Code Jam 2020 - Round 3

Naming Compromise

Problem

Cameron and Jamie are about to welcome a second baby into their lives. They are already good at working together as parents, but right now they are disagreeing about one crucial thing! Cameron wants to name the baby one name (the string **C**), whereas Jamie wants to name the baby something else (the string **J**).

You want to help them find a *compromise name* that is as close as possible to what each of them wants. You think you can do this using the notion of *edit distance*. The edit distance between two strings S_1 and S_2 is the minimum number of operations required to transform S_1 into S_2, where the allowed operations are as follows:

- Insert one character anywhere in the string.
- Delete one character from anywhere in the string.
- Change one character in the string to any other character.

For example, the edit distance between `CAMERON` and `JAMIE` is 5. One way to accomplish the transformation in 5 steps is the following: `CAMERON` to `JAMERON` (change) to `JAMIERON` (insert) to `JAMIEON` (delete) to `JAMIEN` (delete) to `JAMIE` (delete). Any transformation from `CAMERON` into `JAMIE` requires at least this many operations.

To make the compromise name N as close as possible to the original desires of the parents, you want N to be a non-empty string such that the sum of the edit distances between **C** and N and between **J** and N is as small as possible. Out of all those choices for N, to make sure the compromise is fair, you must choose an N such that the difference between those two edit distances is also as small as possible. Please find a compromise name for Cameron and Jamie.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each case consists of a single line with two strings **C** and **J**: the names that Cameron and Jamie have proposed for the baby, respectively. Each of these names is made up of uppercase English alphabet letters.

Output

For each test case, output one line containing `Case #x: y`, where `x` is the test case number (starting from 1) and `y` is a name that meets the requirements mentioned in the statement. Note that `y` must contain only uppercase English letters.

Limits

Time limit: 20 seconds per test set.
Memory limit: 1GB.
$1 \leq \mathbf{T} \leq 100$.
$\mathbf{C} \neq \mathbf{J}$.

Test Set 1 (Visible Verdict)

1 ≤ length of **C** ≤ 6.
1 ≤ length of **J** ≤ 6.
The i-th letter of **C** is an uppercase `X`, `Y`, or `Z`, for all i.
The i-th letter of **J** is an uppercase `X`, `Y`, or `Z`, for all i.

Test Set 2 (Hidden Verdict)

1 ≤ length of **C** ≤ 60.
1 ≤ length of **J** ≤ 60.
The i-th letter of **C** is an uppercase English alphabet letter, for all i.
The i-th letter of **J** is an uppercase English alphabet letter, for all i.

Sample

Sample Input	Sample Output
4 XYZZY ZZYZX Y Z YYXXYZ ZYYXXY XZXZXZ YZ	Case #1: ZZY Case #2: Z Case #3: ZYYXXYZ Case #4: ZYZX

The above cases meet the limits for Test Set 1. Another sample case that does not meet those limits appears at the end of this section.

In Sample Case #1, the edit distance from `XYZZY` to `ZZY` is 2 (delete the first two letters), and the edit distance from `ZZYZX` to `ZZY` is 2 (delete the last two letters). `XZZX` and `ZYYZY` would also work. No possible name has a sum of edit distances that is less than 4.

`ZY`, for example, has the same edit distance to **C** as to **J** (3, in each case). However the sum of those distances would be 6, which is not minimal, so it would not be an acceptable answer.

`XZZY` is also unacceptable. Its edit distances to **C** and **J**, respectively, are 1 and 3. The sum of those edit distances is minimal, but the difference between the two (|1-3| = 2) is not minimal, since we have shown that it is possible to achieve a difference of 0.

In Sample Case #2, `Y` and `Z` are the only acceptable answers.

In Sample Case #3, notice that input length restrictions do not apply to the output, so the shown answer is acceptable in either test set. Another possible answer is `YYXXY`.

In Sample Case #4, the edit distance between `XZXZXZ` and `ZYZX` is 3, and the edit distance between `YZ` and `ZYZX` is 2. The sum of those edit distances is 5, and their difference is 1; these values are optimal for this case.

The following additional case could not appear in Test Set 1, but could appear in Test Set 2.

```
1
GCJ ABC
```

`Case #1: GC` is one of the possible correct outputs.

