Algorithms for Programming Contests
WS18 - Week 13

Chair for Foundations of Software Reliability and Theoretical Computer Science,
TU München

Tobias Meggendorfer, Philipp Meyer,
Christian Müller, Gregor Schwarz

Welcome to our practical course! This problem set is due by

Wednesday, 06.02.2019, 6:00 a.m.

Try to solve all the problems and submit them at

`https://judge.in.tum.de/conpra/`

This week's problems are:

All students are invited to join this contest. If you do not have an account yet register on the website given above or write a message to conpra@in.tum.de, we have additional accounts. You may work together in teams of two students. Please write a message to us if you want to have a team account.

Sample solutions, statistics and small prizes for the winners will be given on Wednesday, 06.02.2019, at 12:15 p.m. in room MI 00.08.038. Happy solving!

Six points are awarded for each problem, resulting in 60 points in total. However, only 40 of them will count towards the total number of points, and the additional 20 points will be counted as bonus points.

If the judge does not accept your solution but you are sure you solved it correctly, use the "request clarification" option. In your request, include:

- the name of the problem (by selecting it in the subject field)
- a verbose description of your approach to solve the problem
- the time you submitted the solution we should judge

We will check your submission and award you half the points if there is only a minor flaw in your code.

If you have any questions please ask by using the judge's clarification form.

Problem A
Rotating Rectangles

It's one of these beautiful late summer days: the sun is shining, people are outside on the grass drinking cold beer, and everybody seems to have a relaxing day. Jhonny is watching them trough the window of the library. Unfortunately, there is absolutely no way he can join them. His Algorithm exam takes place next week and whenever Jhonny thinks about it, he feels rather queasy.

In particular there is this one geometry problem he simply cannot wrap his head around. It reads as follows: "Given two rectangles R_1 and R_2, find an algorithm to check whether R_2 and can be placed inside of R_1 such that none of its corners lie outside of the boundary of R_1."

When Jhonny first read this problem, it sounded very basic and trivial him. In fact, he had already written down an algorithm and continued to the next problem, when his friend Paul pointed out to him that both rectangles may not only be translated arbitrarily but can also be rotated in an arbitrary way - and not only in multiples of 90 degrees. Ever since, Jhonny has been staring at the problem without making any progress. Maybe you can help him out?

Input

The first line of the input contains an integer t. t test cases follow, each of them separated by a blank line.

A test case consists of four integers a, b, c, d, where $a \times b$ is the dimension of rectangle R_1, and $c \times d$ is the dimension of rectangle R_2.

Output

For each test case, output one line containing "Case #i: s" with i being the number of the test case starting at 1, and s being `possible` if there is a way to place R_1 inside of R_2. Otherwise print out `impossible` for s.

Constraints

- $1 \leq t \leq 100$
- $0 \leq a, b, c, d \leq 10^5$
- If it is possible to place R_2 inside of R_1, it is guaranteed that there exists a placement such that the gap between the corners of R_2 and the boundary of R_1 is at least 0.001.

Sample Input 1	Sample Output 1
3 5 4 3 2 6 2 3 3 8 8 1 9	Case #1: possible Case #2: impossible Case #3: possible

This page is intentionally left blank.

Problem B
Spider-Rectangulaire

The software company *MakroHart* is about to release its newest operating system *Skylight 11*. Because users have complained about the limited quantity of built-in games in the past, the company decided to add a new game to their operating system: *Spider-Rectangulaire*, a fascinating, wickedly hard, one-person card game.

