer t, on a line by itself, indicating how many testcases are present.
Each testcase will contain 8 lines with the state of the board. The board will have a single white piece `o`, some black pieces `x`, and empty places `.`. White's side of the board is at the bottom of the board. So if the white piece were to reach to top row of the board, it would become a king.

In between each testcase is a blank line.

Constraints

$1 \le t \le 5$

There will always be at least 1, and no more than 16, black pieces in each game.

The game board will always be 8x8 squares in size.

Output Format

For each testcase, output, on a line by itself, the number of possible ways the white can win, or `0` if he cannot.

Sample Input

```
3
.......o
.x.x.x..
xxxx.xx.
........
........
.x.xx..x
x.......
..x...x.

........
........
....o...
........
....x...
........
........
........

...o....
........
...x....
........
........
........
........
........
```

Sample Output

```
12
0
5
```

Explanation

The first testcase is the state of the board in the 56th second of the YouTube video. There are 12 ways in which this game can be won. These ways are represented below:

1. down 7, left 3, up 6, left 2, down 4, right 4, up 4, left 3, down 4, left 3, up 4, right 5, down 6, left 5, up 5, right 2

2. down 7, left 3, up 6, left 2, down 4, right 4, up 4, left 3, down 4, left 3, up 4, right 5, down 6, left 5, up 5, right 3
3. down 7, left 3, up 6, left 2, down 4, right 4, up 4, left 3, down 4, left 3, up 4, right 5, down 6, left 5, up 5, right 4
4. down 7, left 3, up 6, left 2, down 4, right 4, up 4, left 3, down 4, left 3, up 4, right 5, down 6, left 5, up 5, right 5
5. down 7, left 3, up 6, left 2, down 4, right 4, up 4, left 3, down 4, left 3, up 4, right 5, down 6, left 5, up 5, right 6
6. down 7, left 3, up 6, left 2, down 4, right 4, up 4, left 3, down 4, left 3, up 4, right 5, down 6, left 5, up 5, right 7
7. down 7, left 3, up 6, right 2, down 4, left 4, up 4, right 3, down 4, left 5, up 4, right 3, down 6, left 3, up 5, right 2
8. down 7, left 3, up 6, right 2, down 4, left 4, up 4, right 3, down 4, left 5, up 4, right 3, down 6, left 3, up 5, right 3
9. down 7, left 3, up 6, right 2, down 4, left 4, up 4, right 3, down 4, left 5, up 4, right 3, down 6, left 3, up 5, right 4
10. down 7, left 3, up 6, right 2, down 4, left 4, up 4, right 3, down 4, left 5, up 4, right 3, down 6, left 3, up 5, right 5
11. down 7, left 3, up 6, right 2, down 4, left 4, up 4, right 3, down 4, left 5, up 4, right 3, down 6, left 3, up 5, right 6
12. down 7, left 3, up 6, right 2, down 4, left 4, up 4, right 3, down 4, left 5, up 4, right 3, down 6, left 3, up 5, right 7

There is no way for white to win the second testcase.

For the final testcase, white has a king, and white can capture the single black piece, and land on any of the five spaces below the piece.

Mancala'h

You and your friend Alessandro are taking part to an archaeological mission that aims to explore a newly discovered tomb of an ancient pharaoh in Egypt. After an adventurous trip through tunnels, doors and rooms, you and your fellow archaeologists arrive in front of a huge closed door and find a mysterious artefact that appears to be a sort of puzzle for opening the door. The artefact is composed of some inscriptions and a massive wooden disc with many bins carved at the perimeter. Some of the bins contain seeds.

On a side of the artefact, an inscription says: "Harvest all the seeds in the first bin. Sow them in the following bins, one-by-one. Rotate the disc by one step. Repeat."

On the other side of the artefact, a second inscription says: "How far you can go before starting to endless repeat the same harvesting pattern? How far can you go backwards?"

After a fellow archaeologist translated the inscriptions, your friend Alessandro exclaims: "I know this game: it's *Mancala'h*! I know that from every possible configuration, the game evolves to a periodic status, meaning that at some point you start repeating the same pattern. For example, the configuration [1, 1, 1] evolves to [0, 2, 1] because you take the seed in the first bin and put it in the second bin. Then, after a clockwise rotation of the disc, the second bin with two seeds becomes the first and we can remove the leading zero from notation. So, the configuration is now [2, 1], which evolves to [2, 1] itself, because you take the two seeds in the first bin and put one in the second bin and one in the third bin, which was empty. In this case we can say that the depth of the original [1, 1, 1] configuration is equal to 1, meaning that in a single step we reach a periodic status."

"The situation can be more complex, for example the configuration [4] evolves to [1, 1, 1, 1], which in turn evolves to [2, 1, 1], which evolves to [2, 2], which evolves to [3, 1], which evolves again to [2, 1, 1], and so on. In this case the depth of the configuration [4] is 2 and the period is 3 steps long."

"The first thing we need to find is exactly the depth of the configuration we have found here."

"Going backwards in the game evolution is not as easy as going forward: for example, we have seen that configurations [1, 1, 1, 1] and [3, 1] evolve both to [2, 1, 1] in one step. On the contrary, the configuration [1, 2] cannot be the evolution of

any valid configuration, because we *exclude* all the so-called *not connected*configurations, which are those that contain a zero, such as [1, 0, 2]."
"The second thing we need to find, in fact, is the height of the given configuration, which is equal to the maximum number of backwards steps or, in other words, the distance (in terms of number of steps) of the farthest previous configuration. We cannot iterate backwards over the period, so the height is the length of the longest sequence of *unique* configurations that leads to the current configuration."
Help the archaeologists to solve the mystery by finding the depth D and the height H of given Mancala’h configurations.

Input Format

The input contains a single Mancala’h configuration. A Mancala’h configuration is defined by the sequence [N_1, N_2, N_3, ..., N_L] of L ($1 \leq L \leq 100$) integers separated by a blank-space. Each integer N_i ($1 \leq N_i \leq 500$) represents the number of seeds in the ith bin.

Constraints

The size of the Mancala'h board is large enough that you will never have a board in which all of the bins are filled. In other words, there are always more bins than seeds.

The number of unique configurations reachable from any input (either forwards or backwards) is at most $5 * 10^6$.

Output Format

The output is a single line containing two integers D and H, separated by a blank-space. The first integer $D$is the depth of the Mancala’h configuration specified in the input. The second integer H is the height of the Mancala’h configuration specified in the input.

