Algorithms for Programming Contests WS20/21 - Week 13

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

Mikhail Raskin, Martin Helfrich, Maxi Weininger, Christoph Welzel

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

Wednesday, 17.02.2021, 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:

You may work together in teams of two students.

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
Soda Slurper

Tim is an absolutely obsessive soda drinker, he simply cannot get enough. Most annoyingly though, he almost never has any money, so his only obvious legal way to obtain more soda is to take the money he gets when he recycles empty soda bottles to buy new ones. In addition to the empty bottles resulting from his own consumption he sometimes finds empty bottles in the street. One day he was extra thirsty, so he actually drank sodas until he could not afford a new one anymore.

Input

The first line of the input contains an integer t. t test cases follow. Each test case consists of a single line containing three integers e f c, where e is the number of empty soda bottles in Tim's posession at the start of the day, f is the number of empty soda bottles found during the day and c is the number of empty bottles required to buy a new soda.

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 sodas Tim drunk on his extra thirsty day. Each line of the output should end with a line break.

Constraints

- $1 \leq t \leq 20$
- $0 \leq e \leq 1000$
- $0 \leq f \leq 1000$
- $2 \leq c \leq 2000$

Sample Input 1	Sample Output 1
2 9 0 3 5 5 2	Case #1: 4 Case #2: 9

This page is intentionally left blank.

Problem B
Jungle Network

On the flight home after her last holidays, Lea's plane crashed somewhere in the jungle. Luckily, everybody on board survived, but they have no food except for what they found in the wreckage of the aircraft (which is not that good since Lea chose a very cheap airline).

The radio set which is built into the plane and some batteries are the only thing they have left to call for help. It turns out there are lots of radio sets in the jungle run by the tribes living there, but their supply of batteries is limited and they therefore don't use them often. It would be great if each village was able to pass messages to each other village. The people there love to help each other and forward messages through the radio network until they reach their recipients. Help the people in the jungle to create such a network by telling them which villages need to establish a radio connection in a way that the power consumption is minimal. In return the villages' inhabitants promised to forward your emergency call out of the jungle.

You will be given the coordinates of all villages owning a radio device. To establish a connection between two villages, each of them needs power equal to the square of their distance. Note that the power consumption is not depending on the number of messages sent. To make it even worse, some of the devices are not in best condition, so the inhabitants also tell you the maximum amount of power the devices may use without breaking. Note that each device may establish several connections, each of them using at most the maximum power level. You will join the network as each village will do, too. Your plane has crashed at coordinates $(0, 0)$ and since you have a very modern device it has no limits in power consumption as long as the recipient is located in the jungle. All villages and the plane have enough batteries remaining to power as many connections with their device as they want, but they want to minimize the total power consumption.

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 single line containing n, the number of villages. n lines follow. The i-th line consists of three integers x_i, y_i, and c_i where x_i and y_i are the village's coordinates and c_i is the maximum power level their radio device can consume.

Output

For each test case, print a line containing "Case #i: x" where i is its number, starting at 1, and x is the minimum total power all radio sets consume to set up the network, or the string "impossible" if there is no way to connect all villages.

Constraints

- $1 \leq t \leq 20$
- $1 \leq n \leq 1000$
- $-100 \leq x_i, y_i \leq 100$ for all $1 \leq i \leq n$
- $0 \leq c_i \leq 40000$ for all $1 \leq i \leq n$
- There won't be two distinct villages at the same position and no village will be at position $(0, 0)$.