In this new game a player is given a set of cards, each displaying a unique spider. While all cards are in shape of perfect two-dimensional rectangles, they might differ in their dimensions. Initially a player is given n cards. In order to win, he must stack his cards in a way that they form no more than k piles. At first glance *Spider-Rectangulaire* may seem like a trivial game. However, one has to obey the strict rules regarding which two cards can be stacked onto one another:

- The dimensions of a pile's bottom card may be arbitrarily large.
- A card c can only be placed on top of card d if the four corners of card c do not lie outside of the boundaries of card d.
- Cards may only be rotated by 90 degrees, meaning that all four sides of any two cards in the same pile must be parallel to one another.
- Except for a pile's bottom and top card, any card in the pile must touch exactly two other cards.
- If a pile contains at least 2 cards, both the top and the bottom card must touch exactly one other card.

Of course, Lea really wants to crack the highscore once the new game is published, so she is wondering if one can compute in advance whether she can win the game or not.

Input

The first line of the input contains an integer t. t test cases follow, each of them separated by a blank line.

The input consists of one line containing two integers n and k, where n is the number of cards a player is given initially, and k is the maximal number of piles a player may use to stack all of his cards. n lines follow, each containing two integers x_i and y_i denoting that the i-th card on the player's hand is of dimension $x_i \times y_i$.

Output

For each test case, output one line containing "Case #i: s" with s being `possible` if there is a way for Lea to win the game and s being `impossible` if she will definitely lose.

Constraints

- $1 \leq t \leq 20$
- $1 \leq n \leq 500$
- $1 \leq k \leq 100$
- $1 \leq x_i, y_i \leq 100$

Sample Input 1	Sample Output 1
2 6 2 5 2 5 5 6 3 1 6 5 4 4 5 6 2 5 6 3 8 4 4 3 6 7 2 2 2	Case #1: possible Case #2: impossible

Problem C
Make Campaigning Great Again!

Dagobert J. Pump is desperate. The presidential election in Greatmania is going to take place just a few weeks from now and his only remaining opponent O'Bummer leads in all national polls. This has to change today - starting right here and right now!

The voting system in Greatmania is somewhat special. The country is subdivided into several districts. Each citizen in Greatmania is allowed to cast exactly one vote. However, people living together in the same district have to vote unanimously, meaning they all have to vote either for Dagobert J. Pump or his opponent O'Bummer. In the end, all votes from all districts are summed up and the candidate with the most votes wins the election. Because the total number of people in Greatmania is odd, there can never be a tie between the two candidates.

It is unknown how people in a district decide which candidate they want to endorse. However, if Dagobert invests enough money in campaigning events in a certain district, the people living there will surely vote for him. Dagobert's staff already prepared a table that lists for each district the amount of money Dagobert has to spend in order to convince the people living there to vote for him. All that's left to do is deciding which districts Dagobert is going to visit. His main objective is, of course, to be sure that he is going to win the election. Besides that, he would like to invest as little money as necessary. In case there are several different cheapest options, he would prefer to visit as few as districts as possible. After all he is not the youngest anymore.

Input

The first line of the input contains an integer t. t test cases follow, each of them separated by a blank line.

Each test case starts with a line containing an integer n, where n is the number of districts in Greatmania. n lines follow, each describing one of the districts. Line j contains two integers p_j and c_j, where p_j is the number of people living in district j, and c_j is the amount of money Dagobert has to invest to convince the people living in district j to vote for him.

Output

For each test case, output one line containing "Case #i: x y" where i is the number of the test case starting at 1, x is the minimum amount of money that Dagobert needs to invest to be sure that he is going to win the election and y is the minimum number of districts he needs to visit in this particular case.

Constraints

- $1 \leq t \leq 30$
- $1 \leq n \leq 100$
- $1 \leq p_j \leq 1000$
- $0 \leq c_j \leq 10^8$
- $\sum_{j=1}^{n} p_j$ is odd.