Code Jam 2020 - Round 3

Thermometers

Problem

You are part of a team of researchers investigating the climate along the coast of an island. The island's coast is modeled as a circle with a circumference of **K** kilometers. There is a lighthouse on the coast which occupies a single point on the circle's circumference. Each point on the coast is mapped to a real number in the range [0, **K**); formally, point x is the point on the coast that is x kilometers away from the lighthouse when walking clockwise along the coast. For example, if **K** = 5, point 0 is the point where the lighthouse is, point 1.5 is the point that is 1.5 kilometers away from the lighthouse in the clockwise direction, and point 2.5 is the point that is located at the diametrical opposite of the lighthouse.

You are in charge of studying coastal temperatures. Another team installed a coastal temperature measuring system that works as follows: a number of thermometers were deployed at specific points to measure the temperature at those points. No two thermometers were placed at the same point. In that team's model, points without thermometers are assumed to have the same temperature as the one measured by the closest thermometer. For points that are equidistant from two thermometers, the thermometer in the clockwise direction is used (the first one you would encounter if walking clockwise from the point).

Unfortunately, you do not know how many thermometers the system used or where they were placed, but you do have access to the system's temperature data. It is given as two lists of **N** values each $\mathbf{X_1}$, $\mathbf{X_2}$, ..., $\mathbf{X_N}$ and $\mathbf{T_1}$, $\mathbf{T_2}$, ..., $\mathbf{T_N}$, representing that each point x where $\mathbf{X_i} \leq x < \mathbf{X_{i+1}}$ is assigned temperature $\mathbf{T_i}$, for each $1 \leq i < \mathbf{N}$, and each point x where $0 \leq x < \mathbf{X_1}$ or $\mathbf{X_N} \leq x < \mathbf{K}$ is assigned temperature $\mathbf{T_N}$. The points are enumerated in the clockwise direction, so $\mathbf{X_i} < \mathbf{X_{i+1}}$, for all i.

You want to determine the smallest number of thermometers that, when placed in some set of locations, could have produced the observed data.

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow; each consists of three lines. The first line of a test case contains two integers **K** and **N**: the circumference of the island and the size of the lists representing the temperature data. The second line contains **N** integers $\mathbf{X_1}$, $\mathbf{X_2}$, ..., $\mathbf{X_N}$. The third line contains **N** integers $\mathbf{T_1}$, $\mathbf{T_2}$, ..., $\mathbf{T_N}$. The way in which the integers in the second and third line represent the temperatures is explained above.

Output

For each test case, output one line containing `Case #x: y`, where `x` is the test case number (starting from 1) and `y` is the minimum number of thermometers that could have produced the observed input data, as described above.

Limits

Time limit: 30 seconds per test set.
Memory limit: 1 GB.
$1 \leq \mathbf{T} \leq 100$.
$2 \leq \mathbf{N} \leq \min(100, \mathbf{K})$.

$0 \le \mathbf{X_1}$.
$\mathbf{X_i} < \mathbf{X_{i+1}}$, for all i.
$\mathbf{X_N} < \mathbf{K}$.
$184 \le \mathbf{T_i} \le 330$, for all i.
$\mathbf{T_i} \ne \mathbf{T_{i+1}}$, for all i.
$\mathbf{T_1} \ne \mathbf{T_N}$.

Test Set 1 (Visible Verdict)

$2 \le \mathbf{K} \le 10$.

Test Set 2 (Hidden Verdict)

$2 \le \mathbf{K} \le 10^9$.

Sample

Sample Input

```
3
2 2
0 1
184 330
3 2
0 1
184 330
10 3
1 5 9
184 200 330
```

Sample Output

```
Case #1: 2
Case #2: 3
Case #3: 3
```

In Sample Case #1, at least 2 thermometers are needed because there are two different temperatures measured. It is possible to produce the data using exactly 2 thermometers, with one thermometer (measuring 184) at point 0.5 and another (measuring 330) at point 1.5. Note that point 0 and point 1 are equidistant from both thermometers, so the thermometer in the clockwise direction is used. The temperature measured at point 0 comes from the thermometer at point 0.5 and the temperature measured at point 1 comes from the thermometer at point 1.5.



The data from Sample Case #2 could not be produced with just 2 thermometers. It could be produced with 3 thermometers if they were placed at point 0.2, point 1.8, and point 2.8, measuring 184, 330 and 330, respectively. There are other ways to place 3 thermometers that would also yield the input data.



In Sample Case #3, one way to produce the data with 3 thermometers is to place them at point 0, point 2 and point 8, measuring 330, 184 and 200, respectively.