Sample Input 1	Sample Output 1
`` 11 1 1 1 10 2 0 1 1 0 2 10 1 1 1 1 7 -32 -44 25382 -7 -63 19323 28 95 12389 -56 36 17439 -21 -4 2197 76 95 36065 48 -58 23070 3 30 -94 5076 -39 98 9331 -41 11 9577 5 -46 -31 24927 -66 -48 6370 -46 -49 12710 5 15 38589 -92 88 23578 2 -24 -80 8210 -10 88 36990 2 -86 -83 31777 -80 39 23565 6 19 -73 21839 62 -20 4721 -94 -47 5502 -97 71 37975 -89 -55 35212 -68 -1 22005 6 -37 95 6703 29 -71 28837 -77 -17 15139 -14 31 23199 46 99 23871 -43 -91 247 4 75 -57 35800 66 -25 18198 -43 -16 37936 -1 -47 38088 ```	Case #1: 4 Case #2: 4 Case #3: impossible Case #4: 45518 Case #5: impossible Case #6: 37580 Case #7: 29640 Case #8: 44412 Case #9: 44866 Case #10: impossible Case #11: 20786

Problem C
Cat vs Dog

The latest reality show has hit the TV: "Cat vs. Dog". In this show, a bunch of cats and dogs compete for the very prestigious BEST PET EVER title. In each episode, the cats and dogs get to show themselves off, after which the viewers vote on which pets should stay and which should be forced to leave the show.

Each viewer gets to cast a vote on two things: one pet which should be kept on the show, and one pet which should be thrown out. Also, based on the universal fact that everyone is either a cat lover (i.e. a dog hater) or a dog lover (i.e. a cat hater), it has been decided that each vote must name exactly one cat and exactly one dog.

Ingenious as they are, the producers have decided to use an advancement procedure which guarantees that as many viewers as possible will continue watching the show: the pets that get to stay will be chosen so as to maximize the number of viewers who get both their opinions satisfied. Write a program to calculate this maximum number of viewers.

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 consists of a single line containing three integers c d v, where c is the number of cats, d is the number of dogs and v is the number of voters. v lines follow, each containing two pet identifiers. The first is the pet that this voter wants to keep, the second is the pet that this voter wants to throw out. A pet identifier starts with one of the characters "`C`" or "`D`", indicating whether the pet is a cat or dog, respectively. The remaining part of the identifier is an integer id giving the number of the pet. So for instance, "`D`42" indicates dog number 42.

Output

For each test case, print a line containing "Case #i: x" where i is its number, starting at 1 and x is the maximum possible number of satisfied voters for the show. Each line of the output should end with a line break.

Constraints

- $1 \leq t \leq 20$
- $1 \leq c, d \leq 100$
- $0 \leq v \leq 500$
- For pet identifier `C`id: $1 \leq id \leq c$
- For pet identifier `D`id: $1 \leq id \leq d$

Sample Input 1	Sample Output 1
2 1 1 2 C1 D1 D1 C1 1 2 4 C1 D1 C1 D1 C1 D2 D2 C1	Case #1: 1 Case #2: 3

This page is intentionally left blank.

Problem D
Charged Dust

Lea builds a physics experiment on her desks. She has some microscopic positively charged particles inside a flat square frame constraining their movements. Two charges q_1 and q_2 at a distance l are repelled by a force $\frac{q_1 q_2}{l^2}$ (as Lea uses CGS-Gaussian units, there are no constants involved). Lea first glues some charges inside the square or nearby, then adds a few more charges inside the square. She is interested in where the free charges might end up. There is a bit of friction so a configuration does not need perfect balance of electrostatic forces to remain stable.

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 three space-separated integers, n, m, and l, where n is the number of glued charges, m is the number of free charges inside the square, and l is the length of a side of the square. n lines with three space-separated integers u_j, v_j, c_j each follow, each line meaning that the j-th glued charge c_j is at the position (u_j, v_j). A line with n space-separated integers f_j follows, with f_j being the charge of the j-th free-moving particle inside the square.

The square has vertices $(0,0), (l,0), (l,l), (0,l)$.

Output

For each test case, output one line "Case #i:" with i being the number of the test case starting at 1, followed by n lines with two space-separated floating-point values x_j and y_j each. The output for each test case should be separated by a blank line.