Sample Input 1

```
3
4
1 3
2 6
1 3
1 2

3
2 5
6 15
3 6

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

Sample Output 1

```
Case #1: 8 2
Case #2: 15 1
Case #3: 5 2
```

Sample Input 2

```
5
4
7 410
55 561
18 759
43 900

4
67 837
21 70
68 774
15 815

2
29 675
46 391

8
22 128
50 785
97 423
42 362
57 311
90 280
52 920
21 893

3
23 906
3 325
39 297
```

Sample Output 2

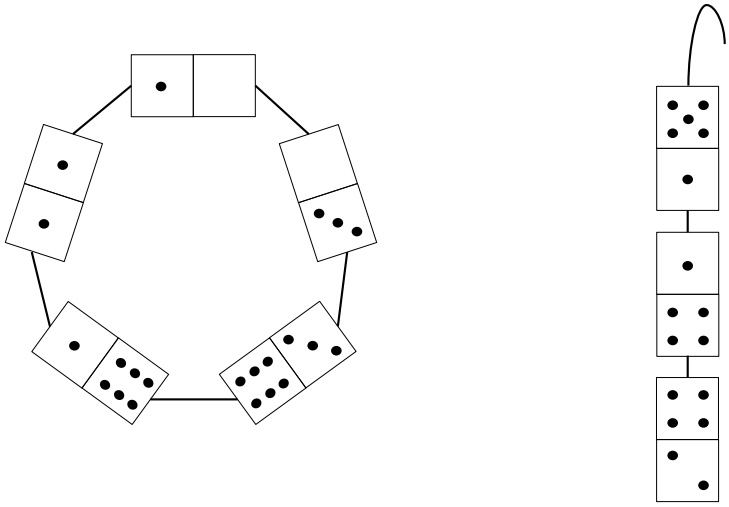
```
Case #1: 971 2
Case #2: 844 2
Case #3: 391 1
Case #4: 1014 3
Case #5: 297 1
```

Problem D
Domino Jewelry

It could have been all so much fun! And now Lea is lying all by herself in a hospital in a random Austrian village. What happened? A couple of weeks ago, Lea qualified to participate in this year's official ICPC training camp of the prestigious Technical University of Garchobirsk. Since Lea was afraid that the other participants were introverted, socially awkward nerds with vitamin D deficiency, she took her precious bottle of Cetraria Islandica hard liquor (Islandic moss liquor) with her to be able to cheer herself up if things became as bad as anticipated.

After having dominated the first two contests, she rewarded herself with a couple of shots before going to bed. Even though Lea is an experienced drinker, she woke up with a stomach ache the next morning. However, this did not stop her from joining the others on a hiking trip. But after having walked a couple of miles, her medical condition suddenly worsened. The next thing Lea remembered was hanging on the rope attached to an emergency helicopter that took her to the closest hospital.

A couple of hours later, Lea already feels a bit better - but now she is bored to death. Eventually, the medical stuff hands her a set of dominoes. After having played several rounds against herself, she suddenly gets the brilliant idea to make jewelry out of the dominoes! She either wants to make a necklace or a single dangle earring. And of course both the necklace and the earring should comply with the well-known domino rules: The sides of adjacent dominoes should always show the same number of points. However, in case of an earring Lea does not care if the upper side of the first domino piece matches lower side of the last domino piece. To visualize her demands, Lea quickly sketched an example of a valid necklace and earring:



Because there are only a few domino pieces, Lea wants to use all of them to either make a necklace or an earring. No domino piece should be left after she is done. Lea would like to have a program that lets her know in advance if it is even possible to craft a necklace or an earring with respect to her demands. Unfortunately, she still feels a bit dizzy from the helicopter ride. But maybe you can help her out?

Input

The first line of the input contains an integer t. t test cases follow, each of them separated by a blank line.

Each test case starts with two integers n k, where n is the total number of dominoes and k is the number of distinct dominoes up to rotations. k lines follow describing one kind of domino piece. Each line contains three integers a_i b_i c_i where a_i and b_i describe the two sides of the i-th kind of domino piece and c_i is the number of available domino pieces of the i-th kind.

Output

For each test case, output one line containing "Case #i: s_1 s_2" with i being the number of the test case starting at 1, and s_1 being "yes" if it is possible to craft a domino necklace out of all the available pieces such that no piece is unused, and "no" otherwise. Similarly, s_2 should be "yes" if it is possible to craft a domino dangle earring out of all the available pieces such that no piece is unused, and "no" otherwise.

Constraints

- $1 \leq t \leq 100$
- $2 \leq n \leq 10^8$
- $1 \leq k \leq n$
- $0 \leq a_i \leq b_i \leq 10^5$ for all $1 \leq i \leq k$
- $1 \leq c_i \leq n$ for all $1 \leq i \leq k$
- $\sum_{i=1}^{k} c_i = n$

Sample Explanation

The first and second sample test case refer to the necklace and earring shown in the problem statement.

Sample Input 1	Sample Output 1
3 5 5 0 1 1 1 1 1 1 6 1 3 6 1 0 3 1 3 3 1 5 1 1 4 1 2 4 1 10 5 1 1 3 6 6 2 1 2 1 5 6 3 2 5 1	Case #1: yes yes Case #2: no yes Case #3: no yes

Sample Input 2

