Algorithms for Programming Contests SS16 - Week 13

Chair for Efficient Algorithms (LEA), TU München

Moritz Fuchs, Philipp Hoffmann, Christian Müller, Chris Pinkau, Stefan Toman

This problem set is due by

Wednesday, 13.07.2016, 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, 13.07.2016, at 14:15 p.m. in room MI 00.13.009A. Happy solving!

Eight points are awarded for each problem, but only four of them will count towards the total number of 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
Snake

Some day, Lea got overwhelmed with nostalgia. She rummaged through one of her old cabinets until she found her old mobile phone - a Nokia 3310. Then she sat in a corner for several hours to enjoy the simplicity of Snake.

Figure A.1: Snake on a Nokia 3310[1]

Here, Lea controls a little snake that makes her way across a rectangular grid to munch on some little bits of food scattered around the playing field. If the head of the snake moves over a location with a bit of food, it grows in size by 1, i.e. the head moves into the location where the food is, but the tail does not move. If the snake moves outside of the grid, it reappears on the other side. Lea loses the game whenever the snake runs into some part of itself. However, the game seems to have changed a little from when she grew up (she asks herself what her mobile has been up to in that cabinet). Nowadays, all the little bits of food appear at the beginning of the game so Lea can now plan ahead where to go.

Can you tell her how her planned gameplay will work 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 a line containing two space-separated integers n and m, where n is the size of the playing field (an $n \times n$ grid) and m is the number of blocks of food bits. One line follows containing two integers c_s, r_s, the starting position of Leas snake. The snake starts in column c_s and row r_s with length 1 and always faces to the right. m lines follow, detailing the placement of the bits of food: The i-th line contains four space-separated integers c_i, r_i, w_i and h_i. This describes a rectangular block of food, with one bit of food at every grid location starting from column c_i and row r_i and extending w blocks to the right and h blocks down. If the starting point (c_s, r_s) is contained in this block, it will not contain food. A grid location can only ever contain one single bit of food, even if some of the m

[1]Image source: https://apkpure.com/classic-snake-1997/com.byq.nokia_snake

rectangular blocks overlap.

One line follows containing an integer l and a string s with l characters, describing Leas planned gameplay. The string contains one letter for every timestep with “F” being a step forward, “R” means the snake turns to the right, then steps forward and “L” means the snake turns to the left and then steps forward (all relative to the direction the snake is facing).

Output

For each test case, print a line containing “Case #i: $steps$ $points$” where i is its number, starting at 1, $steps$ is the amount of steps of Leas plan that can be executed successfully and $points$ is the amount of bits of food the snake will munch on until $steps$ steps have been reached. Each line of the output should end with a line break.

Constraints

- $1 \leq t \leq 20$
- $1 \leq n \leq 10^4$
- $0 \leq m \leq 100$
- $1 \leq l \leq 10^4$
- $1 \leq c_s, r_s \leq n$
- $1 \leq c_i \leq c_i + w_i - 1 \leq n$ for all $1 \leq i \leq m$
- $1 \leq r_i \leq r_i + h_i - 1 \leq n$ for all $1 \leq i \leq m$
- s contains only the letters “F”, “R” and “L” and is of length l.

Sample explanation

For the first sample, in the first case, the grid is 4 by 4 with a bit of food in the upper left and lower right corner. Lea moves the snake to the right, then upwards for two steps, another step to the right (into the upper right corner) and then down to the lower right corner where the snake munches on the bit of food.

For the second case the grid is 10 by 10. The snake starts in $(1, 1)$ and makes its way to the lower right while picking up the two bits of food along the way.

In the third case the snake goes to the right, picking up all 6 bits of food, then turns around and hits itself at the 10-th step.

In the last case, the snake goes straight through the grid twice, picking up all 4 bits of food along the way, then turns around and hits itself. In this case the snake traverses several walls before it hits itself.

