Thirty-fifth Annual
University of Central Florida

High School Programming Tournament

Problems

Problem Name	Filename
Minimum Excluded String	mexstr
Blob Tag	blob
Triangular Trees	triangles
Build-a-House	house
Worse Code	worse
Knapsack?	knapsack
Molecular Mole	mole
Sharon's Sausages	sausage
Alphabetic Road Trip	roadtrip
Night Watch	night
Bouncing DVD	dvd
Improovion Expectations	expectations
No Mor	nomor
Interstellon Melon	melon
Glenn and Glenn's Pizza	pizza
Octopus Garden	octopus
Stealing Spider-Man	spiderman

Call your program file: *filename*.c, *filename*.cpp, *filename*.java, or *filename*.py

For example, if you are solving Octopus Garden:
Call your program file: octopus.c, octopus.cpp, octopus.java, or octopus.py
Call your Java class: octopus

Minimum Excluded String

Filename: `mexstr`

We define the minimum excluded (mex) string to be the smallest string that is not contained in a given string. A string is considered smaller than another string if its length is shorter. If their lengths are equal, we then compare them character by character from left to right. The first location where we find unequal characters, we say the string containing the character which comes earlier in the alphabet is smaller. For example, "orange" is smaller than "blueberry" (based on lengths), and "autocratic" is smaller than "automobile" (based on "c" is earlier in the alphabet than "m").

The Problem:

Given a string, find the mex string of that given string.

The Input:

The first line of the input begins with a single, positive integer, *t*, representing the number of strings to process. Then, *t* lines follow, each containing a single string, *s*, of between 1 and 500 letters in length, inclusive. It is guaranteed that the string contains only lowercase letters.

The Output:

For each string, output a single line of the mex string of the input string.

Sample Input:

```
2
banana
abcdefghijklmnopqrstuvwxyz
```

Sample Output:

```
c
aa
```

Blob Tag

Filename: `blob`

Glenn is a P. E. (Physical Education) teacher at a local elementary school and has come up with a new hit game called "Blob Tag." The rules are fairly simple: every child starts with empty hands and will run around the playground; if any two bump into each other, they must lock hands immediately (one hand each) and cannot separate for the rest of the game. They can only lock hands if both have a free hand (so if a child bumps into more than two other children, the child will only stick to the first two).

Glenn is worried about cheating, so before the game starts, he coats each child's hands in Hyper Sticky Play-Tar (HSPT). However, after the game, he realizes that it might be difficult for students to learn when they go back to class if they are stuck to one another and covered in tar, and figures he should fix this issue. To do this, he needs to know the number of students stuck together in the largest blob.

The Problem:

Given the number of students in Glenn's class, and the full list of every time two children bump into each other, determine the number of students in the largest group at the end of Blob Tag.

The Input:

The first line of the input will contain a single, positive integer, ***c***, representing the number of P.E. classes Glenn teaches (and in which he repeats the same child-tarring mistake). Each class then follows in the input. The first line of each class will contain two integers, n and m ($1 \leq n \leq 1{,}000$; $0 \leq m \leq 1{,}000$), where n is the number of students and m is the number of collisions during the game, respectively. This is followed by m lines, each containing two integers, a and b ($1 \leq a \leq n$; $1 \leq b \leq n$; a != b), indicating that the a^{th} student and the b^{th} student bump into each other. These collisions are given in chronological order, and it is guaranteed that no pair of students will collide with each other more than once within one P.E. class.

The Output:

For each of Glenn's P.E. classes, output a single line containing a single integer, g, representing the number of children in the largest blob at the end of the class.

(Sample Input and Sample Output follow on next page)

Sample Input:

```
2
5 3
1 2
2 3
4 5
10 5
1 2
4 3
1 3
2 4
4 5
```

Sample Output:

```
3
4
```

Triangular Trees

Filename: `triangles`

Peter is a lumberjack that is a huge fan of triangles, and has recently received a number of large shipments, n, of trees. In each shipment there are m sticks, each stick being of a positive unique integer length. No sticks within a single shipment are of the same length. Peter wants to determine the maximum number of unique acute, right and obtuse triangles that he can make with his sticks. To compute this number of each type, he will consider all the sticks each time.

The Problem:

Given the number of sticks and their lengths in a shipment of trees, determine how many acute, right and obtuse triangles can be made by Peter.

The Input:

The input will start with a single, positive integer, n, representing the number of shipments of trees Peter is expecting to arrive. Each of the next n lines each represents a shipment and will start with a positive integer, m ($1 \leq m \leq 2{,}000$), representing the number of sticks in that shipment. This is followed by m integers on the same line, each representing the length of a stick, s ($1 \leq s \leq 2{,}000$), in that shipment.

The Output:

For each shipment, output the number of unique acute, right, and obtuse triangles (in that order, each separated by a single space) that Peter can make with his sticks from that shipment.