```
8
4 4
0 2 1
0 5 1
2 4 1
4 5 1

6 2
0 2 2
2 2 4

6 2
0 4 4
4 4 2

6 5
1 2 1
1 3 2
1 4 1
2 3 1
3 4 1

6 4
0 4 2
0 5 1
4 5 1
5 5 2

4 4
0 5 1
1 4 1
4 4 1
4 5 1

6 4
0 4 1
1 3 1
3 3 1
3 4 3

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

Sample Output 2

```
Case #1: yes yes
Case #2: yes yes
Case #3: yes yes
Case #4: yes yes
Case #5: no yes
Case #6: no yes
Case #7: no yes
Case #8: no yes
```

This page is intentionally left blank.

Problem E
Stargazing

One of Lea's favorite things to do after a stressful week is to sit outside at night and simply look at the stars. In her early childhood her dad taught her all about the different star constellations and to this day Lea can still recall most of them.

Tonight is a boring night though - neither Andromeda, nor her other favorite constellation, the Great Bear, are visible! However, after some time Lea notices that all constellations visible tonight form trees as defined in Lea's graph theory class at university.

Back in the days Lea's father always challenged her to find stars in each of these tree-like constellations that fulfill certain properties. Lea's favorite challenge was to spot a star which, if removed from the constellation, would split the tree in a forest such that none of the remaining trees in the forest contain more than half of the stars of the original constellation. Unfortunately, a star with this particular property is not so easy to spot and Lea is not even sure if such a star even always exists. Therefore, she decides to write a short computer program that does the job for her.

Input

The first line of the input contains an integer t. t star constellations follow, each of them separated by a blank line.

Each constellation starts with one integer n, where n is the number of stars (labeled from 0 to $n-1$) in the particular constellation. The following $n-1$ lines describe the lines connecting the stars in the constellation. Each line contains two integers x_i and y_i denoting that the stars labeled x_i and y_i are connected by a line.

You may safely assume that each constellation forms a tree, i.e. an acyclic connected graph.

Output

For each test case, output one line containing "Case #i: c" with i being the number of the test case starting at 1, and c being the label of the star with the desired property, or `impossible` if no such star exists. If there are multiple stars that fulfill the desired property, you can choose one of them arbitrarily.

Constraints

- $1 \leq t \leq 10$
- $2 \leq n \leq 10^5$
- $0 \leq x_i, y_i < n$ for all $1 \leq i \leq n$

Sample Input 1	Sample Output 1
3 4 0 1 1 2 2 3 5 0 1 0 2 0 3 0 4 8 0 1 1 2 2 3 1 4 2 5 4 6 4 7	Case #1: 1 Case #2: 0 Case #3: 1

Problem F
Catch them all!