Sample Input

```
2 1 1
```

Sample Output

```
0 3
```

Explanation

The depth of the Mancala’h configuration [2, 1, 1] is 0 because the period includes the given configuration:

```
[2, 1, 1] → [2, 2] → [3, 1] → [2, 1, 1]
```

The height is 3 because there are three backwards evolutions, two of which contain 3 steps:

```
[2, 1, 1] ← [1, 1, 1, 1] ← [4] ← [1, 3]
[2, 1, 1] ← [3, 1] ← [2, 2] ← [1, 1, 2]
```

```
[2, 1, 1] ← [3, 1] ← [1, 2, 1]
```

Note that for the following is not a valid backwards evolution because it repeats the configuration [2, 1, 1]:

```
[2, 1, 1] ← [3, 1] ← [2, 2] ← [2, 1, 1]
```

As an additional example, suppose that the input was:

```
3 1 2 3 2
```

For this Mancala'h, the depth is 6:

```
[3, 1, 2, 3, 2] → [2, 3, 4, 2] →[4, 5, 2] →
[6, 3, 1, 1] → [4, 2, 2, 1, 1, 1] → [3, 3, 2, 2, 1] →
[4, 3, 3, 1] → [4, 4, 2, 1] → [5, 3, 2, 1] →
[4, 3, 2, 1, 1] → [4, 3, 2, 2] → [4, 3, 3, 1]
```

The height is 1 because the only previous configuration possible is:

```
[3, 1, 2, 3, 2] ← [1, 2, 1, 2, 3, 2]
```

Game of Stones 1

Alice and Bob play a game. The game is turn based: Alice moves first, then Bob, and so on. There are Npiles of stones; in every pile there is an odd number of stones. At every turn, the one to play must pick a pile and splits it into 3 piles with an odd number of stones each.

The player who cannot split any pile loses. As this game is too simple for both of them, they decided to play multiple games in parallel. The rules remain the same, but at every turn, the one to play must first pick a game and then split a pile only in that game. The one who loses is the one that can't split any pile in any game, i.e. all the piles in all the games have only 1 stone. Bob still thinks that he is at a disadvantage, since he is the second to move. Your task is to find the winner if both the players play optimally.

Input Format
The input begins with an integer T, giving the number of test cases in the input.
Each testcase begins with an integer G, on a line by itself, giving the number of games to be played in parallel.
The G games are then described in two lines as follows: The first line gives the number of piles in the game, and the second contains the number of stones in each of the piles.

Constraints

1 <= T <= 10

1 <= [Number of piles in all games in a testcase] <= 10^5

1 <= [Number of stones in a pile] <= 10^9

Output Format
For each testcase, output the winner, i.e. either `Alice` or `Bob`, on a line by itself.

Sample Input

```
2
2
3
1 3 5
2
3 7
1
5
1 3 5 7 9
```

Sample Output

```
Alice
Bob
```

Explanation

The sample input can be annotated as follows:

```
2 (the number of tests)
2 (the number of parallel games for the first test)
3 (the number of piles in the first game)
1 3 5
2 (the number of piles in the second game)
3 7
1 (the number of parallel games for the second test)
5 (the number of piles)
1 3 5 7 9
```

Mysterious Maze

There is a valuable treasure that is hidden below a complex maze. The maze is made of rooms and are square in shape, and a maze of size N will have $N \times N$ rooms with all of them closed initially. When a room is open, one can enter into it from any of its adjacent open rooms; two rooms are adjacent if they share a common wall.

The maze was built in a way that it opens itself by opening its rooms randomly. A maze is said to be open if there is at least one path from any one of the rooms facing the top of the maze to any room on the bottom side facing the treasure. Anyone, who attempts to enter the maze without being able to reach the treasure and return, will be cruelly killed by the maze.

The local government has spent years researching the maze and figured out a way to determine the sequence of rooms being opened in almost real time. Based on this data, the government has posed the following challenge, with a small percentage of the treasure to whomever solves the problem:

Given the sequence of room openings, determine when the maze becomes open, or if it remains closed throughout.

Input Format
Input begins with a single integer N, which denotes the size of maze.
All of the next lines (except the last one) denotes the sequence of the rooms the maze is opening. Each line contains 2 integers X and Y which denotes the row and column of the room opened by the maze. The last line just includes -1 and marks the end of input.
Constraints
1 <= X, Y <= N <= 1000
Output Format
Output a single integer R based on the final status of the maze. R denotes the number of room openings that occur before the maze first becomes open, or -1 if the maze remains closed.
Sample Input
```
4
1 4
2 3
3 2
4 3
4 1
2 1
1 1
```

```
-1
```

Sample Output

```
-1
```

Explanation

It is easy to understand if you plot the maze. The following is the state of the maze at the end of the inputs. `X` indicates that a room is closed and `O` that a room is open. Note that there is no path from the top of the maze to the bottom of the maze.

```
O X X O
O X O X
X O X X
O X O X
```

Consider the second input sample (which is available if you click on the `Run Code` button):

```
4
1 4
2 3
3 2
4 3
4 1
2 1
1 1
3 1
3 4
2 2
-1
```

Below is a figure with the maze after 7 rooms are open. Note that there is no path from the top of the maze to the bottom of the maze, and therefore the maze is closed.

```
O X X O
O X O X
X O X X
O X O X
```

However, after 8th room is open, there is a path, as shown below:

```
O X X O
O X O X
O O X X
O X O X
```

Thus, the expected output is:

```
8
```

Safety

Vangelis the bear wants to create a tool that will make his passwords stronger. In order to do so, he thought of some transformations, that should make his passwords stronger when applied, and a verification method to check if his tool is doing its job as expected.

Vangelis improvised three kinds of commands for his tool:

1. Check if the substring that starts at position i and ends at position j (inclusive) of the current password is equal to the substring that starts at position k of the current password and has length j - i + 1 (it is guaranteed that this substring exists). If the answer is yes, print 'Y', else print 'N'. The input format of the command is: `1 i j k`
2. Replace the substring that starts at position i and ends at position j (inclusive) of the current password, with the substring that starts at position k of the *original* password and has length j - i + 1 (it is guaranteed that this substring exists). The input format of the command is: `2 i j k`
3. Replace each letter in the string that starts at position i and ends at position j (inclusive) of the current password with the next letter of the Latin alphabet, except if the input letter is 'z' where it would be replaced with 'a'. Examples, 'a' will be replaced by 'b', 'b' will replaced by 'c', 'z' will be replaced by 'a' etc. The format of the command is: `3 i j`

Please note that these operations do not increase the size of the password and that all indices start from 1.