Putting all charges f_j at their respective positions (x_j, y_j) should result in an equilibrium of forces for every free charge. That means that all forces on this charge add up to at most 10^{-6}. The forces affecting a free charge are:

- Being repelled by glued charges
- Being repelled by other free charges
- Reaction forces from the edges of the square (perpendicular to the corresponding edge) in case the particle pushes against a side (or two sides) of the square.

All the free-moving charges must lie inside the square. You are allowed to adopt tolerance of up to 10^{-6} for particle touching an edge.

Constraints

- $1 \leq t \leq 20$
- $0 \leq n \leq 50$
- $1 \leq m \leq 10$
- $1 \leq l \leq 10^6$
- $-10^9 \leq u_i, v_i \leq 10^9$ for $1 \leq i \leq n$
- $1 \leq c_i, f_j \leq 10^6$ for $1 \leq i \leq n$, $1 \leq j \leq m$

Sample Input 1	Sample Output 1
2 0 2 10 1 2 4 2 4 -5 2 99 2 -5 99 9 2 99 2 9 99 1 1	Case #1: 10 0 0 10 Case #2: 2.54217129919265 1.4578286047149183 1.4578287008073503 2.5421713952850817

Problem E
Hangar

Lea loves technology and follows the related news. She has heard that a huge hangar is being build to assemble a flying ship. As for some unclear reason the hangar is built in a hurricane-prone zone, its corners need special fortification! The company constantly changes the plans and the schedule, so there are many suitable fortified spots. Lea wonders, what is the largest area of a rectangular hangar with the corners in the already fortified spots.

Input

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

Each set starts with one integer n, where n is the number of corners in the particular set. n lines follow describing the coordinates of the prepared corners. Each line contains two integers x_i and y_i where x_i is the x-coordinate and y_i is the y-coordinate of the i-th cornerstone position.

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 the integer s being the largest possible area of a rectangle with the corners in the prepared points. If there is no such rectangle, s is 0.

Constraints

- $1 \leq t \leq 50$
- $1 \leq n \leq 600$
- $0 \leq x_i, y_i \leq 10000$ for $1 \leq i \leq n$
- Total number of points across the cases in a single input is at most 1500

Sample Input 1

```
3
1
1 1

4
0 1
1 0
2 1
1 2

6
0 1
1 0
2 1
1 2
0 20
2 20
```

Sample Output 1

```
Case #1: 0
Case #2: 2
Case #3: 38
```

This page is intentionally left blank.

Problem F
Smoking Gun

> “*Andy: Billy the Kid fired first*”
>
> “*Larry: No, I'm sure I heard the first shot coming from John!*”

The arguments went back and forth during the trial after the big shoot-down, somewhere in the old wild west. Miraculously, everybody had survived (although there were serious injuries), but nobody could agree about the exact sequence of shots that had been fired. It was known that everybody had fired at most one shot, but everything had happened very fast. Determining the precise order of the shots was important for assigning guilt and penalties.

But then the sheriff, Willy the Wise, interrupted: “Look, I've got a satellite image from the time of the shooting, showing exactly where everybody was located. As it turns out, Larry was located much closer to John than to Billy the Kid, while Andy was located just slightly closer to John than to Billy the Kid. Thus, because sound travels with a finite speed of 340 meters per second, Larry may have heard John's shot first, even if Billy the Kid fired first. But, although Andy was closer to John than to Billy the Kid, he heard Billy the Kid's shot first - so we know for a fact that Billy the Kid was the one who fired first!”

Your task is to write a program to deduce the exact sequence of shots fired in situations like the above.

Input

On the first line a positive integer: the number of test cases t. After that per test case:

- one line with two integers n and m: the number of people involved and the number of observations.
- n lines with a string S and two integers x and y: the unique identifier for a person and his/her position in Cartesian coordinates, in metres from the origin.
- m lines of the form “$S1$ heard $S2$ firing before $S3$”, where $S1$, $S2$ and $S3$ are identifiers among the people involved, and $S2 \neq S3$.