Sample Input 1

```
4
4 2
2 3
1 1 1 1
4 4 1 1
7 FLFRRFF

10 2
1 1
5 5 1 1
6 6 1 1
12 FFFFRFFFFLFF

10 1
4 5
5 5 6 1
13 FFFFFFLLFLRFF

5 1
1 2
1 2 5 1
15 FFFFFFFFFFLLLRF
```

Sample Output 1

```
Case #1: 7 1
Case #2: 12 2
Case #3: 9 6
Case #4: 12 4
```

Sample Input 2

```
5
5 3
5 2
1 4 1 2
2 3 1 1
4 4 1 1
7 FFRFFFL

9 1
6 8
3 5 4 4
10 FFFFFFRRFF

10 3
6 1
9 8 2 1
9 1 2 1
1 1 1 1
9 RFFFLFFLL

4 1
1 1
1 1 4 4
8 LRFFFFLF

9 2
8 3
6 5 2 1
3 2 4 3
8 FFFFFFRF
```

Sample Output 2

```
Case #1: 7 1
Case #2: 10 1
Case #3: 9 0
Case #4: 4 4
Case #5: 8 4
```

Problem B
Absurdistanian Calendar

One of the first things Lea learnt when she first visited Absurdistan was their unusual approach to using calendars. They decided that the weekend is the best part of the week, so they extended it to three days with a new "Chillday" between Saturday and Sunday. This day is used to recover between the parties on the other days of the weekend. Thus, an absurdistanian week has eight days.

Also, they are very superstitious. The Absurdistanians believe that you have to be very careful to not cause any accident on every thirteenth of the month, no matter whether it is a Friday or any other day of the week. Skipping this day of the month is of no use since this will only result in more disasters. After this bad day there will probably not be any good day this month anymore, so they just end each month after the thirteenth.

Lea is very confused by this local calendar and while thinking about it she forgot the dates of her friend's birthdays. She knows how many days are still left until the birthdays and needs to know on which day of the month each birthday occurs. The month itself is not important, she just wants to know for all birthdays whether it will be on a thirteenth of a month. To make things a little easier Lea wrote the numbers of days until the birthdays in base 8 to represent a full week by 10 for instance. So she sits down on this Chillday which happens to be the third of the month and asks for your help. Can you solve the problem?

Input

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

Each test case consists of a single line containing the number x of days to pass until the birthday of one of Lea's friends in base 8.

Output

For each test case, output one line containing "Case #i: y" where i is its number, starting at 1, and y is the number of the birthday during its month. Print these days as usual, namely in base 10. Each line of the output should end with a line break.

Constraints

- $1 \leq t \leq 20$
- $1 \leq x \leq 8^{5 \cdot 10^7}$

Sample Explanation

In the first sample Lea wants to know about a day 17_8 days in the future, that is 15_{10} days base 10. Today is the third, so it takes 10 more days to finish the month. Therefore, the birthday in question is the fifth of its month.

Sample Input 1	Sample Output 1
5 17 12 13 1 12345	Case #1: 5 Case #2: 13 Case #3: 1 Case #4: 4 Case #5: 9

Sample Input 2

```
20
62025701
732005
6522000
6505013
3573353
43537
4510425464
442557432
552260
65672
10070270
166255113
3055763
6101546
1543116
300610251
6612345631
4561615336
610624
303335476
```

Sample Output 2

```
Case #1: 13
Case #2: 12
Case #3: 10
Case #4: 8
Case #5: 11
Case #6: 9
Case #7: 9
Case #8: 10
Case #9: 13
Case #10: 8
Case #11: 4
Case #12: 6
Case #13: 9
Case #14: 4
Case #15: 9
Case #16: 3
Case #17: 2
Case #18: 1
Case #19: 1
Case #20: 9
```

Problem C
Printing

Lea is not very fond of reading on her monitor, she prefers printed documents. However, madness has befallen all the printers! If you send them a print job, the do not print the whole job but randomly choose just one page that they print. Thus Lea has to print the same document multiple times until she can gather all the pages she needs.

Lea has already sent a number of print jobs to the different printers. Can you tell her if it is possible that she will find every page of the document she wanted to print at least once?

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 two integers n, the number of printers, and m, the number of pages in Lea's document. The printers and pages are numbered from 1 to n or m, respectively. n lines follow describing the print jobs Lea has sent: The i-th line contains a non-empty string with the numbers of the pages printer i was sent. The numbers are comma-separated and may be given as sections where the first and last pages are separated by a dash. For instance the string "1,10,3,5-8" represents pages 1, 3, 5, 6, 7, 8 and 10.