Sample Input:

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

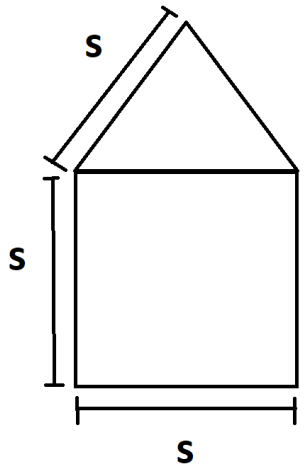
Sample Output:

```
0 1 1
1 1 2
0 1 2
```

Build-a-House

Filename: `house`

Bob the Builder wants to build a house. Of course, since he is a character on TV, he lives in a 2-dimensional world. For this reason he will build a house as shown in the figure below:



As you can see in the figure, Bob spent so much time preparing the brilliant design that he has forgotten the true folly in his ways. He forgot to buy materials! He needs to buy a long plank from Home Depot, which he can then cut into pieces for the building. This plank needs to be at least as long as the *outer* perimeter of the house. Can you help him determine what this length is?

The Problem:

Given the side length as described, compute the perimeter of the house.

The Input:

The input will start with a single, positive integer, t, representing the number of houses to build. Then, t lines will follow (one for each house), each with a single integer, s ($1 \leq s \leq 100$), representing the side length shown in the figure.

The Output:

For each house, output a single integer, representing the perimeter of the house.

(Sample Input and Sample Output follow on next page)

Sample Input:

```
5
16
5
10
100
44
```

Sample Output:

```
80
25
50
500
220
```

Worse Code

Filename: `worse`

Do you know Morse code? Well, we invented something better – Worse code! Worse code is an encoding scheme in which uppercase letters are represented by dots (*), and the number of dots is equal to the letter's position in the alphabet.

Please refer to the Worse code manual below:

A	*	**N**	**********
B	**	**O**	**********
C	***	**P**	**********
D	****	**Q**	**********
E	*****	**R**	**********
F	******	**S**	**********
G	*******	**T**	**********
H	********	**U**	**********
I	*********	**V**	**********
J	**********	**W**	**********
K	**********	**X**	**********
L	**********	**Y**	**********
M	**********	**Z**	**********

In order to distinguish between different characters, we add a space in between them.

The Problem:

Our mutual friend has sent us a word written in Worse code, but counting the dots can be time consuming, and you risk losing track! Therefore, we decided to have you write a program to decode the word in Worse code into English.

(Sample Input and Sample Output follow on next page)

The Input:

The first line of the input will begin with a single, positive integer, *t*, representing the number of words to process. For each word, a single string of asterisks and spaces follows representing the encoded word in Worse code. It is guaranteed that the input corresponds to an English word, there will be no leading or trailing spaces on any line, and the encoded word (of asterisks and spaces) will be of length of the string is no greater than 10,000.

The Output:

For each word, output a single line containing the decoded word in English.

Sample Input:

```
3
**** * ****
**** ***** ***** ******************
****** ************ *************************
```

Sample Output:

```
DAD
DEER
FLY
```

Knapsack?

Filename: `knapsack`

You are playing the newest most popular Massively Multiplayer Online Role-Playing Game! You just finished a raid with the guildies and need to pick up loot to bring back and sell on the market place. Fortunately, when you pick up an item you are able to tell its weight, and you use an add-on that tells you the current value of every item you pick up. Unfortunately, your inventory can only hold so much weight.

Find out what is the maximum total value you can gain by picking up loot, without going over the inventory capacity! For each raid you have done with your guildies, you are given the loot that dropped for you, including each drop's weight and value, as well as your inventory capacity.

The Problem:

For each raid determine what is the maximum total value you can gain by picking up loot, without going over the inventory capacity! You can assume that your inventory is completely empty at the end of each raid.

The Input:

The first line of the input will contain a single, positive integer, t, representing the number of raids you finished. The next t lines will contain the drops for each raid. Each raid will begin with two integers, n ($1 \leq n \leq 1{,}000$) and m ($995{,}000 \leq m \leq 1{,}000{,}000$), representing the number of loot items that dropped for you, and your inventory capacity, respectively. The next n lines will each contain 2 integers, v ($1 \leq v \leq 1{,}000{,}000$) and w ($1 \leq w \leq 1{,}050$), representing the value and weight, respectively, of each item.

The Output:

For each raid, output an integer on a new line representing the maximum value of loot you can collect for that raid.