Before he starts coding, Vangelis wants you to create a prototype application that will perform this basic functionality.

Given a password that is composed of N ($1 \le N \le 300{,}000$) lowercase Latin characters, you will execute a series of commands on the password, including transforming the password with type 2 and type 3 commands, and printing the result of type 1 commands.

Input Format

The first line contains the original password.

The second line is an integer M, $1 \le M \le 300{,}000$, that represents the number of operations that will be given to your program.

Lines 3 to $M + 2$ contain the input information for one of the command types.

Note:

Some of the test cases are very large, and may require you to speed up input handling in some languages.

In C++, for example, you can include the following line as the first line in your main function to speed up the reading from input:

```
std::ios_base::sync_with_stdio (false);
```

And in Java, you can use a BufferedReader to greatly speed up reading from input, e.g.:

```
BufferedReader reader = new BufferedReader(new InputStreamReader(System.in));
// Read next line of input which contains an integer:
int T = Integer.valueOf(reader.readLine());
```

Output Format

For each type 1 command, print, on a line by itself, the output of the command.

Sample Input

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

Sample Output

```
N
Y
N
N
```

Explanation

The first command compares the `bbbb` with `bbbx`, and since they are not equal, the program should output `N`.
The second command replaces the substring from position 2 to 5 with the substring from position 7 to 10*in the original password*, and thus the password is now `bzbzcrzbzcj`.

The third command compares the substring from position 2 to 6 with itself, and thus the expected output is `Y`.
The fourth command compares the substring `zbz` with the substring `bzc`, and thus the output should be`N`.
The fifth command shifts the characters in the substring from position 2 to 5, changing the password to`bacadrzbzcj`.
The last command compares the substring `bac` with the substring `zcj`, and outputs `N`.

Forum Threads

Consider a list of posts in an internet forum divided in pages. In order to keep the pages as aesthetically pleasant as possible, the graphics team came up with the optimal number of posts in every page. Then the user experience team decided that it would be convenient to group posts into threads, this is, list all replies to a post (and the replies to those replies) consecutively, and that a thread should not be split in different pages. Now you should find a layout for the posts such that

1. posts in the same thread are listed consecutively;
2. threads are ordered by the time of the first post;
3. posts in the same thread are in the same page; and
4. the maximum difference between the number of posts per page and the intended number is minimal.

Note: the goal is to minimize how bad the worst page is, not the sum of how bad all of the pages are.

Input Format

The input consists of a number of test cases.

The first line of each test case contains two integers a, p, the intended number of posts per page and the number of posts respectively. Then there is a line for each post, in the order they were posted. The i^{th} line contains a positive integer x if the i^{th} post is a reply to the x^{th} post, or 0 if the i^{th} post starts a new thread.

Constraints

$0 \leq a, p \leq 1\,000$
$0 < x < i$

Output Format

Output one line per test case with one integer, the maximum difference between the number of posts per page and the intended number in a layout that satisfies the conditions.

Sample Input

```
3 6
```

```
0
1
2
0
4
5
3 6
0
0
0
0
0
3
```

Sample Output

```
0
1
```

Explanation

In the first test case there are 6 posts that are to be distributed at a ratio of 3 posts per page. There are 2 threads with 3 posts each, so a layout with the first thread in the first page and the second thread in the second page fits perfectly.

In the second test case there are 2 single posts, a thread with 2 posts, and 2 more single posts. Since the middle thread cannot be split between the 2 pages, a layout with a one post imbalance is optimal. The optimal configuration may not be unique. In fact in this test case, there are three configurations that are optimal:

- Posts 1–3 could be put in the first page, and posts 4 and 5 in the second. In this case, the first page would contain one extra post (6, which is a reply to post 3), and the second page would contain one fewer.
- Posts 1 and 2 could be put in the first page, and posts 3–5 in the second.
- Posts 1 and 2 could be put in the first page, post 3 (with its reply) in the second, and posts 4 and 5 in the third.

Island Hopping 1

Bob is the captain of a boat in the island nation of Artskjid. Most of the time he cannot go directly from his starting island to the ending island because his boat cannot hold enough fuel for the entire journey, and so he must make stops to refuel.

Bob's boat holds up to 100 units of fuel. There is an additional complication in that fuel is rationed on the islands, and the maximum amount of fuel that he can receive at each stop varies from island to island. When stopping at an island, the fuel taken from previous stops at that island do not affect the current one. Furthermore, the fuel ration for an island resets every time Bob stops there.

While he can visit an island multiple times on the journey (in order to get fuel), he cannot visit the same island consecutively. He must go to one or more different islands, before returning to an island.

Bob wants to know the minimum amount of fuel he needs in order to complete his journey.

Input Format
The input starts with a line containing an integer t, $1 \leq t \leq 10$, which gives the number of test cases in the input.
Each test case begins with a line containing an integer n, where n is the number of islands.
Then there are n lines containing a description of the islands in the form:

```
[name] [fuel]
```

where `[name]` is the name of the island, in the form of a string made up only of letters (no spaces), and `[fuel]` is an integer indicating the amount of fuel available, per stop, on the island.
On the next line is an integer m, where m is the number of navigable channels between islands.
Then there are m lines containing a description of the available paths between islands:

```
[island1] [island2] [fuel_needed]
```

where `[island1]` and `[island2]` are island names that appeared in the first n lines of the testcase, and `[fuel_needed]` is an integer indicating the amount of fuel needed to travel between these islands. Note that travel can occur in either direction, i.e. from `[island1]` to `[island2]` or from `[island2]` to `[island1]`.

Bob will always start with the island named `start`, and end at the island named `end`. Note that he starts with an empty tank of gas, so the `[fuel]` listed for the `start` island gives Bob's initial amount of fuel.

Constraints

$2 \le n \le 50$

$0 \le m \le 500$

$0 \le$ `[fuel]` ≤ 100

$0 \le$ `[fuel_needed]` ≤ 200

Output Format

For each test case, you should output, on a line by itself, either the minimum units of fuel needed for Bob to get from the starting point to the ending point, or `Impossible` if there is no way for Bob to reach the ending point from the starting point.