Output

For each test case, output one line containing "Case #i: x" where i is its number, starting at 1, and x is "yes" if it is possible that every page of the document has been printed already, "no" otherwise. Each line of the output should end with a line break.

Constraints

- $1 \leq t \leq 20$
- $1 \leq n \leq 250$
- $1 \leq m \leq 250$
- No page will be mentioned several times per line of input, in particular sections will not overlap.
- Sections of pages will always be given with the smaller index first.

Sample Input 1

```
5
3 3
1
1-2
2,3

2 2
1
1

2 5
1-4
2-4

2 1
1
1

3 5
3-5
3-5
1
```

Sample Output 1

```
Case #1: yes
Case #2: no
Case #3: no
Case #4: yes
Case #5: no
```

Sample Input 2

```
9
4 5
1,5,2-3
3-5
4-5
2-3

3 5
1,2,4-5
3-5
1,2-3

1 3
2

3 5
5,2-3,1
1-5
1-4

3 1
1
1
1

1 3
2,3

3 3
2-3
1
2-3,1

4 4
1-2
1-2
3-4,2
1-4

1 3
2-3,1
```

Sample Output 2

```
Case #1: no
Case #2: no
Case #3: no
Case #4: no
Case #5: yes
Case #6: no
Case #7: yes
Case #8: yes
Case #9: no
```

This page is intentionally left blank.

Problem D
Surveillance

Burglars are around! Lea's aunt fears that they will get to her house next and asks Lea to stay with her all night to defend her belongings. Lea has naturally better plans for tonight, so she has a another proposal. Lea will buy and install some surveillance cameras together with her aunt.

The cameras they buy are cutting-edge technology: They scan their environment using quantum technology and can detect any intruder with ease. To do so, they send out some quantum waves that are not blocked by anything: buildings, plants, not even by the great aquarium Lea's aunt installed in her bathroom. Unfortunately, the waves disturb each other, so no two cameras may watch the same area of the property.

The area a camera watches is a perfect circle around the point where the camera is installed. The software to control the cameras allows for only one size of the surveillance radius for all cameras. This is a huge restriction, so Lea spent some time hacking the software. She got bored with the awful code of the software, so she just implemented a way to set the radius of the first camera individually while the other cameras still need to have the same surveillance radius. A radius can never be set to 0 to switch the camera off but can be as small or large as required.

Lea wants to set the radii in a way that all but the first one are the same, no surveillance area overlaps with another and the total area seen by the cameras is maximal. Can you help her compute this area?

Input

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

Each test case starts with a line containing an integer n, the number of cameras. n lines follow describing the cameras. The i-th of them contains two space-separated real numbers x_i and y_i, the coordinates of the i-th camera. The radius of the first camera can be chosen arbitrarily, the radii of all other cameras have to be the same.

Output

For each test case, output one line containing "Case #i: a" where i is its number, starting at 1, and a is the maximum area that can be watched by the cameras. Each line of the output should end with a line break.

Your output must have an absolute or relative error of at most 10^{-6}.

Constraints

- $1 \leq t \leq 20$
- $2 \leq n \leq 10^4$
- $-100.0 \leq x_i, y_i \leq 100.0$ for all $1 \leq i \leq n$

Lea's aunt's property is very big, so we can assume it to be an infinite two-dimensional plane.

Sample Input 1

```
3
7
0.0 0.0
-2.0 0.0
2.0 0.0
-1.0 1.732
1.0 1.732
-1.0 -1.732
1.0 -1.732

3
-1.0 0.0
0.0 0.0
1.0 0.0

2
0.0 0.0
1.0 0.0
```

Sample Output 1

```
Case #1: 21.99018096
Case #2: 3.14159265
Case #3: 3.14159265
```

Sample Input 2

```
6
3
85.89749 8.041092
13.502266 80.74486
1.2792587 28.864044

2
10.871872 98.02222
-10.477295 -23.74871

5
6.6840286 99.15872
63.19284 37.411926
-99.12375 6.2733994
13.110519 -79.276566
51.04373 -90.824745

5
41.790314 22.764038
69.2648 -50.786854
0.49011993 -74.49112
5.434021 75.42493
18.820465 -99.197754

4
96.94888 -62.5437
-52.94235 -28.406075
-36.57235 4.2853622
27.117096 -20.070099

5
-76.06155 -0.68807983
12.408653 -34.555946
-44.162727 -99.990585
79.7007 -27.338135
-56.049953 22.013031
```