Sample Input:

```
2
2 1000000
1000000 1050
1000000 999
4 999000
14 1000
7 100
3 10
1 1
```

Sample Output:

```
2000000
25
```

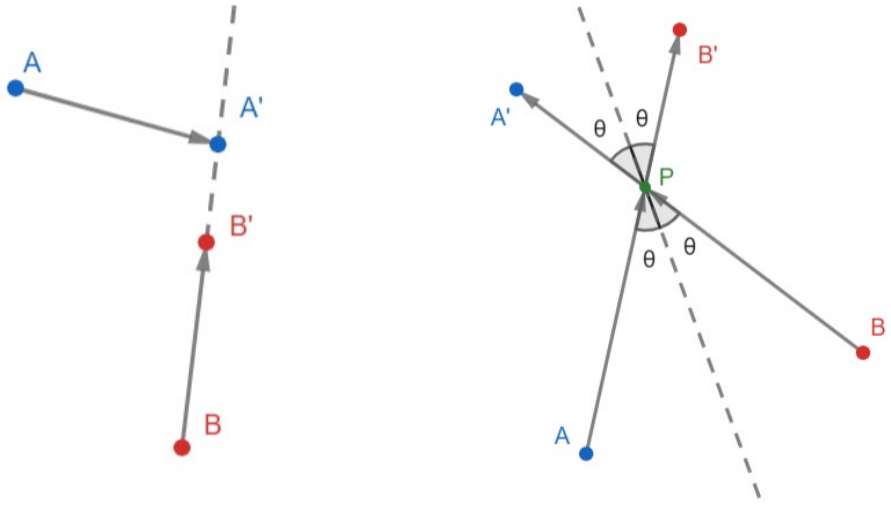
Molecular Mole

Filename: `mole`

Winston wants to create a giant mole. In order to do this, he studies modifications that take place in the molecular structure of moles. In his studies, he discovers that the structure consists of a number of molecules that all start in unique positions. Of course, Winston only studies moles with two dimensions, so the starting positions of each molecule inside the mole can be described by a pair of (x, y) integer coordinates on a flat plane. At time 0, each molecule's position is considered at their starting position. Additionally, each molecule moves in a given direction, given by an integer directional vector (dx, dy). The molecule will follow the direction of this vector, and move at precisely 1 unit of distance per 1 unit of time. This means that while each direction vector may have different magnitudes, the molecules all have the same speed.

Winston observes that over time, the molecules will collide with each other. A collision is defined to occur when two molecules occupy the same position at the same point in time.

To define a formal collision between two molecules, first consider the molecules the moment before the collision. When the molecules collide, they will reflect off the angle bisector between the two direction vectors. The angle bisector between two vectors is the line that has the same angle between itself and the first vector as the angle between itself and the second vector. This reflection means that the angle between the angle bisector and the direction of the molecule *before* the collision is the same as the angle between the angle bisector and the direction vector of the molecule *after* the collision. Some examples are pictured below:



In the image on the left, molecule A and molecule B arrive at the same point at the same time. This point is labeled as P. The angle bisector between A's old direction vector (AP) and B's old direction vector (BP) is shown as the dashed line. A's new direction vector is shown as PA' and B's new direction vector is shown as PB'. All angles labeled as θ are equal.

In the image on the right, molecule A and molecule B do not collide. Molecule A crosses molecule B's path at a time before molecule B crosses molecule A's path. A' denotes the position of molecule A after some time t, and B' denotes the position of molecule B after the same amount of time (i.e., the vectors AA' and BB' are the same length).

Winston is a simple man, and will only consider moles of molecular structure such that no collisions between 3 or more molecules at the same place and time will occur. He realizes that if he is able to count the number of collisions for which each molecule takes part within the first 1 million seconds for a multitude of moles, he can genetically create the ultimate mole. Please help Winston!

The Problem:

Given the molecules' starting positions and their direction vectors, determine, for each molecule, how many collisions it will have with another molecule within the first 1,000,000 seconds, inclusive. Note that due to the setup of the molecules, there will be a finite number of collisions for each molecule. In addition, two molecules are defined to have collided if they are within 0.000001 of each other at the same time; furthermore, no one molecule will have two collisions within 0.000001 seconds of each other.

The Input:

The input will start with a single, positive integer, t, representing the number of moles to consider. Then, t moles will be described. For each mole, the first line will contain a single integer, n ($1 \leq n \leq 1{,}000$), describing the number of molecules in the mole. Then, n lines will follow, each line describing a molecule. The i^{th} line contains four integers, x, y, dx and dy ($-10{,}000 \leq x \leq 10{,}000$; $-10{,}000 \leq y \leq 10{,}000$; $-10{,}000 \leq dx \leq 10{,}000$; $-10{,}000 \leq dy \leq 10{,}000$; $|dx| + |dy| > 0$), representing the starting coordinate for the i^{th} molecule (x, y), and the direction vector for the i^{th} molecule (dx, dy), respectively.

The Output:

For each of t moles, output n lines. The i^{th} of these lines should contain a single integer denoting the number of collisions in which the i^{th} molecule will participate.

Sample Input:

```
2
2
2 1 -4 4
-2 1 1 1
4
5 7 0 -2
5 6 0 3
5 0 0 1
4 7 12 0
```

Sample Output:

```
1
1
2
2
1
1
```

Sharon's Sausages

Filename: `sausage`

Sharon is a food-loving individual, and recently he has discovered that he loves the taste of different types of sausages!