Sample Input

```
2
3
start 2
end 0
midway 50
3
start midway 1
end midway 90
start end 99
5
start 1
end 0
amity 2
atlantis 3
azkaban 4
5
start end 101
start amity 1
atlantis amity 2
azkaban atlantis 3
azkaban start 1
```

Sample Output

```
93
Impossible
```

Explanation

In the first test case, Bob can start with 2 units of fuel, and go to `midway` island. Since this trip uses 1 unit of fuel, he arrives with 1 unit of fuel, and then adds 50 more, for a total of 51 units of fuel.

He still cannot reach the final island. Instead, he returns to `start` island (with 50 units of fuel upon arrival). He adds 2 more units of fuel, for a total of 52 units of fuel, and then he goes back to `midway`.

He arrives with 51 units of fuel, and then fills up his tank so that it now holds 100 units of fuel. He now has enough fuel to make it to the destination, using 90 more units of fuel.

There are no other trips that require less fuel, so you would output 93, the total of units of fuel that was used.

For the second test case, there is no way to reach the `end` island with fuel that fits in Bob's tank.

Full Adder

We would like your help to create a basic adder. However, this adder, should work in any base, with any set of symbols.

Input Format
The first line of input contains an integer b, a space, and a list of b symbols that make up the base. The symbols are listed in order from the least significant symbol to the most significant symbol. In other words, the first symbol listed corresponds to 0, the second corresponds to 1, etc. These symbols can be numbers, uppercase letters, or lowercase letters.

The remaining lines contain the addition problem to be solved, as shown in the sample input and output. The operands will be non-negative numbers expressed in the given base. Note that the last line contains question marks which must be replaced with the correct value.

Constraints
$2 \leq b \leq 62$
The numbers to be added can contain up to 10^7 symbols.
Output Format

The first four lines of output should be identical to the input. The last line should contain the solution to the problem, with the answer aligned appropriately.

Sample Input
```
10 0123456789
   752
+76045
------
   ???
```
Sample Output
```
10 0123456789
   752
+76045
------
 76797
```
Explanation

The first sample corresponds to a normal base-10 addition problem.

Additional sample problems are available if you click on the `Run Code` button.

The second sample problem has the following input:

```
10 wj8Ma04HJg
    H
+8J4J
-----
  ???
```

This is a base-10 problem with different symbols. `H` corresponds to the digit `7` and `8J4J` is the number`2868`. When adding these numbers, the result is `2875`, which is represented as `8JH0` in the given base. Thus the expected output is:

```
10 wj8Ma04HJg
    H
+8J4J
-----
 8JH0
```

Finding Shelter

A group of N soldiers found themselves in a dangerous zone and are under heavy fire from the enemy. Gigel, the commander, has a map with the position of the soldiers and knows the coordinates of N safety shelters with a capacity of one person each. Gigel wants to make a plan to save the soldiers with the lowest risk. The risk of a soldier being hurt is the distance between him and his assigned shelter. Gigel wants each soldier to have a good chance of surviving, so the maximal distance between a soldier and his assigned shelter should be as low as possible. If there are multiple solutions, he also wants to minimize the sum of distances to lower the total risk.

Given N, the positions of the N soldiers, and the position of the N shelters, you have to assign a shelter to each soldier. Find the lowest maximal distance and the corresponding sum of distances.

Input Format

The first line contains the integer N, on a line by itself.

The next N lines contain two space-separated floating point numbers, with the i^{th} line giving the x and $y$coordinates for the i^{th} soldier.

The next N lines contain two space-separated floating point numbers, with the i^{th} line giving the x and $y$coordinates for the i^{th} shelter.

The floating point numbers will not have more than three digits after the decimal point.

Note: The distance between a soldier and a shelter is equal to the Euclidean distance between their coordinates.

Constraints

$1 \le N \le 500$

$0 \le x, y \le 1000$

Output Format

The first line of output should contain the maximal distance between a soldier and his shelter.

The second line of output should contain the sum of the distances that all soldiers must travel.

Note that these numbers must be within 10^{-4} of the expected output.

Sample Input

```
4
0.0 0.0
```

```
0.0 1.0
1.0 0.0
1.0 1.0
2.0 0.0
0.0 2.0
0.0 1.0
1.0 0.0
```

Sample Output

```
1.00000
4.00000
```

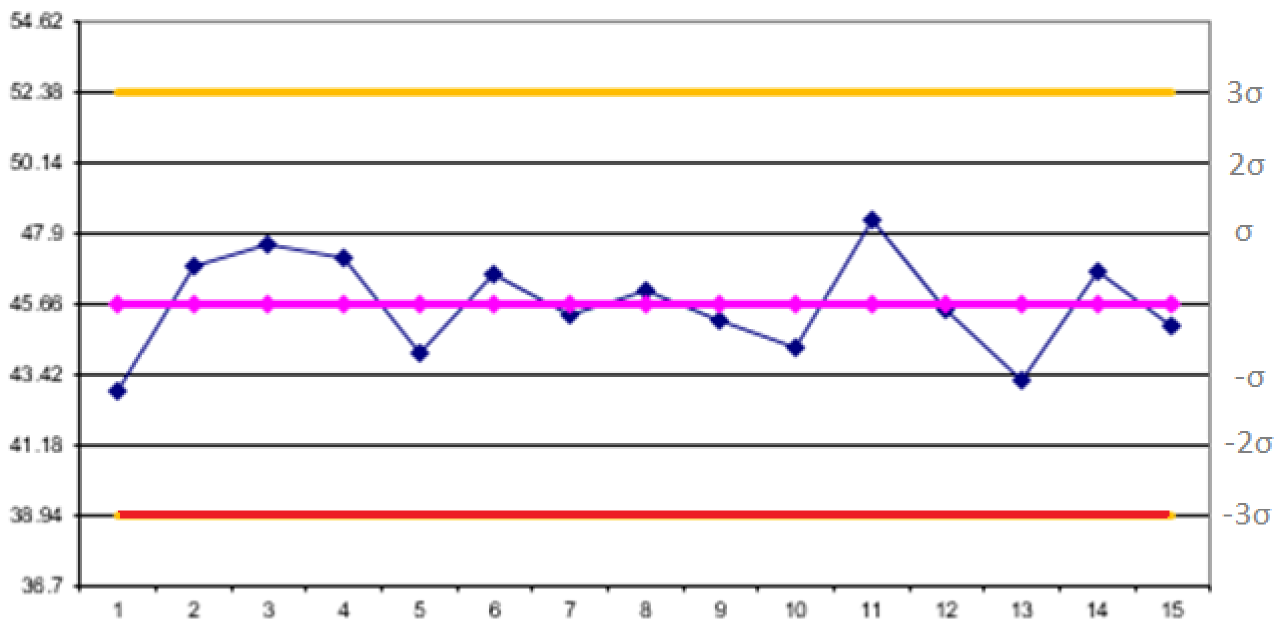
Explanation

You can assign soldier 1 to shelter 3, soldier 2 to shelter 2, soldier 3 to shelter 1 and soldier 4 to shelter 4. There is no other assignment with either lower maximal distance, or with equal maximal distance but lower sum of distances.