Code Jam 2020 - Round 3

Pen Testing

Problem

You have **N** ballpoint pens. You know that each has a distinct integer number of units of ink between 0 and **N**-1, but the pens are given to you in random order, and therefore you do not know which pen is which.

You are about to go on a trip to the South Pole (where there are no pens), and your luggage only has room for two pens, but you know you will need to do a lot of important postcard writing. Specifically, the two pens you choose must have a total of at least **N** ink units.

Your only way to get information about the pens is to choose one and try writing something with it. You will either succeed, in which case the pen will now have one unit of ink less (and is now possibly empty), or fail, which means that the pen already had no ink left. You can repeat this multiple times, with the same pen or different pens.

Eventually, you must select the two pens to take on your trip, and you succeed if the total amount of ink remaining in those two pens is at least **N** units.

You will be given **T** test cases, and you must succeed in at least **C** of them. Note that all test sets in this problem are Visible.

Input and output

This is an interactive problem. You should make sure you have read the information in the Interactive Problems section of our FAQ.

Initially, your program should read a single line containing three integers **T**, **N**, and **C**: the number of test cases, the number of pens, and the minimum number of test cases you must succeed in. (Note that the value of **N** is the same for all test sets, and is provided as input only for convenience; see the Limits section for more details.)

Then, your program needs to process all **T** test cases at the same time (this is done to reduce the number of roundtrips between your solution and the judging program). The interaction is organized into rounds.

At the beginning of each round, your program must print one line containing **T** integers: the **i**-th integer is the number of the pen you want to try writing with in the **i**-th test case, or 0 if you do not want to write with any pen in this test case in this round. The pens are numbered from 1 to **N**.

Be aware that flushing the output buffer after each one of these integers, instead of only once after printing all **T**, could cause a Time Limit Exceeded error because of the time consumed by the flushing itself.

The judge responds with one line containing **T** integers: the **i**-th integer is the amount of ink spent in the **i**-th test case in this round. It will be equal to 1 if the writing in the **i**-th test case was successful. Otherwise, it will be equal to 0, which could mean that you tried to write in the **i**-th test case but the pen you chose had no ink left, or that you did not try to write in the **i**-th test case at all.

You may participate in at most **N**×(**N**+1)/2 rounds. Note that this is enough to be confident that all pens are empty.

When your program is ready to submit an answer for all test cases, it must print a line containing the number 0 **T** times. This line is not counted towards the limit on the number of rounds, and the judge will not send a response.

Then, your program must print another line with 2×**T** integers: the (2×**i**-1)-th and the (2×**i**)-th integers in this line are the distinct numbers of the pens that you take to the South Pole in the **i**-th test case. The judge will not send a response, and your program must then terminate with no error.

If the judge receives unexpected output from your program at any moment, the judge will print a single number -1 and not print any further output. If your program continues to wait for the judge after receiving a -1, your program will time out, resulting in a Time Limit Exceeded error. Notice that it is your responsibility to have your program exit in time to receive a Wrong Answer judgment instead of a Time Limit Exceeded error. As usual, if the memory limit is exceeded, or your program gets a runtime error, you will receive the appropriate judgment.

You can assume that the pens are given to you in random order. These orders are chosen uniformly at random and independently for each test case and for each submission. *Therefore even if you submit exactly the same code twice the judge will use different random orders*.

Limits

Time limit: 90 seconds per test set.
Memory limit: 1GB.
N = 15.

Test Set 1 (Visible Verdict)

T = 20000.
C = 10900 (**C**=0.545×**T**).

Test Set 2 (Visible Verdict)

T = 20000.
C = 12000 (**C**=0.6×**T**).

Test Set 3 (Visible Verdict)

T = 100000.
C = 63600 (**C**=0.636×**T**).

Testing Tool

You can use this testing tool to test locally or on our platform. To test locally, you will need to run the tool in parallel with your code; you can use our interactive runner for that. For more information, read the instructions in comments in that file, and also check out the Interactive Problems section of the FAQ.

Instructions for the testing tool are included in comments within the tool. We encourage you to add your own test cases. Please be advised that although the testing tool is intended to simulate the judging system, it is **NOT** the real judging system and might behave differently. If your code passes the testing tool but fails the real judge, please check the Coding section of the FAQ to make sure that you are using the same compiler as us.