Sample Output 2

```
Case #1: 23856.75320794
Case #2: 48015.93349141
Case #3: 22009.72272370
Case #4: 12864.66353914
Case #5: 20987.36209114
Case #6: 11508.33809639
```

Problem E
Pathing

Lea is programming the next new hit game, Age of Conquercraft, a real time strategy game. Unfortunately, her units do not run as they are supposed to: Instead of taking the shortest path to the their target, they sometimes take a longer path or run into a dead end and have to walk back. Can you help her and write a better pathfinding algorithm?

The terrain that Lea uses is divided into squares of 1x1 cm which are either passable or impassable. The units are considerd to be a point (they do not occupy space) and can move in any direction (in particular, they do not have to move perpendicular to the grid) but cannot move into impassable squares.

Input

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

Each test case starts with a description of the grid. The first line contains three integers w h n, where w and h are the width and height of the grid and n is the number of impassable locations. All following coordinates are 1-bases with $(1, 1)$ being the upper left corner.

n lines follow, describing the impassable locations. The i-th line contains four integers x_i y_i w_i h_i, describing an impassable square starting at grid position (x_i, y_i) of width w_i and height h_i in the grid, extending to the right and downward. The grid description is followed by a blank line.

Two lines follow, the j-th of which contains two integers a_j b_j, the first is the start point (a_1, b_1) of the unit and the next is the target point (a_2, b_2).

Output

For each test case, print a line containing "Case #i: c" where i is its number, starting at 1, and c is a space separated list of coordinates $(X_1, Y_1)(X_2, Y_2) \dots (X_k, Y_k)$ (with brackets and comma) such that

- $(X_1, Y_1) = (a_1, b_1)$ is the start point,
- $(X_k, Y_k) = (a_2, b_2)$ is the target point,
- for every $1 \le i < k$ it is possible to move from (X_i, Y_i) to (X_{i+1}, Y_{i+1}) in a straight line without entering an impassable grid location, and
- the length of the path along the points in c is minimal.

Any optimal solution (up to relative or absolute errors of at most 10^{-4}) will get accepted. Each line of the output should end with a line break.

Constraints

- $1 \le t \le 20$
- $5 \le w, h \le 5000$
- $0 \le n \le 100$
- $1 \le x_i < x_i + w_i \le w$ for all $1 \le i \le n$.
- $1 \le y_i < y_i + h_i \le h$ for all $1 \le i \le n$.
- $1 \le a_j \le w$ for all $1 \le j \le 2$.
- $1 \le b_j \le h$ for all $1 \le j \le 2$.

- The start point and the target points will neither touch nor be contained in an impassable location.
- The impassable locations will not touch or overlap with each other or the border.