One of Jhonny's favorite things to do on the weekend is to get out his old console, the GameGirl, and play his favorite game: Mainzelmon! The goal of the game is to catch little monsters, so-called Mainzelmons, and train them by letting them fight against the ones from other players. If a Mainzelmon reaches a certain number of experience points, it turns into another Mainzelmon which is stronger and looks slightly different. Jhonny was already able to catch the cute but rather weak Mainzelmon FAUbsi. After some intense fights against his enemy player, team RocKIT, his FAUbsi turned into a FAUboga, and eventually even became a FAUboss. One thing that's bugging Jhonny is that he has never managed to catch the strongest and most seldom Mainzelmon of the game: MewTUM!

Recently his friend Paul told him that MewTUM can only be caught if one owns at least one Mainzelmon of every available Mainzelmon type in the game. Unfortunately, Jhonny has lost track of which Mainzelmon types he already has. Luckily, the game provides an option to print out a list of all the Mainzelmons that a player has caught so far. However, this list is not sorted and also contains some Mainzelmon types more than once, as Jhonny has caught Mainzelmons of his favorite types multiple times. Since Jhonny is a particularly lousy programmer, he asks you to write a program which outputs a sorted list containing each type of Mainzelmon he owns only once. Further, he would like to compute some values for statistical purposes. The exact formats of the input and output lists are described below.

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 10^4$) which is the total number of Mainzelmons that Jhonny owns.
- The following n lines describe Jhonny's Mainzelmons further in detail. Each line consists of an integer x_i, a string s_i, an integer k_i, and a double c_i, where
  - x_i ($1 \leq x_i \leq 10^4$) is the ID of the i-th Mainzelmon,
  - s_i ($1 \leq |s| \leq 20$) is the type of the i-th Mainzelmon,
  - k_i ($1 \leq k \leq 10^9$ is the number of experience points of the i-th Mainzelmon)
  - c_i ($1 \leq c_i \leq 10^3$) is monetary value of the i-th Mainzelmon.

Each Mainzelmon type is identified with a distinct ID, and Mainzelmons of the same type share the same ID. The Mainzelmon types solely consist of small and capitalized letters of the English alphabet.

Output

The output consists of:

- One line with an integer m, an integer ℓ, and a double d, where
  - m is the number of distinct Mainzelmon types that Jhonny owns.
  - ℓ is the sum of experience points of all Mainzelmons Jhonny owns.
  - d is the average monetary value of all Mainzelmons Jhonny owns. The average should be correct up to an absolute or relative error of at most 10^{-2}.
- m lines each containing an integer x_i and a string s_i, where
  - x_i is the ID of the i-th Mainzelmon type, and
  - s_i is the name of the corresponding type.

The m lines should be sorted in ascending order by the ID.

Sample Input 1

```
3
16 FAUbsi 110 10.00
25 Peecrapchew 20 15.00
16 FAUbsi 70 5.11
```

Sample Output 1

```
2 200 10.036666666666667
16 FAUbsi
25 Peecrapchew
```

Sample Input 2

```
4
18 FAUboss 200 564.86
9 TUMtok 350 653.72
17 FAUboga 150 58.59
10 FAUpy 30 1.01
```

Sample Output 2

```
4 730 319.54499999999996
9 TUMtok
10 FAUpy
17 FAUboga
18 FAUboss
```

Problem G
Attack on Alpha-Zet

Space pirate Captain Krys has recently acquired a map of the artificial and highly secure planet Alpha-Zet which he has been planning to raid for ages. It turns out the whole planet is built on a 2D plane with modules that serve as one room each. There is exactly one module at every pair of integer coordinates and modules are exactly 1×1 units big. Every module is bidirectionally connected to at least one adjacent module. Also, for any two modules there exists exactly one path between them. All in all the modules create a rectangular maze without any loops.

On the map Captain Krys has marked several modules he wants to visit in exactly the marked order. What he intends to do there is none of your business, but he promises you a fortune if you determine the number of modules he has to walk through along the route (since there are no loops he will always take the direct route from one marked module to the next). The first marked module indicates where he starts his journey, the last where he wants to finish.