When he was invited to the HSPT potluck, he decided that he would contribute 4 sausages for the cause in order to spread his love for sausages. Sharon also loves symmetry, so he chose the 4 sausages in such a way that the first and fourth sausages have the same length, and the second and third sausages have the same length.

While on his way to the potluck, Sharon came across a sausage field, which consisted of many sausages lined up in a row (how glorious!). To his horror, though, he dropped his 4 sausages into the field and now they're lost! Luckily he explained the situation to the owner of the sausage field and he has allowed Sharon to choose 4 sausages from the field. Sharon doesn't remember the lengths of the original sausages that he chose, but he remembers that he dropped the first sausage to the left of the second, the second to the left of the third, and the third to the left of the fourth. Sharon now wants to know how many candidates there are for choosing 4 sausages from the field such that his conditions are met. Can you help him?

The Problem:

Given n sausage lengths, output the number of ways to choose 4 of them with positions a, b, c, d such that $1 \leq a < b < c < d \leq n$, while also making sure that $length_a = length_d$ and $length_b = length_c$.

The Input:

The first line of the input contains a single, positive integer, p, representing the number of potlucks in which Sharon is invited. Each potluck is then defined on two lines. The first line of each potluck contains a single integer, n ($4 \leq n \leq 10^5$), representing the number of sausages in the sausage field. The next line contains n space-separated integers, $length_i$ ($1 \leq length_i \leq 100$), representing the length of the i^{th} sausage, respectively.

The Output:

Output the number of ways to select 4 sausages that meet Sharon's requirement.

(Sample Input and Sample Output follow on next page)

Sample Input:

```
2
8
2 1 3 4 2 1 2 3
8
1 3 3 7 1 3 3 7
```

Sample Output:

```
2
3
```

Alphabetic Road Trip

Filename: `roadtrip`

Josh is in his twenties and wants to live wildly. While he is not much of a partier, he does love to explore. One of his favorite ways to explore is to take road trips, but he is tired of simple trips. Instead, he wants to take an alphabetic road trip!

An alphabetic road trip is a journey on which the travelers visit exactly one city beginning with each letter in alphabetical order. For example, some stops on an alphabetic road trip may be, in order: Altamonte, Baylake, Casselberry, Daytona… Yankeetown, Zellwood. It would not be considered an alphabetic road trip, however, if a city beginning with a latter letter were visited before a former letter (like Baylake before Altamonte) or if any letter were repeated or skipped all together. Alphabetic road trips must always begin at a city that starts with an 'A'.

Like anyone in their twenties, Josh is concerned about the cost of his explorations. Specifically, the cost of gas. While he has a special map that shows him the cost, in gas, of travelling between some pairs of cities, he doesn't know how to go about minimizing the cost of his whole trip. Also, the alphabet is very long; there's no way Josh can visit every letter! Instead, he's only considering trips up to the letter 'J' (for Josh!). Note that it is okay for Josh to pass through a city without considering it as a stop on his trip and stopping in a city does not take any additional gas.

The Problem:

Given a list of cities and the cost to travel between them, determine the minimum cost of Josh's alphabetic road trip.

The Input:

The input begins with an integer, t, indicating the number of alphabetic road trips Josh is considering travelling. Each trip begins with two integers, n and m ($10 \leq n \leq 10^5$, $9 \leq m \leq 10^6$), indicating the number of cities and the roads between them, respectively. The following n lines each contain a string, a_i ($1 \leq |a_i| \leq 10$) that begins with a capital letter (A through J, inclusive) followed by lowercase letters, indicating the name of the i-th city. Next there are m lines of the form:

```
A B d
```

describing a road between a pair of cities, A and B, (where A and B are distinct names consistent with the previous input) and a single integer, d ($1 \leq d \leq 1{,}000$), the cost of gas to travel between A and B. Gas prices are the same from B to A, and roads are bidirectional. Note that there is guaranteed to be at least one possible road trip.

The Output:

For each trip, output a single line of a single number representing the minimum cost for Josh to complete his alphabetic road trip.

Sample Input:

```
2
11 10
Alabama
Alaska
Buffalo
Columbia
Delaware
Elfville
Florida
Georgia
Hawaii
Idaho
Jupiter
Alabama Jupiter 1
Jupiter Alaska 2
Idaho Jupiter 1
Georgia Idaho 1
Hawaii Georgia 1
Florida Hawaii 1
Elfville Florida 1
Elfville Delaware 1
Delaware Columbia 1
Buffalo Columbia 1
10 9
Avalon
Harrison
Dover
Bermuda
Gillian
Camelot
Jackson
Elliot
Florida
Iguana
Dover Avalon 3
Avalon Gillian 10
Florida Avalon 71
Avalon Bermuda 5
Jackson Avalon 5
Avalon Harrison 52
Elliot Avalon 6
Iguana Avalon 4
Camelot Avalon 5
```