Sample explanation

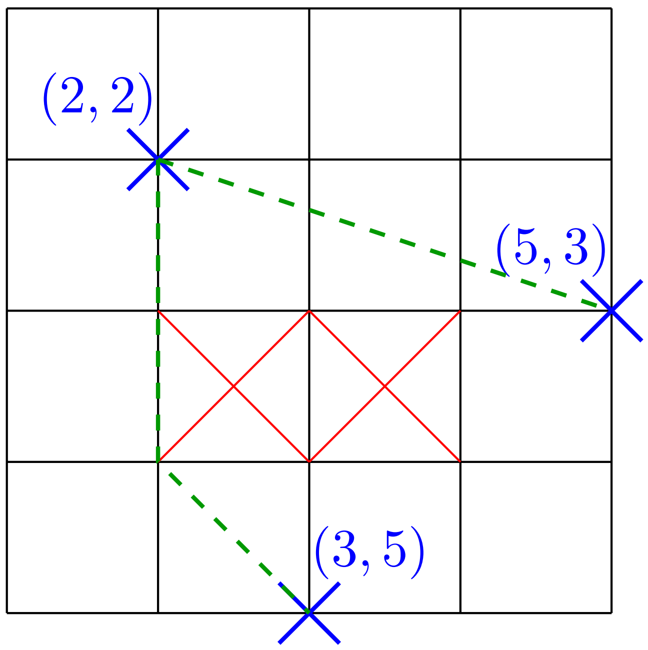

Figure E.1: Illustration of the first sample input

In the first sample, depicted in Figure E.1, in the first case it is possible to move from the starting point to the target in a straight line. In the second case, this is not possible because of the impassable locations, therefore we move via the point $(2, 4)$. Another accepted solution would be $(2, 2)$ $(2, 3)$ $(2, 4)$ $(3, 5)$. The solution $(2, 2)$ $(4, 3)$ $(4, 4)$ $(3, 5)$ would be wrong because the path is longer then the optimal one.

Sample Input 1

```
2
6 6 1
2 3 2 1
2 2
5 3

6 6 1
2 3 2 1
2 2
3 5
```

Sample Output 1

```
Case #1: (2,2) (5,3)
Case #2: (2,2) (2,4) (3,5)
```

Sample Input 2

```
4
7 6 2
2 2 2 1
3 4 3 1
3 1
4 6

7 7 3
2 3 2 1
3 5 3 1
5 2 1 2
3 2
5 7

7 7 2
4 1 2 2
3 2 1 3
2 1
3 1

7 8 2
3 4 4 1
1 6 6 1
4 8
1 8
```

Sample Output 2

```
Case #1: (3,1) (2,2) (2,3) (3,5) (4,6)
Case #2: (3,2) (4,3) (5,4) (6,5) (6,6) (5,7)
Case #3: (2,1) (3,1)
Case #4: (4,8) (1,8)
```

This page is intentionally left blank.

Problem F
Soccer Lineup

Tactics are hard. During the recent EM soccer tournament, Lea met "Ragnar O. S. Fußballsson", the coach of the small national team of Fireland. Fireland is a small island somewhere in the ocean with fewer inhabitants than sheep. Thus, it is hard to establish a very good national soccer team. Nevertheless, its inhabitants are very proud and always fight back against the bigger countries and its coach naturally comes up with brilliant tactical plans on how to beat the other team.

Lea got to talk to him for a while and he elaborated on his tactics: Famous soccer tactics include the $3-4-3$ (3 defenders, 4 midfielders and 3 strikers) or the $4-3-3$ (4 defenders, 3 midfielders and 3 strikers) or some other elaborate placement of team members.

The players also wear numbered shirts with numbers from 1 to 11 and for every firelandic inhabitant it is perfectly clear that two players can never play in the same position if the sum of the numbers of their shirts is 13. Otherwise the team is doomed and will surely lose. Thus, valid soccer lineup assigns a position to every player, has exactly one goalkeeper and the shirt numbers of no two players in the same position sum up to 13. While Ragnar rambles on, Lea cannot help but marvel how many different soccer tactics there could be and began dreaming about the infamous $6-2-2$ that, in her imagination, would beat all the agressive teams. Can you tell her how many possible soccer lineups there are for a given team?

Input

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

Each test case consists of exactly 11 lines, each describing a single player: The i-th line describes the player with shirt number i (numbered from 1 to 11) and contains a string s_i that describes the possible positions player i can play. s_i then contains a character c if player i can play on position c. Position characters are either G (Goalkeeper), D (Defender), M (Midfielder) or S (Striker).

Output

For each test case, print a line containing "Case #i: x" where i is its number, starting at 1 and x is the amount of valid soccer lineups. Each line of the output should end with a line break.

Constraints

- $1 \leq t \leq 20$
- s_i consists only of the characters "G", "D", "M", "S".
- s_i contains every character at most once.
- s_i contains at least one character.

Sample Input 1

```
3
G
D
D
D
D
D
M
M
S
S
S

GS
D
D
D
D
M
M
M
MS
S
S

G
D
D
D
D
M
MS
MS
MS
MS
S
```

Sample Output 1

```
Case #1: 1
Case #2: 0
Case #3: 8
```

Sample Input 2

```
5
GM
DM
DSG
GSM
S
S
SGD
GM
D
D
DGM

SD
MSD
MD
DG
D
SGD
S
GS
MSD
GSM
DGS

GS
D
SDG
SD
SD
MD
M
GS
DM
MS
S

DS
DSG
MSD
GS
SDM
MG
MGD
SDM
MG
DM
M

D
MDS
SMD
GDM
G
SDG
MGS
SGD
GDS
SD
S
```

Sample Output 2

```
Case #1: 24
Case #2: 120
Case #3: 24
Case #4: 240
Case #5: 0
```

Problem G
Rallye

Lea participates in this year's "Rallye Absurdistan". There is a challenging track through cities, forests and deserts and Lea wants to finish as fast as possible.