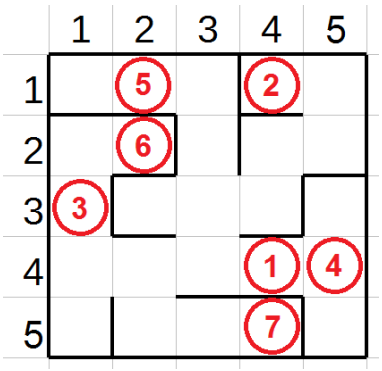


Figure G.1: Illustration of test case 2 of sample input 1

Input

The first line of the input contains an integer t. t test cases follow, each of them separated by a blank line.

The input consists of one line with two integers h and w describing the height and the width of the maze. $h + 1$ lines follow, describing the maze in ASCII, each line containing $2 \cdot w + 1$ characters. The description always follows these rules:

- In every row, columns with odd index (starting at index 1) contain either vertical walls or spaces and columns with even index contain either horizontal walls or spaces.
- The first row describes the northern wall of the maze (which always consists only of horizontal walls). Every subsequent row describes a row of modules.
- A module is located at every even column index. Its western and eastern walls are located at the directly neighboring odd column indices respectively, its northern wall is located at the same column index but one row above and its southern wall can be found at its own position. If a wall is missing, the corresponding position contains a space instead.

After the description of the maze, an integer m is given. Each of the following m lines describes a marked module with two integer coordinates x and y. The first pair of coordinates is the start point of the journey, the last pair the end point. Modules may appear multiple times but never twice or more in a row. $(1, 1)$ is the top left module and (h, w) is the bottom right module.

It is guaranteed that the maze itself is enclosed. Furthermore it is guaranteed that exactly one path exists between any two modules.

Output

Output one integer, the number of modules Captain Krys has to travel through if he follows the route in the exact order given in the input.

Constraints

- $1 \leq t \leq 2$
- $2 \leq h, w \leq 1000$

- $2 \leq m \leq 10^4$
- $1 \leq x \leq h$
- $1 \leq y \leq w$

Sample Input 1

```
2
2 6
 _ _ _ _ _ _
|  _ _ _ _ _|
|_ _ _ _ _ _|
5
1 5
1 1
1 6
1 1
1 5

5 5
 _ _ _ _ _
|_ _   |_  |
|  _|  |  _|
| |_    _| |
|     _ _  |
|_|_ _ _|_|
7
4 4
1 4
3 1
4 5
1 2
2 2
5 4
```

Sample Output 1

```
Case #1: 18
Case #2: 43
```

Problem H
Commander-in-Chief

A rainy Sunday afternoon. What should Lea and her friends do? They wanted to go tightrope walking between two mountain summits, but this is not going to happen with rain pouring down on them, and making the rope slippery and all. So, they decide to stay in and dust off an old warfare strategy board game they played a lot when they were younger. Epic battles between huge armies, invading continents, conquering new worlds. Though, all of this is only possible for the best players. And Lea really does want to be one of those players. Roughly explained, the game is about stationing armies on different regions on a map, and letting them fight against opposing armies from a neighbouring region in order to try to conquer that region.

As far as she can remember, the best strategy for her was to subdivide each army into smaller squadrons and letting each of them fight on a different battleground. However, the real strategic trick in this game was that each of these squadrons should be of the same size, even if they belong to different armies. To maximise Lea's probability to conquer the world, the squadrons should be as big as possible, and no troops should be left behind without a squadron. Luckily, Lea has time to calculate this squadron size during the other players' turns.

Input

The first line of the input contains an integer t. t test cases follow, each of them separated by a blank line.

Each test case starts with a line containing an integer n, the number of armies Lea commands. The next line contains n space-separated integers $a_1 \dots a_n$ denoting the sizes of the n armies.