Download testing tool

Sample Interaction

The following interaction does not correspond to any of the three test sets, as its values of **T** and **N** are too small. It merely serves to demonstrate the protocol.

Input to your program	Output of your program
2 5 1	
	4 5
1 0	
	4 3
0 1	
	0 2
0 1	
	0 0
	3 4 3 4

Here is the same interaction, explained:

```
// The following reads 2 into t, 5 into n and 1 into c.
t, n, c = readline_int_list()
// The judge secretly picks the number of units for each pen:
// in test case 1: 2 0 4 1 3
// in test case 2: 1 3 2 4 0
// We write with the 4-th pen in test case 1, and with the 5-th pen in test case 2.
printline 4 5 to stdout
flush stdout
// Reads 1 0, as the 4-th pen in test case 1 still had ink left,
// but the 5-th pen in test case 2 did not.
a1, a2 = readline_int_list()
// We write with the 4-th pen in test case 1 again, and with the 3-rd pen in test case 2.
printline 4 3 to stdout
flush stdout
// Reads 0 1.
a1, a2 = readline_int_list()
// We only write in test case 2 this time, with the 2-nd pen.
printline 0 2 to stdout
flush stdout
// Reads 0 1.
a1, a2 = readline_int_list()
// We decide we are ready to answer.
printline 0 0 to stdout
flush stdout
// We take the 3-rd and the 4-th pens to the South Pole in both test cases.
printline 3 4 3 4 to stdout
flush stdout
// In test case 1, the remaining amounts in the 3-rd and the 4-th pens are 4 and 0, and 4+0<5,
```

```
// so we did not succeed.
// In test case 2, the remaining amounts in the 3-rd and the 4-th pens are 1 and 4, and 1+4≥5,
// so we succeeded.
// We have succeeded in 1 out of 2 test cases, which is good enough since c=1.
exit
```

Code Jam 2020 - Round 3

Recalculating

Problem

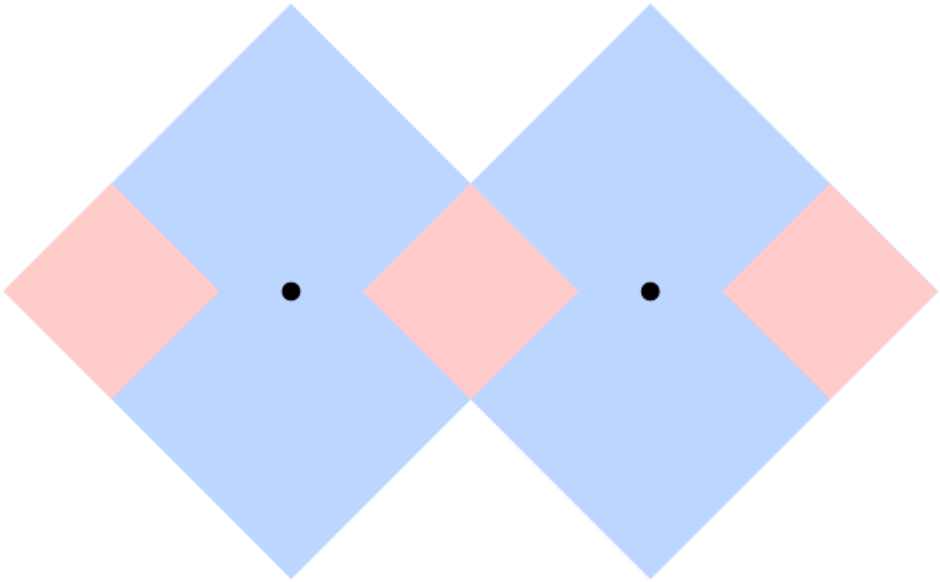
You are working for the Driverless Direct Delivery Drone Directions Design Division of Apricot Rules LLC. The company is about to take its first drone "Principia" to market. You are tasked with designing backup systems for Principia, in case it loses access to its primary geolocation systems (like GPS), but it still needs a way to get directions. Principia is designed for use on a flat region; formally, the region is a Cartesian plane in which the coordinates are in meters. One or more points on this plane are *drone repair centers*. No two drone repair centers are at the same location.

Principia has a system that can retrieve the relative locations of drone repair centers that are within an L_1 distance (which is also known as Manhattan distance) of at most **D** meters of its location. The information retrieved is a set of repair center locations relative to Principia's current location. For example: "there is a repair center 4 meters north and 3.5 meters west, and another one 2.5 meters east". Notice that the information does not identify repair centers; it gives their locations relative to Principia.