The rallye is a little extraordinary since the participants are allowed to switch their vehicles at certain checkpoints. Each driver may bring a rallye car and a motocross bike to switch between depending on the track. The rallye lasts some days and you may switch the vehicle without time loss between days. Each day has a certain number of checkpoints (the same for each day of the rallye) and to switch the vehicle you need some minutes. For logistic reasons you are only allowed to switch the vehicle at one checkpoint per day, but you are allowed to switch every night no matter whether you switched at one of the checkpoints that day.

Lea tested the complete track and knows all times she needs with each vehicle. What is the best time she could need to finish the rallye?

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 the three space-separated integers d, c and m where d is the number of days the rallye lasts, c is the number of checkpoints per day, and m is the number of minutes Lea needs to switch vehicles. One lines follows containing $d \cdot (c+1)$ integers describing the times Lea needs with her rallye car between the start/end points of each day and the checkpoints. Another line follows containing $d \cdot (c+1)$ integers describing the times Lea needs with her motocross bike between the start/end points of each day and the checkpoints.

Output

For each test case, output one line containing "Case #i: x" where i is its number, starting at 1, and x is the optimal time Lea needs to finish the rallye. Each line of the output should end with a line break.

Constraints

- $1 \leq t \leq 20$
- $1 \leq d \leq 10^4$
- $1 \leq c \leq 10$
- m and all other times are between 1 and 10^3 inclusively.

Sample Explanation

In the first sample the rallye has 2 days with 1 checkpoint each. To switch vehicles Lea needs 10 minutes. On the first day it is best to stick to the rallye car which takes $90 + 90 = 180$ minutes. On the second day it is best to start with the motocross bike and take 61 minute to the first checkpoint. Then switch to the rallye car and arrive at the finish line after 43 more minutes. This takes $61 + 10 + 43 = 114$ minutes on the second day which is $180 + 114 = 294$ minutes in total.

In the second sample it is better to stick to the motocross bike on the second day resulting in a total time of 293 minutes.

Sample Input 1

```
2
2 1 10
90 90 75 43
95 98 61 67

2 1 10
90 90 75 43
95 98 61 52
```

Sample Output 1

```
Case #1: 294
Case #2: 293
```

Sample Input 2

```
7
2 2 71
31 83 97 86 96 17
95 28 97 95 20 76

2 1 69
98 71 40 74
75 94 36 62

3 1 24
25 92 64 79 39 48
99 78 67 76 7 15

2 2 34
74 96 93 37 37 82
22 92 43 58 3 21

2 2 49
33 91 37 4 75 37
8 80 35 7 1 68

2 1 1
85 4 62 21
44 78 57 60

2 2 6
31 30 80 41 84 31
30 31 85 47 71 16
```

Sample Output 2

```
Case #1: 402
Case #2: 267
Case #3: 282
Case #4: 239
Case #5: 199
Case #6: 128
Case #7: 275
```

Problem H
Water Temple

As you know by now, Lea has always been a great adventurer, always at the ready to explore the unknown. Again, she booked a trip to Templonia, to find another forgotten temple with relics of the past. After months of searching, she found the legendary "Water Temple" - a temple to honor the water spirits, built by a long forgotten people. As she explores the entrance levels, she notices that the temple continues for many levels below ground. Luckily, she even found a map that shows her how many rooms there are and how to get to each one.

There is just a slight problem - the lower levels are all filled with water. The ancient priests must have tried to preserve the water spirits holy sanctum. So to explore all of the rooms, Lea has to drain some of the water first. Otherwise, she would not get to see the marvelous artifacts hidden in the deeper layers of the temple.