Always Be In Control

Engineers use a technique called "statistical process control" to manage and improve engineering processes. For example, suppose a manufacturing process is producing widgets of some sort and the diameter of a widget, measured in microns, is important to the ability to use that widget in a later assembly. Many things can affect the diameter of a widget (humidity, temperature, quality of raw materials, etc.), so there is going to be some variation in diameter from one set of widgets to the next. Statistical process control would sample the diameter of a widget over time to make sure that the variation is consistent.

One of the core techniques of statistical process control is a control chart, which is used to monitor some aspect of the process over time to see if the process is behaving consistently. A control chart plots the sampled statistic over time and includes upper and lower control limits that describe the variation in the data. Those control limits are called 3-sigma limits as they represent about three standard deviations away from the mean of the data. Here is an example control chart:



🟪 Center Line CL_X

🟥 Lower Control Limit LCL_X

🟨 Uper Control Limit UCL_X

🟦 Process Variable

⬛ Sigma limits

A process is considered to be "in control" with respect to a given variable if its variation is predictable. When analyzing a control chart, the process is out of control if any of the following occur:

1. A single point falls outside the 3-sigma control limits.
2. At least two out of three successive values fall on the same side of, and more than two sigma units away from, the center line.
3. At least four out of five successive values fall on the same side of, and more than one sigma unit away from, the center line.
4. At least eight successive values fall on the same side of the center line.

There are many ways to build control charts and selecting the right one depends on the type of data you have and the question you are trying to answer. For this problem, you are going to build a variation of an Xbar chart, in which we group the data into subgroups of n sequential values. For each subgroup, we compute r_i, the range of the values, and X_i, the average of the values. (The range is the maximum value minus the minimum value in the subgroup). The control chart will be a plot of the raw data values (in order). The upper control limit (UCL), lower control limit (LCL), and the center line (CL) are computed as follows:

$UCL_X = X_{ave} + A_2 R_{ave}$

$LCL_X = X_{ave} - A_2 R_{ave}$

$CL_X = X_{ave}$

Where X_{ave} is the average of the X_i values, R_{ave} is the average of the range values, and A_2 is a constant that depends on the size of the groups we created, as shown in the table below.

Size of group (n)	A2
2	1.880
3	1.023
4	0.729
5	0.577
6	0.483
7	0.419
8	0.373
9	0.337
10	0.308

Input Format

The first line of the input will be an integer between 1 and 20, inclusive, that is the number of test cases in the input.

Each test case will be specified by one line of space separated integers. The first will be x, 1 ≤ x ≤ 10,000, the number of data points in the test case. The second will be n, 2 ≤ n ≤ 10, the number of elements in a subgroup. That will be followed by x space separated integers for the test case containing the sequential data gathered from an engineering process. These will be integers with values between -10,000 and 10,000, inclusive.

The last subgroup may be incomplete (i.e. it may not contain n elements). The last subgroup should be treated like a normal subgroup, even if it is incomplete. For example, let's say the subgroup had the entries <1,6,2>. If n = 10, this subgroup is incomplete. The range would be 5 (6 - 1 = 5), and the average would be 3 ((1 + 6 + 2)/3 = 3). If there is only 1 item in this subgroup, the average would be equal to the number, and the range would be 0.

Output Format

You are to calculate the three sigma control limits and then test the data to see if it is in control or out of control. For each test case, output, on a line by itself, either "In Control" or "Out of Control" as appropriate.

Note that the output is case-sensitive.

Sample Input

```
1
25 5 -13 -18 4 15 -3 10 9 -1 17 -1 -2 20 -20 10 -4 2 2 -5 -1 -14 4 -9 13 4 12
```

Sample Output

```
Out of Control
```

Explanation

The table below shows the necessary calculations for these 25 data points, given that there are 5 items in a subgroup.

```
DATA     SUBGROUP AVERAGE     SUBGROUP RANGE
 -13
 -18
   4
  15
  -3              -3                33
  10
   9
  -1
  17
  -1             6.8                18
  -2
  20
 -20
  10
  -4             0.8                40
   2
   2
  -5
  -1
 -14            -3.2                16
   4
  -9
  13
   4
  12             4.8                22

GRAND AVERAGE   1.24              25.8


UCL          16.1266
```

```
CENTER LINE     1.24
LCL          -13.6466

SIGMA          4.9622
```

For these calculations, A_2 is 0.577 because we grouped five items in a group. As shown in the table, X_{ave} is 1.24, and R_{ave} is 25.8. Since the control limits are "3-sigma" lines, sigma is one third of the distance between the center line and the upper control limit.

This process would be considered out of control because there are a number of points, e.g. `-18` and `20`, that are more than three sigma from the center line. Note that in a real world analysis, you would need much more data to draw this conclusion.

Flower Games

Joy and her friends found a flower with N petals and want to play a modified version of the He loves me... he loves me not game. The girls number the petals with numbers from 1 to N in the clockwise direction. They will traverse the petals in circular order starting with 1, then 2, ..., then N, then 1... At the first petal, they will shout "He loves me", at the second "He loves me not" and tear it, at the third "He loves me", at the fourth "He loves me not" and tear it. The girls will continue the game until there is only one petal left. The task is to identify the number of the last petal.

Input Format

The input begins with an integer T, giving the number of test cases in the input.
Each testcase consists of an integer N, on a line by itself.

Constraints

1 <= T <= 100000

1 <= N < 2^63

Output Format

The location of the last petal, on a line by itself.

Sample Input

```
4
2
3
4
6
```

Sample Output

```
1
3
1
5
```

Explanation

There are four test cases in the input.

With 2 petals, one would skip 1, tear 2, and then only 1 is left.

With 3 petals, one would skip 1, tear 2, skip 3, tear 1, and then only 3 is left.

With 4 petals, one would skip 1, tear 2, skip 3, tear 4, skip 1, tear 3, and then only 1 is left.

With 6 petals, one would skip 1, tear 2, skip 3, tear 4, skip 5, tear 6, skip 1, tear 3, skip 5, tear 1, and then only 5 is left.