You quickly realized that there may be points on the map where this information may not be enough for Principia to uniquely determine its current location. This is because there might be two (or more) different points from which the information looks the same. Points with this property are called *non-distinguishable*, while all other points are called *distinguishable*.

Formally, the information retrieved by Principia when located at point (x, y) is Info(x, y) := the set of all points (z - x, w - y), where (z, w) is the location of a repair center and $|z - x| + |w - y| \leq$ **D**. Here |z - x| and |w - y| denote the absolute values of z - x and w - y, respectively. A point (x_1, y_1) is non-distinguishable if and only if there exists another point (x_2, y_2) such that $\text{Info}(x_1, y_1) = \text{Info}(x_2, y_2)$.

For example, suppose **D**=4 and there are repair centers at points (0, 0) and (5, 0). The point (0, 0) is non-distinguishable because Info(0, 0)={(0, 0)}=Info(5, 0). This means that point (5, 0) is also non-distinguishable. On the other hand, Info(3.5, 0.1)={(-3.5, -0.1), (1.5, -0.1)} is not equal to the information from any other point, which means that point (3.5, 0.1) is distinguishable. The following picture illustrates the regions of distinguishable points (in red) and non-distinguishable points (in blue):



Principia is deployed to a point that is chosen uniformly at random from the set of all points that are within **D** meters (using L_1 distance) of at least one repair center — that is, the set of all points (x, y) such that Info(x, y) is non-empty. The probability of that choice belonging to a given continuous set of points S is proportional to the number of square meters of S's area. In the example above, each red square has an area of 4.5 square meters, while each blue section has an area of 23 square meters. Therefore, the probability of Principia being deployed within each red square is 4.5/(3×4.5 + 2×23) and the probability of it being deployed within each blue section is 23/(3×4.5 + 2×23). Since the border between adjacent differently-colored sections has area equal to 0, the probability of Principia being deployed exactly on the border is exactly 0.

Given the locations of all repair centers, what is the probability that the point to which Principia is deployed is distinguishable?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each test case starts with a line containing two integers **N** and **D** representing (respectively) the number of repair centers and the maximum L_1 distance from which Principia can retrieve information from a repair center, as described above. Then, **N** lines follow. The i-th of these contains two integers $\mathbf{X_i}$ and $\mathbf{Y_i}$ representing the coordinates of the i-th repair center. The unit of measurement for all coordinates and **D** is meters.

Output

For each test case, output one line containing `Case #x: y z`, where `x` is the test case number (starting from 1) and `y` and `z` are non-negative integers. The fraction `y/z` must represent the probability of Principia being at a distinguishable location, if one is chosen uniformly at random from all locations that are within **D** meters of at least one repair center (using L_1 distance). If there are multiple acceptable values for `y` and `z`, choose the one such that `z` is minimized.

Limits

Memory limit: 1GB.
$1 \le \mathbf{T} \le 100$.
$1 \le \mathbf{D} \le 10^7$.
$-10^9 \le \mathbf{X_i} \le 10^9$, for all i.

$-10^9 \leq \mathbf{Y_i} \leq 10^9$, for all i.
$(X_i, Y_i) \neq (X_j, Y_j)$ for all $i \neq j$. (No two repair centers share the same location.)

Test Set 1 (Visible Verdict)

Time limit: 20 seconds.
N = 2.

Test Set 2 (Visible Verdict)

Time limit: 60 seconds.
$2 \leq \mathbf{N} \leq 10$.

Test Set 3 (Visible Verdict)

Time limit: 120 seconds.
For 6 cases, **N** = 1687.
For **T**-6 cases, $2 \leq \mathbf{N} \leq 100$.

Sample

Sample Input

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

Sample Output

```
Case #1: 27 119
Case #2: 0 1
Case #3: 0 1
Case #4: 1 5
```

The above cases meet the limits for Test Set 1. Another sample case that does not meet those limits appears at the end of this section.

Sample Case #1 is described and depicted in the statement.

The points in the middle red region are all distinguishable points because they are the only points that retrieve information from both repair centers, and each point in that region retrieves a distinct set of information.

The points in the left and right red region each receive information from only one repair center, but that information is always unique, so these are all distinguishable points. For example, if Principia knows it is 3 meters east of a repair center, it can be sure it is not 3 meters east of the repair center at (0, 0), because then it would have retrieved information from both repair centers. So it must be 3 meters east of the repair center at (5, 0).