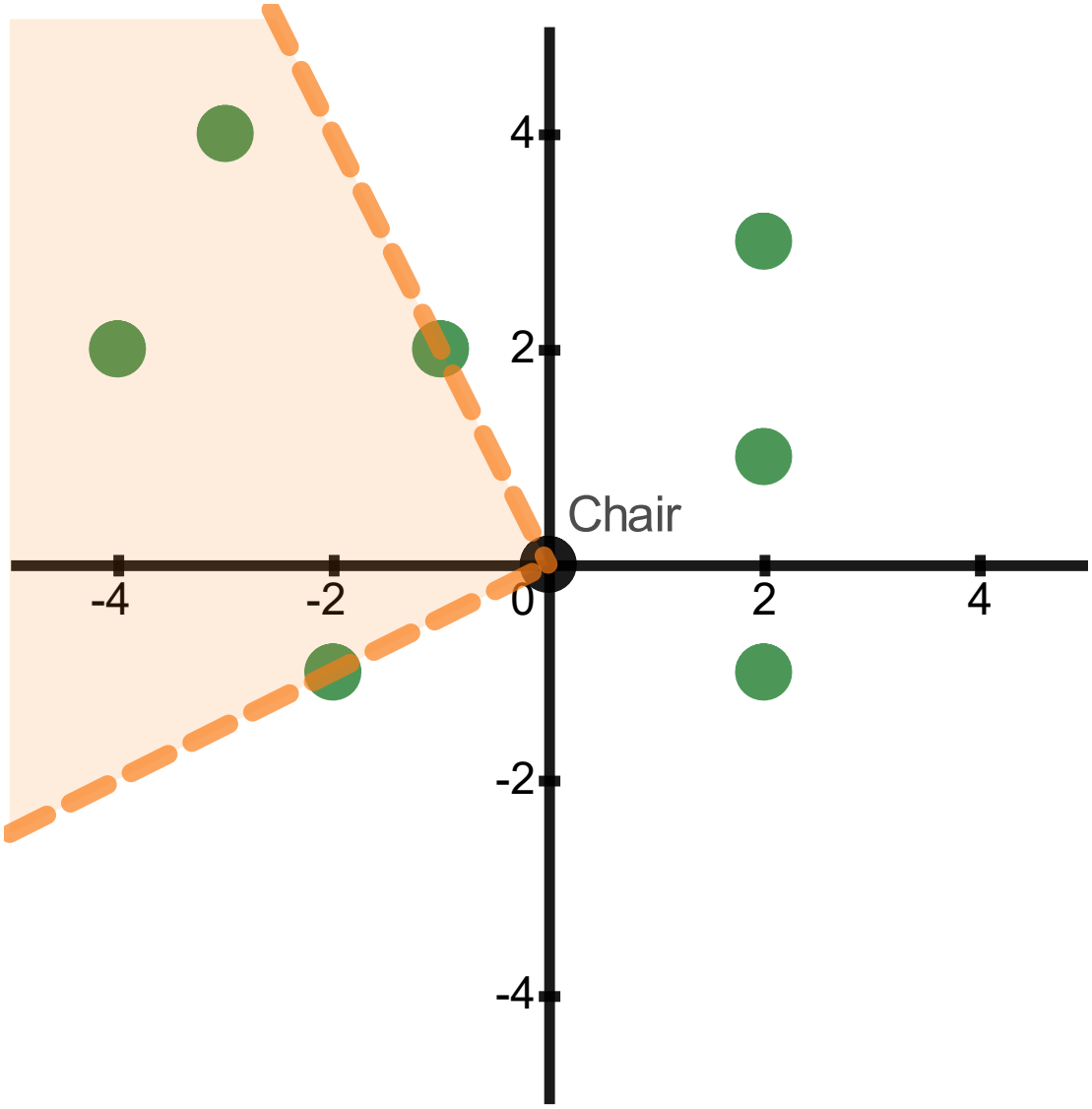
Sample Output:

```
19
317
```

Night Watch

Filename: `night`

Anorak has recently taken up the night shift at Gregarious Simulation Systems (GSS), where his primary job is to monitor several server racks, and ensure that they stay in working order all night. The server racks are all mounted on wheels, and are placed in random locations inside of Anorak's monitoring room, complete with his spinny chair, from which Anorak has a 90 degree field of view. Below is an example room, where the black point is Anorak's chair, the green points are server racks, and the orange cone is Anorak's field of view.



Since GSS makes so much money every year, they can afford to lose some of their servers every night. Anorak loves spinny chairs, but somewhat oddly hates spinning in them. In order to completely avoid spinning in his chair during his shift, Anorak decides to move some of the server racks into a more optimal position, so that he can monitor a satisfactory number of them all at once, without having to spin at all!

The Problem:

Given a list of n initial server rack locations, Anorak's chair location, and the number of servers k that his boss allows him to lose every night, determine the minimum number of server racks he must move in order to guarantee that he can monitor at least n-k server racks without having to move his chair any further from the position that he starts it at. Anorak can start his chair at any angle.