If a person was never mentioned as $S2$ or $S3$, then it can be assumed that this person never fired, and only acted as a witness. No two persons are located in the same position.

The test cases are constructed so that an error of less than 10^{-7} in one distance calculation will not affect the output.

Output

Per test case:

- One line containing “Case #i:” where i is its number, starting at 1, and the ordering of the shooters that is compatible with all of the observations, formatted as the identifiers separated by single spaces.

If multiple distinct orderings are possible, output “unknown” instead. If no ordering is compatible with the observations, output “impossible” instead.

Constraints

- $1 \leq t \leq 100$
- $2 \leq n \leq 100$
- $1 \leq m \leq 1000$
- S consists of up to 20 lower and upper case letters.
- $0 \leq x, y \leq 1000000$

Sample Input 1	Sample Output 1
`3` `4 2` `BillyTheKid 0 0` `Andy 10 0` `John 19 0` `Larry 20 0` `Andy heard BillyTheKid firing before John` `Larry heard John firing before BillyTheKid` `2 2` `Andy 0 0` `Beate 0 1` `Andy heard Beate firing before Andy` `Beate heard Andy firing before Beate` `3 1` `Andy 0 0` `Beate 0 1` `Charles 1 3` `Beate heard Andy firing before Charles`	`Case #1: BillyTheKid John` `Case #2: impossible` `Case #3: unknown`

Problem G
Treasure Hunt Revisited

As you might remember, not long ago, Lea went on a Treasure Hunt, following a trail of hints during a university event. She soon realized that Prof. Escher must have taken that challenge very seriously when she found that the hints led her to a nearby harbour and from there into the wide ocean. Determined to find what was hidden, she gathered her fellow students and became the captain of the "White Pearl", leading them further along the trail on the map. After a while and a few hard trials (wild storms, mean pirates and scurvy scurvy) they finally found a secluded island somewhere in the sea. Still following the map, they dug up a metal chest buried under an idol of some long forgotten god.

Alas, when Lea (in her position as captain and navigator) tried to open the chest, she found it was locked with a combination lock. Luckily the map had a few hints on what the combination was:

- The solution you seek is called x.
- x is a prime number.
- x is larger than k.
- x has as many digits as k.
- Either all permutations of x are prime numbers or there are more than $log_{10}(x)$ permutations[1] of x that are prime numbers.
- x is the smallest number such that all these conditions hold.
- There will always exist at least one solution such that all conditions hold.

Can you help Lea and her fellow students figure out the solution and finally get to the prize their professor left them?

Input

The first line of the input contains an integer t. t test cases follow, each on a separate line. Each test case consists of a single integer k as explained above.

Output

For each test case, output one line containing "Case #i: x" where i is its number, starting at 1, and x is the number Lea can use to open the treasure chest. Each line of the output should end with a line break.