Ellipse Art

In IEEEXtreme 9.0, you met the famous artist, I.M. Blockhead. This year we want to introduce you to another famous artist, Ivy Lipps. Unlike I.M., Ivy makes her art by painting one or more ellipses on a canvas. All of her canvases measure 100 by 100 cms.

She needs your help. When she is done with the painting, she would like to know how much of the canvas is unpainted.

Input Format

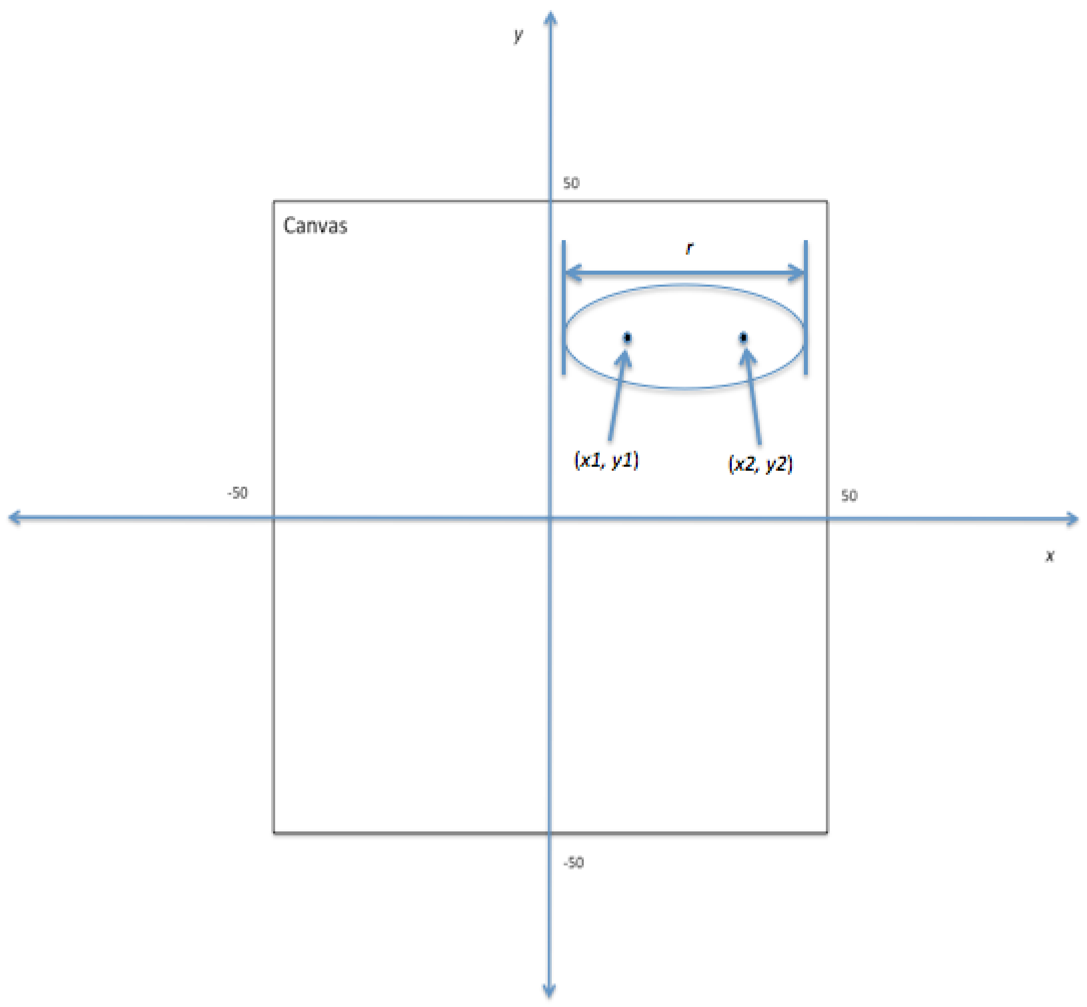
The first line of input contains t, $1 \le t \le 8$, which gives the number of test cases.
Each test case begins with a single integer, n, $1 \le n \le 40$, which indicates the number of ellipses that Ivy has drawn.
The following n lines give the dimensions of each ellipse, in the following format:

```
x1 y1 x2 y2 r
```

Where:

- $(x1, y1)$ and $(x2, y2)$ are positive integers representing the location of the foci of the ellipse in cms, considering the center of the canvas to be the origin, as in the image below.
- r is a positive integer giving the length of the ellipse's major axis

You can refer to the Wikipedia webpage for background information on ellipses.



Coordinate system for the canvas.

Constraints

$-100 \leq x1, y1, x2, y2 \leq 100$
$r \leq 200$
$r \geq ((x2 - x1)^2 + (y2 - y1)^2)^{1/2} + 1$

Note that these constraints imply that a given ellipse does not need to fall completely on the canvas (or even on the canvas at all).

Output Format

For each test case, output to the nearest percent, the percent of the canvas that is unpainted.

Note: The output should be rounded to the nearest whole number. If the percent of the canvas that is unpainted is not a whole number, you are guaranteed that the percent will be at least 10% closer to the nearer percent than it is from the second closest whole percent. Therefore you will not need to decide whether a number like 23.5% should be rounded up or rounded down.

Sample Input

```
3
1
-40 0 40 0 100
1
10 50 90 50 100
2
15 -20 15 20 50
-10 10 30 30 100
```

Sample Output

```
53%
88%
41%
```

Explanation

The ellipse in the first test case falls completely within the canvas, and it has an area of approximately 4,712 cm^2. Since the canvas is 10,000 cm^2, 53% of the canvas is unpainted.

In the second test case, the ellipse has the same size as in the first, but only one quarter of the ellipse is on canvas. Therefore, 88% of the canvas is unpainted.

In the final testcase, the ellipses overlap, and 41% of the canvas is unpainted.

N-Palindromes

Alice thinks that contest problem authors' obsession with palindromes is misplaced. She is much fonder of n-palindromes, which are words that are palindromes when the characters at exactly n positions are changed.

For example, Alice knows that her name (in lowercase) is a 2-palindrome, because she can create any of the following palindromes from her name by changing 2 characters: `alila`, `acica`, `elile`, `ecice`.

She also knows that her name is a 3-palindrome, because she can create palindromes by changing characters at 3 positions, e.g. `ecace` and `zlilz`. However, this is only a partial list, and she wants your help in determining the total number of such palindromes.

Note that the characters of an n-palindrome, including the n replacement characters, must all be lowercase English letters.