Output

For each test case, output one line containing "Case #i: x" where i is its number, starting at 1, and x is the biggest possible squadron size as described above. Each line of the output should end with a line break.

Constraints

- $1 \leq t \leq 500$
- $2 \leq n \leq 20$
- $1 \leq a_i \leq 10^9$ for all $1 \leq i \leq n$

Sample Input 1	Sample Output 1
2 3 12 6 21 4 24 7 12 18	Case #1: 3 Case #2: 1

Sample Input 2	Sample Output 2
6 5 35 19 37 11 8 2 48 48 3 44 41 26 3 33 44 33 5 9 18 45 36 36 3 29 29 29	Case #1: 1 Case #2: 48 Case #3: 1 Case #4: 11 Case #5: 9 Case #6: 29

Problem I
Pizza Party

Lea has decided that it is time to host another party! She invited all of her friends and ordered a huge amount of pizza. When the pizza boy brought the food, she was surprised: The pizza was an equilateral triangle!

Making the best out of the situation, she decided to cut the pizza into equally sized equilateral triangles, one for each party attendee. Unsure whether this is possible without leftovers, she asked you to do the calculations. In the event that it is not possible, tell her how many more friends she has to invite over to finish the pizza completely.

Input

The first line of the input contains an integer t. t test cases follow.

Each test case consists of a single line containing an integer n, the number of people at the party.

Output

For each test case, print a line containing "Case #i: x" where i is its number, starting at 1, and x is the number of people she has to invite additionally to those already present. Each line of the output should end with a line break.

Constraints

- $1 \leq t \leq 50$
- $1 \leq n \leq 10^{18}$

Sample Input 1	Sample Output 1
3 2 4 7	Case #1: 2 Case #2: 0 Case #3: 2

This page is intentionally left blank.

Problem J
Absurdistanian Plots

When Lea went back to Absurdistan to visit some friends, she noticed something very strange. Apparently, there is no central authority checking the allocation of parcels and it may happen that several plots of land intersect. Lea recalls that the Absurdistanians are very superstitious, so she read the local laws concerning plot allocation. It turns out that, since round shapes are evil, only polygon-shaped, non self-intersecting plots are allowed. Furthermore, intersections between two plots are strictly prohibited - should two parcels intersect, the owners have to pay a hefty fine.

While visiting her friends, she wanted to make sure that their parcels don't intersect or if they indeed intersect, at least warn her friends and tell them how much they would have to pay if anyone found out. Can you help her determine how large the area of intersection between the two plots is?

Input

The first line of the input contains an integer t. t test cases follow.

Each test case starts with two integers n and m, the number of vertices of the first and second plot, respectively. n lines follow describing the vertices of the first plot. The i-th line contains two real numbers x_i^1 and y_i^1, the x and y coordinates of the i-th vertex, respectively. Then, m lines follow describing the vertices of the second plot analogously. No two intersecting edges are parallel and none of the two polygons is self-intersecting. The points of each polygon may be in clockwise or counter-clockwise order.

Output

For each test case, output one line containing "Case #i: a" where i is its number, starting at 1, and a is the intersection area between the two polygons, up to a precision of 10^{-4}. Each line of the output should end with a line break.

Constraints

- $1 \leq t \leq 20$
- $1 \leq n, m \leq 200$
- $-1000 \leq x_i^1, x_i^2, y_i^1, y_i^2 \leq 1000$

Sample Input 1

```
1
4 4
0 0
1 0
1 1
0 1
0.5 0.5
1.5 0.5
1.5 1.5
0.5 1.5
```

Sample Output 1

```
Case #1: 0.2500000000
```

Sample Input 2	Sample Output 2
1 8 4 -1 -1 2 -1 2 -2 -2 -2 -2 2 2 2 2 1 -1 1 1 -3 1 3 0 3 0 -3	Case #1: 2.0000000000