Constraints

- $1 \leq t \leq 20$
- $1 \leq k \leq 10^8 - 1$

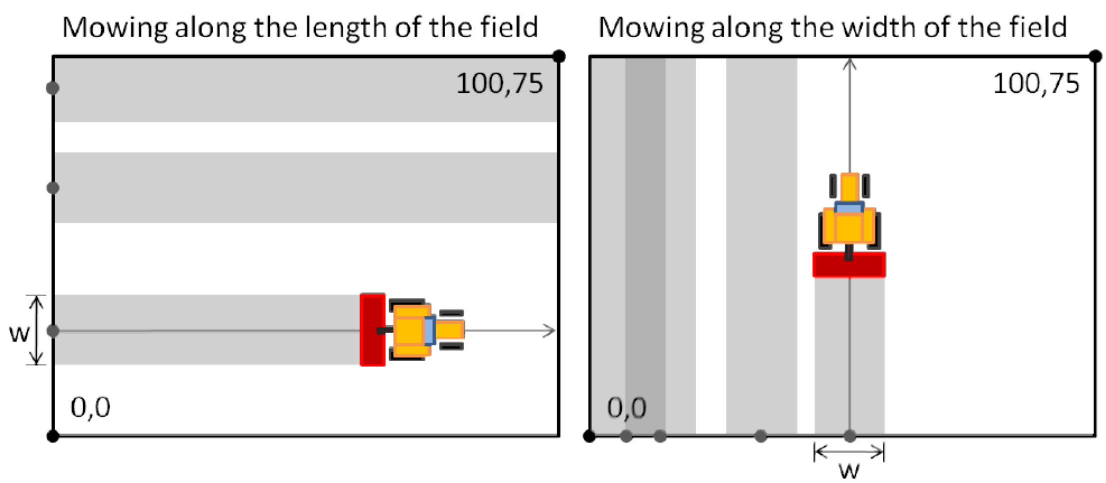
Sample Input 1	Sample Output 1
5 10 870 919 80000 120	Case #1: 11 Case #2: 907 Case #3: 937 Case #4: 80021 Case #5: 131

[1] permutations are counted uniquely, i.e. 112 has 3 permutations

This page is intentionally left blank.

Problem H
Lawnmower

The International Collegiate Soccer[2] Competition (ICSC) is famous for its well-kept rectangular stadiums. The grass playing fields in ICSC stadiums are always 100 meters long, and 75 meters wide. The grass is mowed every week with special lawn mowers, always using the same strategy: first, they make a series of passes along the length of the field, and then they do the same along the width of the field. All passes are straight lines, parallel to the sides of the field.



The ICSC has hired a new lawn-mower, Guido. Guido is very chaotic, and instead of covering the field incrementally, he likes to choose random starting positions for each of his passes. But he is afraid of not doing a good job and being fired by the ICSC, so he has asked you to help him. Write a program to make sure that the grass in the field is perfectly cut: all parts of the field have to be mowed at least once when the mower goes from end to end, and again when the mower goes from side to side.

Input

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

Each test case begins with a line containing two integers n_x and n_y and a real number w, where w describes the width of that particular lawnmower. The next line contains n_x real numbers x_i, describing the starting positions of the mower's center in Guido's end-to-end passes. The last line contains n_y real numbers y_j describing the starting positions in the side-to-side passes.

Any cut will also include its boundaries. For example, if a 2.0-meter wide cut is performed along the 10.0-meter mark, then a strip of grass from 9.0 to 11.0 (including both) will be considered "cut".

Output

For each test case, output one line containing "Case #i: x" where i is its number, starting at 1, and x is either "YES" if Guido has done a good job, or "NO" if some part of the field has not been mowed at least once when the mower was travelling along the length of the field, and again when it was travelling along the width.

Constraints

- $1 \leq t \leq 50$

[2]The ICSC is sponsored by the Association for Sports Machinery (ASM), which started out in the US, so they prefer to use the term "soccer" instead of "football".

- $1 \leq n_x < 1000$
- $1 \leq n_y < 1000$
- $0 < w \leq 50$
- $0 \leq x_i \leq 75$ for all $1 \leq i \leq n_x$
- $0 \leq y_j \leq 100$ for all $1 \leq j \leq n_y$

Sample Input 1

```
4
8 11 10.0
0.0 10.0 20.0 30.0 40.0 50.0 60.0 70.0
0.0 10.0 20.0 30.0 40.0 50.0 60.0 70.0 80.0 90.0 100.0

8 10 10.0
0.0 10.0 20.0 30.0 40.0 50.0 60.0 70.0
0.0 10.0 30.0 40.0 50.0 60.0 70.0 80.0 90.0 100.0

4 5 20.0
70.0 10.0 30.0 50.0
30.0 10.0 90.0 50.0 70.0

4 5 20.0
60.0 10.0 30.0 50.0
30.0 10.0 90.0 50.0 70.0
```

Sample Output 1

```
Case #1: YES
Case #2: NO
Case #3: YES
Case #4: NO
```

Problem I
Forbidden Words

Mark and Polly are computing[3] Hausdorff dimensions[4] of complicated objects. We do not ask you to learn all about that, but this computation depends on a question about words.