The points in the blue regions are all non-distinguishable points. Choose any point in one of those regions, and consider the information that Principia would get from that point. It contains

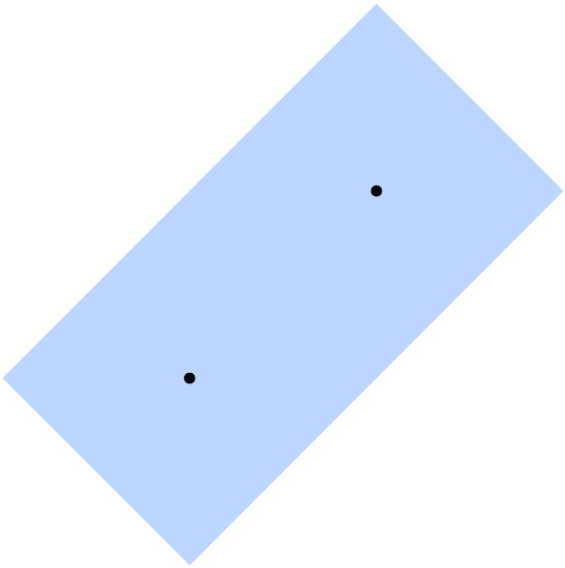
only the one repair center in range. But, there is a corresponding point in the other blue region from which Principia would get exactly the same information.

As explained above, the probability of Principia being deployed to one of the red sections is 4.5/59.5, so the total probability of it being deployed to any of them is 3×4.5/59.5 = 27/119.

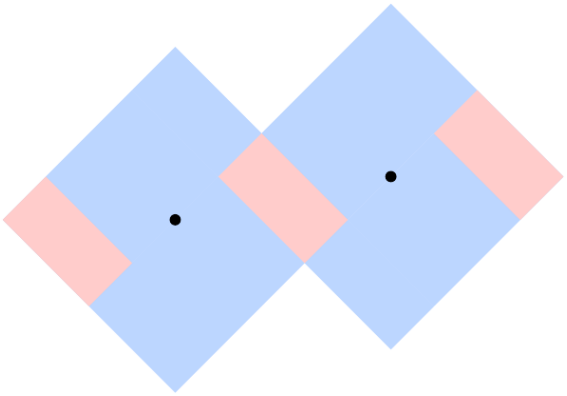
The following picture illustrates Sample Case #2. There is no way to retrieve information from more than one repair center, so every point close enough to one of them is non-distinguishable; the same information is retrieved from an equivalent point near the other one. Remember that `z` (the denominator) must be minimal, so `0 1` is the only acceptable answer.



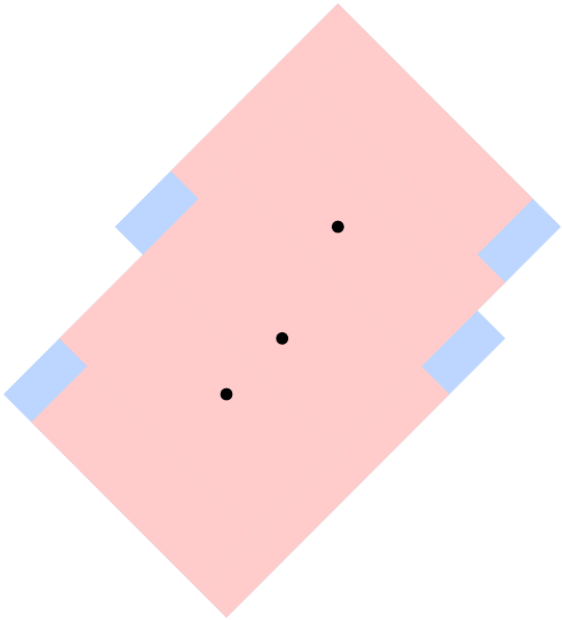
The following picture illustrates Sample Case #3. Notice that the border between the two blue squares consists of distinguishable points. However, since its area is 0, the probability of Principia being deployed there is 0. All other points where Principia can be deployed are non-distinguishable.



The following picture illustrates Sample Case #4.



The following picture illustrates the additional case.



The following additional case could not appear in Test Set 1, but could appear in any of the other test sets.

```
1
3 4
0 0
1 1
2 3
```

The correct output is `Case #1: 101 109`.