The Input:

The input will begin with a single, positive integer, t, representing the number of server rooms for you to analyze. Each server room will begin with integer, k ($0 \leq k \leq 1{,}000$) and an ordered integer pair (x, y) ($-10^7 \leq x \leq 10^7$; $-10^7 \leq y \leq 10^7$), representing the number of server racks you are allowed to lose, and the position of Anorak's chair, respectively. This will be followed with an integer, n ($0 \leq n \leq 1{,}000$; $n \geq k$), representing the total number of server racks. This line will be followed by n lines of ordered pairs of integers, (x, y) ($-10^7 \leq x \leq 10^7$; $-10^7 \leq y \leq 10^7$), which represents server i's position. Anorak will have a direct line of sight to each server (no server will be in front of another from Anorak's perspective).

The Output:

For each server room, output a line with a single integer, c, representing the minimum number of server racks Anorak must move in order to optimally monitor the room

Sample Input:

```
2
1 0 0
7
2 1
2 3
-1 2
-3 4
-4 2
-2 -1
2 -1
0 0 0
2
1 0
-1 0
```

Sample Output:

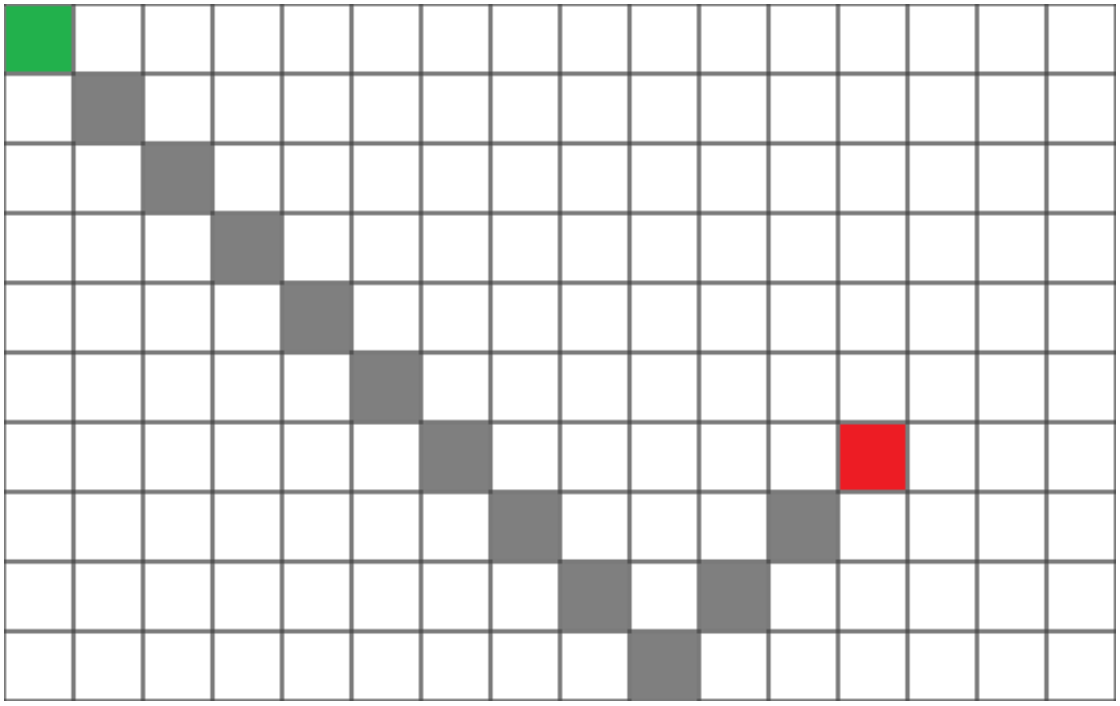
```
2
1
```

Bouncing DVD

Filename: `dvd`

Ever since the flat-panel LCD screens took over the market, most of us find ourselves no longer in need of screen savers, as their original purpose was to protect CRT screens (the large bodied TVs/monitors we no longer see much of these days) from being damaged by static frames over prolonged periods. However, this does not change the fact that the bouncing DVD logo has remained an oddly fascinating topic for the last decade or two. It involves a screen saver where the famous DVD logo travels across the screen in 45-degree angles and bounces off the edges. And whenever the logo precisely bounces off a corner, the people go wild! The logo's speeds in the x and y directions are always equal, allowing us to measure the distance of the logo since the beginning of its journey using a certain procedure.

We can consider every simultaneous incrementation of the x and y coordinates as an increase in the distance by one unit. For example, if the logo starts at the position (0, 0) and moves to (1, 1), the distance so far will be 1 unit. Below is a visual representation of an example scenario. The green space is the starting point of the logo. The gray spaces mark the path of the logo. The distance to the red space is 12 units. If the path was continued, the logo would eventually hit the bottom-right corner of the screen with a total distance traveled of 45 units.



The Problem:

Assuming the top-left corner of the logo starts at the top-left corner of the screen (0, 0), determine the distance to the next corner the logo will touch given the width and height of the screen and of the logo.