We consider some alphabet size $2 \leq d \leq 9$ and words composed of digits from 1 to d. For a fixed set of forbidden words F, words u and v are called *incompatible* if their concatenation contains some word from F, i.e. $uv = afb, f \in F$. Otherwise words u and v are compatible. Two words u and u' of length l are *similar*, if for every word v of the same length l, u and u' are either both compatible with v or both incompatible with v.

Given the alphabet size, the set of forbidden words, and the length l, count how many classes of similar words there are.

Input

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

Each test case starts with a single line containing three space separated integers d, n, and l, where d is the size of the alphabet, n is the number of forbidden words, and l is is the length of the words where we count the similarity classes. n lines follow, each containing a forbidden word (an integer of 1 to $2l$ digits between 1 and d).

Output

For each test case, print a line containing "Case #i: x" where i is the test case number, starting at 1, and x is the number of similarity classes.

Constraints

- $1 \leq t \leq 10$
- $2 \leq d \leq 9$
- $2 \leq l \leq 20$
- $d^l < 10^7$
- $2 \leq n \leq 35$
- $1 \leq x \leq 1000$

Sample Input 1

```
2
2 1 1
11

3 4 3
131
313
2312
2132
```

Sample Output 1

```
Case #1: 2
Case #2: 10
```

[3] https://arxiv.org/abs/2012.07083
[4] https://en.wikipedia.org/wiki/Hausdorffdimension

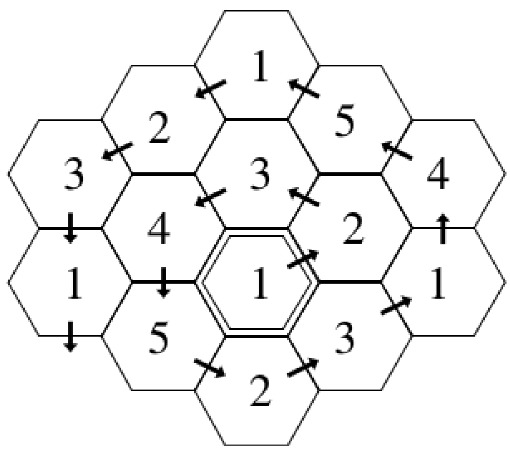
This page is intentionally left blank.

Problem J
Settlers of Catan

The popular board game "Settlers of Catan" starts by creating a random board. This board consists of hexagonal resource tiles containing five different resources: clay, lumber, wool, grain, and ore. For simplicity, we will denote these by the numbers 1 to 5.
Random boards, however, often have multiple equal resource tiles next to each other. This annoys some players. Therefore, we have invented a new way of creating the playing board. Starting in the middle and spiraling outwards, each time we add a new tile to the board we choose the resource of the tile according to the following rules:

- the new tile must be different from its neighboring tiles on the board so far;
- in case multiple tiles are possible, we choose a resource that occurs the least number of times on the board so far;
- in case multiple tiles are still possible, the new resource must have the lowest number possible.

The figure underneath shows how to spiral outwards and which resource tiles are chosen first. We are curious what the number of the resource is on the n-th tile that is added to the board (starting with $n = 1$).



Input

The first line of the input contains an integer t. t test cases follow.
Each test case consists of a single line with one integer n, the number of the tile we are curious about.

Output

For each test case, output one line containing "Case #i: x" where i is its number, starting at 1, and x is the resource of the n-th tile.

Constraints

- $1 \leq t \leq 20$
- $1 \leq n \leq 10^4$

Sample Input 1 | **Sample Output 1**

Sample Input 1	Sample Output 1
4 1 4 10 100	Case #1: 1 Case #2: 4 Case #3: 5 Case #4: 5