Input Format

The input starts with an integer t, on a line by itself, which gives the number of test cases.

Each test case is made up of an integer n followed by a lowercase string.

Constraints

$1 \le t \le 20$

$1 \le n \le$ [length of string] ≤ 500

Output Format

For each test case, you should output, on a line by itself, the total number of palindromes that can be created by changing exactly n characters of the given string. Since this number may be very large, you should output the number modulo (10^9 + 7).

Sample Input

```
3
2 alice
1 racecar
3 alice
```

Sample Output

```
4
25
196
```

Explanation

The problem statement lists the four palindromes that can be made from the string `alice`, by changing 2 characters.

Since you can only change one character in `racecar`, you are constrained to changing the middle letter. This character can be changed to any of the 25 letters other than `e`.

For the last testcase, Alice has found that there are 196 palindromes that can be made from her name, by changing 3 characters.

Binary Quilts

Binary Quilts Incorporated uses custom machinery to make square blankets that consist of 25 black or white squares that are sewn together. Whenever they want to change designs for the blankets, they are limited to a small set of operations, and they would like your help to find the fewest operations needed to change from one design to another.

The designs are specified as a 5 by 5 matrix. The operations that are available change a square region of the matrix. This square region can be any contiguous 1x1, 2x2, 3x3, 4x4, or 5x5 portion of the matrix. The operations on these square regions are:

- Flip a square region of the matrix horizontally.
- Flip a square region of the matrix vertically.
- Negate a square region, i.e. invert all of the colors in this region.

Please see the explanation of the sample input below, for figures with these operations.

Input Format

The first line of input contains t, $1 \le t \le 5$, which gives the number of test cases.

Each test case is preceded by a blank line.

The test case consists of 5 lines. The first 5 characters in the i^{th} line of the test case represent the i^{th} row of the source matrix. The sixth character in the row is a space. The last 5 characters in the row represent the i^{th} row of the target matrix.
The `X` character is used to represent a black square and the `.` character is used to represent a white square.

Output Format

For each test case, you should output, on a line by itself, the minimum number of operations needed to transform the source matrix into the target matrix.

Sample Input

```
4

..... .....
..... .X...
```

```
.X.X. XXX..
..X.. .X...
.X.X. .X...

..... .....
XXX.. XXXX.
X..X. XXXX.
X.... X....
X.... XXXX.

..... .....
X...X XXXX.
XX.XX XXXX.
X.X.X X....
X...X XXXX.

..X.. ..X..
..X.. .X.X.
..X.. .X.X.
..X.. .X.X.
..X.. ..X..
```

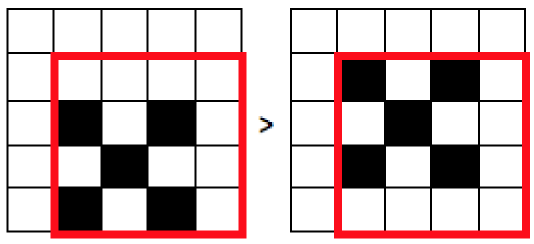
Sample Output

```
4
3
3
1
```

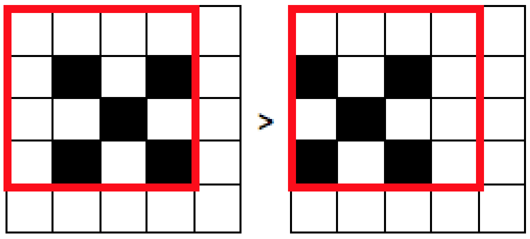
Explanation

In the first test case, the source matrix can be transformed to the target matrix with four operations. For example,

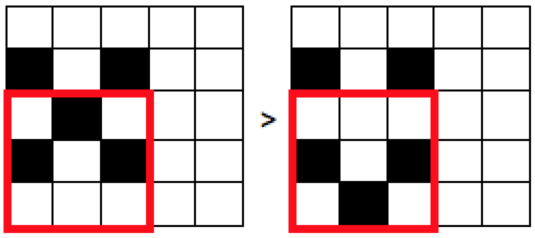
1) Vertical Flip:



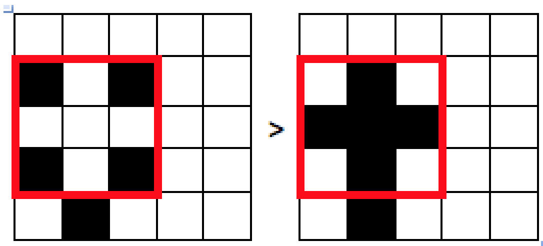
2) Horizontal Flip:



3) Vertical Flip:

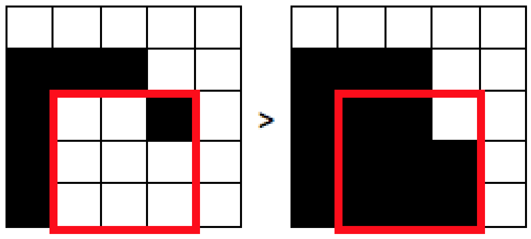


4) Negate:

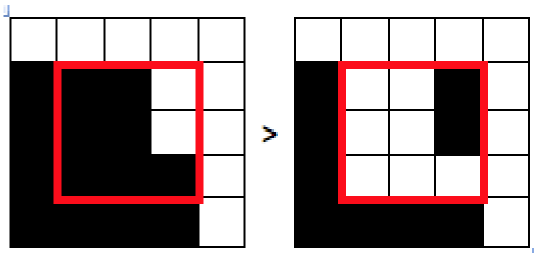


In the second test case, the source matrix can be transformed to the target matrix with three operations. For example,

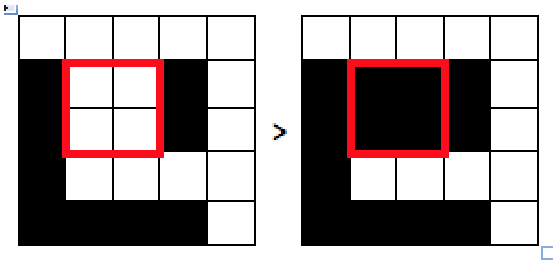
1) Negate:



2) Negate:

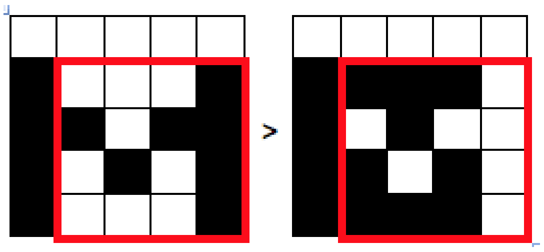


3) Negate:

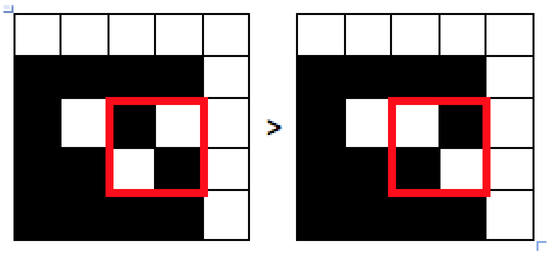


In the third test case, the source matrix can be transformed to the target matrix with three operations. For example,

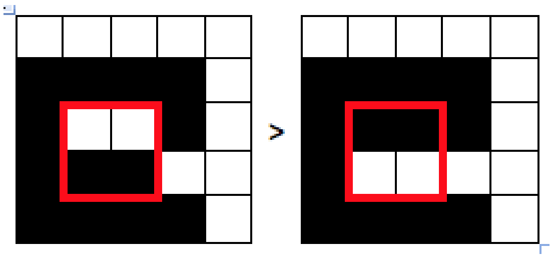
1) Negate:



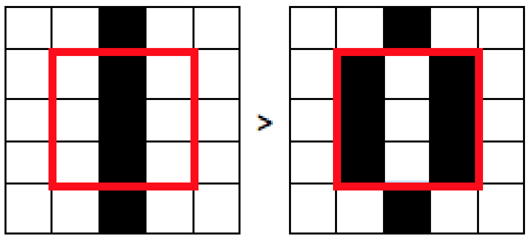
2) Horizontal Flip (or a vertical flip or negate):



3) Negate (or a vertical flip):



In the fourth test case, the source matrix can be transformed to the target matrix with a single negate operation:



P = NP?

When most computer scientists ask, "Does P = NP?", they are asking whether there are decision problems that can be verified in polynomial time, but not solved in polynomial time.

However, when Dr. I.M. Columsie asks this question, he is wondering about the relationship between pizza consumption and success in programming contests. His current research analyzes the results of a previous Xtreme contest. He gathered the final rankings for a selected subset of teams, and divided them into two, non-empty groups - a pizza (p) group, where team members ate copious amounts of pizza during the contest, and a no pizza (np) group, where the team members did not eat pizza. His hypothesis is that the pizza improves performance in these contests.
He wanted to compare the rankings of the two groups to see if the data supports his hypothesis. He started by writing down the sorted rankings within the p group and the np group. He noted that there were no teams that tied in the data he gathered, so all of the rankings were unique (both within each group and across the two groups).

He was about to compute the mean of the rankings for the two groups, when he spilled coffee all over his computer and the written sorted rankings. Unfortunately he lost the raw data on the computer, and a number of the written rankings were no longer readable.

He really hopes that you will be willing to help him by calculating a range of possible values for the sums of the rankings for the two groups. From these sums, he will be able to calculate a range of possible values for the means.

Input Format

The input starts with three integers t, p, and n, where t gives the number of teams in the contest, p gives the size of the pizza group, and n gives the size of the *no pizza* group.

Then there is a blank line.

The next p lines give the rankings, sorted in ascending order, for the *pizza* group. If a ranking is unknown, it will be listed as `?`.

After this first group of rankings, there is a blank line.

Following this are n lines that similarly give the rankings for the *no pizza* group.

Constraints

1 ≤ n + p ≤ t <= 2 * 10^5
Each ranking must be between 1 and t, inclusive.

Output Format

The output consists of two lines in the following format:

```
[p_sum_low] [p_sum_high]
[np_sum_low] [np_sum_high]
```

where

- `[p_sum_low]` and `[p_sum_high]` are the lowest and highest possible values for the sums of the rankings of the p group
- `[np_sum_low]` and `[np_sum_high]` are the lowest and highest possible values for the sums of the rankings of the np group

Note that it is guaranteed that there is at least one ranking that fits the data.

Sample Input

```
100 5 6

1
?
?
7
96

?
?
4
?
98
?
```

Sample Output

```
115 115
214 304
```

Explanation

Because the rankings must be unique, we can deduce constraints on the missing values. For example, since the `1` ranking is in the p group and the first two rankings in the np group must be less than `4`, the first two missing values in the np group must be `2` and `3`. We can also conclude that the first two missing values in the p group

must be 5 and 6. For the remaining missing values, we can establish multiple possibilities.
We know every value from the p group: [1, 5, 6, 7, 96], and, thus, we can establish an exact value for the sums of the possible rankings of this group.
The np group consists of the following list of values: [2, 3, 4, a, 98, b] where a is a value chosen from the set {8, 9, ..., 95, 97}, and b is a value chosen from the set {99, 100}. The smallest sum of the rankings would then be 214, if the rankings were [2, 3, 4, 8, 98, 99], and the largest sum of the rankings would be 304, if the rankings were [2, 3, 4, 97, 98, 100].

Goldbach's Second Conjecture

An integer $p > 1$ is called a prime if its only divisors are 1 and p itself. A famous conjecture about primes is Goldbach's conjecture, which states that

Every *even* integer greater than 2 can be expressed as the sum of *two* primes.

The conjecture dates back to the year 1742, but still no one has been able to come up with a proof or find a counterexample to it. We considered asking you prove it here, but realized it would be too easy. Instead we present here a more difficult conjecture, known as Goldbach's second conjecture:

Every *odd* integer greater than 5 can be expressed as the sum of *three* primes.

In this problem we will provide you with an odd integer N greater than 5, and ask you to either find three primes p_1, p_2, p_3 such that $p_1 + p_2 + p_3 = N$, or inform us that N is a counterexample to Goldbach's second conjecture.

Input Format

The input contains a single odd integer $5 < N \leq 10^{18}$.

Output Format

Output three primes, separated by a single space on a single line, whose sum is N. If there are multiple possible answers, output any one of them. If there are no possible answers, output a single line containing the text "*counterexample*" (without quotes).

Sample Input

```
65
```

Sample Output

```
23 31 11
```

Explanation

In the sample input N is 65. Consider the three integers 11, 23, 31. They are all prime, and their sum is 65. Hence they form a valid answer. That is, a line containing "11 23 31", "23 31 11", or any permutation of the three integers will be accepted. Other possible answers include "11 37 17" and "11 11 43".