The Input:

The first line of the input will contain a single, positive integer, *n*, indicating the number of different DVD logo screens to analyze. The next *n* lines will each contain the input data for each screen. Each of these lines will contain four space separated integers, *w1*, *h1*, *w2*, and *h2* ($0 < w1 < w2 < 1{,}000$; $0 < h1 < h2 < 1{,}000$), where the first two integers represent the width and height of the DVD logo and the second two integers represent the width and height of the screen, respectively.

The Output:

For each DVD logo screen, output a single line with an integer, *d*, which is the distance it takes until the DVD logo hits the next corner.

Sample Input:

```
2
1 1 16 10
100 120 256 144
```

Sample Output:

```
45
312
```

Improovion Expectations

Filename: `expectations`

As a citizen of the legendary *Improovion* species, you receive a personal rating each year. Being a society of ever improving beings, the *Improovions* have declared that citizens that do not improve upon their personal score from the previous year will be banished to the cold unforgiving vacuum of space. No *Improovion* wants that so help a good citizen by computing how much additional score is needed!

The Problem:

Given a list of citizens' projected personal scores over the next n years, how many more points must each citizen gain in order to escape banishment?

The Input:

The input will begin with a single, positive integer, t, representing the number of citizens to evaluate. Each citizen's report will begin with a single, positive integer, n ($n \leq 10^6$), indicating the number of years, followed by n space-separated integers, s_i ($1 \leq s_i \leq 10^5$), representing projected scores for each given year, respectively.

The Output:

For each citizen, output a line with a single integer, p, representing the number of points they must gain to avoid banishment.

Sample Input:

```
3
5 1 2 3 4 5
4 4 2 8 8
10 10 9 8 7 6 5 4 3 2 1
```

Sample Output:

```
0
4
90
```

No Mor

Filename: `nomor`

Derek, the world-famous megalomaniac, has just unveiled his latest stunt: he has copyrighted the letter *e*! How trrifying! You now hav hours of work to do to rmov that lttr from all of your libraris of unrlasd twittr posts. You would lik to writ a program to do it for you.

The Problem:

Writ a program to rmov th copyrightd lttr from a block of txt.

The Input:

The first line of input will contain a single, positive integer, *l*, representing the number of lines of text on which you will have to run your program. Then, *l* lines will follow, containing at most 500 characters each, and only containing lowercase letters (a-z), spaces, and standard US keyboard punctuation. Each word within a line is guaranteed to have at least one non-e letter.

The Output:

Output each line, with the copyrighted letter (e) removed.

Sample Input:

```
3
thinking of dyeing my hair, any suggestions for color?
guys, i just saw the cutest corgi earlier today!
i love beeeeeeeeeeeeeeees! they make delicious honey!
```

Sample Output:

```
thinking of dying my hair, any suggstions for color?
guys, i just saw th cutst corgi arlir today!
i lov bs! thy mak dlicious hony!
```

Interstellon Melon

Filename: `melon`

Melon Musk, owner of SpaceY, tasked his interplanetary search satellites to find nearby melons in space. The satellites have provided very *fruitful* results, having located many muskmelon (which are a real fruit!) in nearby star systems! Mr. Musk can't wait to seize this *once-in-a-melon* chance to lay his hands on some delicious fruity goodness, and he has left the *juicy* details to you.

You have received data from the satellites as to the location and type of the melon. Your task is to determine how long Melon Musk will have to wait to sink his teeth into some delicious fruit. Earth is located at interstellar coordinates (0, 0), naturally, and your task is to compute the distance your rocket, the Honeydew-9, will have to travel to the melon and back. The Helpful Star Patrol Taskforce has restricted all rockets to either traveling parallel to the x- or y-axis at any one time, however, lest you upset the galactic melon lords.

The Problem:

Given the location of a melon in space, determine how much distance your rocket will have to travel from Earth to the melon and back.

The Input:

The input will start with a single, positive integer, m, representing the number of melons for which to answer. Then, m lines will follow, each with two integers, x and y ($0 < x < 10^8$; $0 < y < 10^8$), the coordinates of that particular muskmelon.

The Output:

Output m lines, one for each melon. For each melon, output a single integer, representing the distance from Earth to the melon *and back*. Note that your rocket can only travel parallel to the x- *or* y-axis at any point in time.

Sample Input:

```
3
3 4
5 11
10 13
```

Sample Output:

```
14
32
46
```

Glenn and Glenn's Pizza

Filename: `pizza`

Glenn wants to buy a pizza from the famous Glenn's Pizza pizzaeria! However, since Glenn is quite greedy, he doesn't want to share it. He knows that if someone sees him eating the pizza, they will want to eat it too. Since the pizza is quite big, it would be rude of him to not share it.

Each person has some toppings that they hate and wouldn't eat a pizza that has any of those toppings. Glenn can put toppings on the pizza to prevent people from wanting to eat it. However, Glenn only has enough money for k toppings, and he is willing to spend it all if that's what it takes to share with the minimal number of people possible.