Upon closer inspection of the map, she found that several of the rooms are marked as "control rooms" that can be used to drain some of the water out of the temple. They even carry small numbers that tell Lea just how much water can be drained at this specific room. She also found out the lowest point of every hallway, so she knows how much water to drain until she can go through the hallways and enter the room they lead to. Since the temple seems to be connected by some elaborate pipe system, the water inside the temple has the same level everywhere.

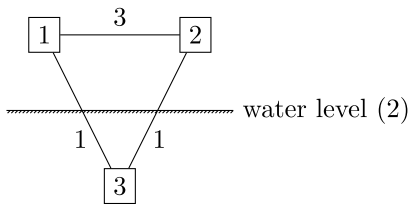

Figure H.1: Sample 1, case 2

Of course Lea wants to explore all the fascinating rooms in the temple, so can you tell her if she can lower the water level enough so that every room is reachable?

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 four space-separated integers n, m, k and l, where n is the amount of rooms in the water temple, numbered from 1 to n, and m is the amount of hallways connecting the rooms, k is the amount of rooms that are control rooms and l is the initial water level. m lines follow describing the hallways: the i-th line contains three space-separated integers a_i, b_i and l_i, describing a hallway from room a_i to b_i. To use this hallway, the water level must have been drained to at least water level l_i. You can assume that a room is drained of water as soon as there is at least one hallway to it that is usable. k lines follow describing the control rooms: the i-th line contains two space-separated integers a_i, d_i specifying that room a_i is a control room. If Lea reaches room a_i and operates the machinery there, she can drain the water throughout the whole temple to any level between the current water level and d_i (inclusively).

Output

For each test case, print a line containing "Case #i: r" where i is its number, starting at 1, and r is the maximum water level that can be left inside the temple so that every room is reachable from the entrance (room 1). Print "Case #i: `impossible`" if Lea cannot lower the water level enough so that every room is reachable. Each line of the output should end with a line break.

Sample explanation

In the first sample, in the first case, the initial water level is 2. This means Lea can use the hallways from room 1 to 2 and from 2 to 3, but not the hallway from 1 to 3 (the water level would have to be lowered to 1 to use it). There are no control rooms, but luckily, Lea can already reach all rooms, so the answer is the inital water level.

In the second case, Lea can only use the hallway from 1 to 2, but has no way to lower the water level any further. Thus she cannot explore the temple fully.

In the third case, Lea can use the hallway from 1 to 2. There, she can lower the water level to 0. However, she only lowers it to 1. Then she can go back to the entrance and use the hallway from 1 to 3 to reach the final room.

Constraints

- $1 \leq t \leq 20$
- $1 \leq n \leq 10^4$
- $0 \leq m \leq 10^6$
- $0 \leq k \leq n$
- $0 \leq l \leq 10^4$
- $1 \leq a_i, b_i \leq n$ for all $1 \leq i \leq n$
- $0 \leq d_i \leq 10^4$
- The temple is connected, i.e. there is a water level so that every room is reachable from the entrance.
- The controls rooms given are unique, i.e. every room is mentioned at most once.

Sample Input 1

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

3 3 0 2
1 2 3
1 3 1
2 3 1

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

Sample Output 1

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

Problem I
Inheritance

Lea might die soon. Or not. However, she wants to make sure that in the unlikely event of her death, there are no quarrels over her inheritance. She has named three people her heirs and her inheritance must be equally shared between them. Furthermore, those direct heirs of Lea have 1, 2 and 3 heirs respectively and should one of the direct heirs die, of course his part of the inheritance must again be equally shared between those heirs. Tell Lea whether for her current amount of money this is certainly possible.

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 amount of money Lea has.

Output

For each test case, print a line containing "Case #i: x" where i is its number, starting at 1, and x is either "yes" if sharing the money equally is possible, or "no" if not. Each line of the output should end with a line break.

Constraints

- $1 \leq t \leq 100$
- $1 \leq n \leq 10^{500}$

Sample Input 1

```
3
12
36
252
```

Sample Output 1

```
Case #1: no
Case #2: yes
Case #3: yes
```

Sample Input 2

```
5
164628
69087
626332
57728
183268643921177320438594398540379208168945539190399241953577295561749327474553690536
```

Sample Output 2

```
Case #1: yes
Case #2: no
Case #3: no
Case #4: no
Case #5: yes
```