The Problem:

Given a list of people and what toppings they hate, find the minimum number of people Glenn has to share the pizza with by adding up to k toppings.

The Input:

The input begins with a single, positive integer, d, representing the number of days that Glenn will buy pizza. Each day begins with three integers, n ($1 \leq n \leq 10$), m ($1 \leq m \leq 50{,}000$) and k ($1 \leq k \leq m$), representing the number of people, the number of toppings available for purchase, and the maximum number of toppings Glenn can buy, respectively. Each of the next n lines describes a person. The line starts with an integer, t ($1 \leq t \leq m$), which is the number of toppings this person hates. The next t integers are the index of a topping, which are numbered 1 to m.

The Output:

For each pizza, print a single integer: the minimal number of people Glenn must share his pizza with. You can assume that if Glenn does not include a topping that is disliked by someone, they will ask him to share the pizza with them.

Sample Input:

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

Sample Output:

```
1
0
```

Octopus Garden

Filename: `octopus`

Deep under the waves, near a cave, lies a certain garden tended by a friendly octopus. He has a collection of shells in a line, and has assigned a unique beauty value to each shell, and wants to sort them in order of increasing beauty.

In one move, he can wave his arms, creating a current in the water that rotates a continuous subsequence, or subarray, of his seashells. More formally, in one move, he can pick a subarray and shift every element one index to the left or to the right, except for the one at the end, which goes to the other side of the subarray. For example, if he has the array [1, 4, 5, 2, 3] and performs a right cyclic rotation on the subarray of indices 2-4, the array becomes [1, 2, 4, 5, 3].

Sorting his seashells could take quite some time, so he would like your help determining how much time it will take him. In exchange, you can visit his garden anytime to rest your head on the seabed!

The Problem:

Given the current state of the octopus' seashell collection, determine the minimum number of cyclic subarray shifts required to sort the shells in increasing order of beauty.

The Input:

The first line will be a single, positive integer, g, representing the number of gardens the octopus wants to tend. For each of g gardens, you will be given an integer, n ($0 < n < 10^5$), representing the number of shells in the garden. The following line contains n integers which describes the current beauty state of the shells in the garden. Each shell's beauty is denoted by $0 < b_i < 10^9$. No two shells will have the same beauty value.

The Output:

You will output g lines, one for each garden to be tended. For each garden, output a single integer w, the minimum number of times the octopus will have to wave his arms, performing cyclic subarray shifts, to sort the array in increasing order of seashell beauty.

(Sample Input and Sample Output follow on next page)

Sample Input:

```
3
5
5 3 4 2 1
4
10 42 12 21
1
1337
```

Sample Output:

```
3
1
0
```

Stealing Spider-Man

Filename: `spiderman`

Headline **S**tory: **P**erpetrator **T**om **H**olland (**S**pider-Man) **P**roven **T**o **H**ave **S**tolen **P**ure **T**opaz! **H**ow **S**cary! **P**lease **T**hink: **H**ow **S**hould **P**rogrammers **T**heorize **H**elpful **S**olutions, **P**ersevering **T**hrough **H**ardly **S**imple, **P**roblematic **T**imes?

You are the head of the **H**ighly **S**mart **P**rogramming **T**eam division in the NYPD and have been tasked with determining if the police have enough time to stop Spider-Man before he escapes. The police have available to them one **H**ighly **S**uspicious **P**olice **T**rap, which can block one road so Spider-Man cannot escape using that road.

The Problem:

Being a part of the sophisticated police department, you have available to you a complete representation of the city in the form of intersections and roads connecting those intersections. Luckily, all locations in the city are connected with some pathway so Spider-Man won't be able to hide! Your task is to determine if there is a road such that if you place your **H**ighly **S**uspicious **P**olice **T**rap on that road, thereby blocking the road, Spider-Man will not be able to escape from the museum (where he stole the topaz) to his getaway helicopter.

The Input:

The input will start with a single, positive integer, t, representing the number of heists for you to solve. Each heist will start with a line containing two integers, n and m ($2 \leq n \leq 1{,}000$; $1 \leq m \leq 1{,}000$), representing the number of intersections and the number of roads, respectively. For the purposes of this problem, intersection 1 will be the museum, and intersection n will be Spider-Man's getaway point. Then, m lines will follow, each containing two integers, u and v ($1 \leq u \leq n$; $1 \leq v \leq n$), denoting that there is a unique *two-way road* between intersections u and v (no roads appear twice in the input within a single heist).

The Output:

For each heist, output "`Halt, Spider-Man! Plans Thwarted!`" if it is possible to place your **H**ighly **S**uspicious **P**olice **T**rap such that Spider-Man cannot escape that heist, or output "`How Sad, Perpetrator Triumphed.`" otherwise.

(Sample Input and Sample Output follow on next page)

Sample Input:

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

Sample Output:

```
Halt, Spider-Man! Plans Thwarted!
How Sad, Perpetrator Triumphed.
```