Technische Universität München
Fakultät für Informatik
Lehrstuhl für Effiziente Algorithmen (LEA)
Prof. Dr. Harald Räcke
Moritz Fuchs, Philipp Hoffmann, Christian Müller
Chris Pinkau, Stefan Toman

Winter Term 2015/16
Problem Set 6
18.11.2015

Algorithms for Programming Contests

This problem set is due by

Wednesday, 25.11.2015, 6:00 a.m.

This problem set differs from the usual one as there are seven problems, but only five of them are solvable in a few seconds of computation time. One of your tasks is therefore to figure out which problems to solve. The scoreboard will be randomized this week to avoid revealing the solvable problems. Submit your solutions at

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

This week's problems are:

Six points will be awarded for each problem solved. Additionally, you will get six points for each problem you identified correctly as impossible. Make submissions for **exactly** five problems to claim this bonus! If you submit for more or less problems you will not get any bonus points. You do not need to actually solve five problems, just submit something. These bonus points are awarded for figuring out which problems are solvable and which problems cannot be handled practically.

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.

A Dancing Queen

Author: Stefan Toman

Lea is a passionate dancer. She has a group of friends who dance together with her and sometimes they even participate in dancing competitions. The next contest is scheduled for tomorrow and Lea's team is working on its choreography. For each move they know which team member performs it best and only that person should dance it. Since all of them have a similar dancing level, it happened that each team member appears exactly two times on this list. Each member of the group may dance at most one of the moves during their performance, so they have to decide for each dancer which one of the two moves to include in their show.

There will also be judges at the competition. Surprisingly, the judges do not care that much about whether the dancers' moves fit together, but they want to see some special moves they really like. Each of the judges has a list with some moves he wants to see. Since the judges do not know which dancer is going to perform which move it may happen, that several or no moves assigned to one dancer appear on their lists.

Lea's team wants to make all judges happy for obvious reasons. Therefore, their plan is to perform dancing moves in a way that each judge sees at least one of the moves on his list. But is this even possible?

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 begins with a line consisting of two integers m, the number of dancers, numbered from 1 to m and n, the number of judges. n lines follow. The i-th line contains the space-separated list of dance moves judge i likes most. Each of these lines contains several integers, where a positive integer a means the first move of dancer a and a negative integer $-b$ means the second move of dancer b. The judge's lines end with 0.

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 there is an assignment of dancing moves such that each judge sees at east one move from his list or "no" otherwise. Each line of the output should end with a line break.

Constraints

- $1 \leq t \leq 20$
- $1 \leq m \leq 150$

- $1 \leq n \leq 350$
- Each judge has at most m entries on his list.

Sample Data

Input

```
8
2 2
1 -2 0
1 2 0

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

5 2
0
-1 -4 0

4 3
0
3 0
-1 3 0

8 5
0
0
5 0
0
-4 0

9 6
1 4 -6 0
0
0
0
1 0
0

19 5
-1 -14 0
4 -15 0
9 12 0
7 10 -17 0
9 0

16 7
0
1 -12 0
0
14 0
15 0
12 0
0
```

Output

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

B Queens Problem

Author: Philipp Hoffmann

Queens are quite picky persons. Lea currently has to deal with an especially nasty one. Her task is to create a painting for the Queen of Templonia, but of course the queen has certain preferences: She wants the painting to be a quadratic grid of $n \times n$ cells, where n is a natural number. Furthermore, n of the squares in the grid should be filled, the rest should be left empty. Any two filled squares should not be in the same row, in the same column, or on the same diagonal through the grid. (E.g.: (1,2) and (3,4) are on the same diagonal, (3,4) and (4,3) are on the same diagonal, but (3,3) is not on the same diagonal as any of the others.)

Lea has prepared multiple paintings. For each she has already decided on a number n, drawn the grid and in some cases even filled out some squares. Can you help her decide which of her paintings can be finished according to the constraints above?

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 an integer n indicating that the grid has dimension $n \times n$. n lines follow, each with exactly n characters. The i-th line describes row i in the grid, the j-th character describes square j. A "." denotes an empty square, an "x" denotes a filled square.

Output

For each test case, output one line containing "Case #i:" where i is its number, starting at 1. If there is a possibility to fill the grid according to the constraints above, output n more lines containing the solution in the same format as the input. If there are multiple solutions, any will be accepted. If it is not possible to complete the grid, output the line "impossible". Each line of the output should end with a line break.

Constraints

- $1 \leq t \leq 20$
- $1 \leq n \leq 15$

Sample Data

Input

```
8
4
..x.
....
....
....

4
..x.
....
x...
....

4
..x.
x...
...x
.x..

2
..
..

1
x

7
...x...
.......
.......
.x.....
.......
.......
......x

5
.x...
...x.
x...x
x....
.....

6
......
......
.....x
...x..
......
......
```

Output

```
Case #1:
..x.
x...
...x
.x..
Case #2:
impossible
Case #3:
..x.
x...
...x
.x..
Case #4:
impossible
Case #5:
x
Case #6:
...x...
x......
....x..
.x.....
.....x.
..x....
......x
Case #7:
impossible
Case #8:
impossible
```

C Brewery Scheduling

Author: Chris Pinkau

After a warm summer day, Lea enjoys a nice cold beverage while sitting in the grass down by the river and watching the sunset. Like most of the people from the region she comes from, she usually enjoys a beer on these occasions. And after having a few sips of the exquisite golden liquid, she contemplates the work that is put behind brewing such a masterpiece. Thus, she decides to visit the **BIER** (**B**rewery of **I**nternational **E**xcellence and **R**elevance), one of the many local breweries, on the next day to learn a bit more about the process behind her favourite beverage. Apart from all the usual stuff about brewing, Lea learns about how the workers in the brewery are scheduled. There are many different tasks to be fulfilled, from unloading barley or hops to operate heavy machinery, and many workers that can fulfil these tasks. The brewery has clustered the tasks into several groups, where each group consists of a chain of tasks that has to be performed in succession, but is independent from tasks in other groups. Some of the tasks require the work of several workers simultaneously, but they all require one time unit. As quirky as Lea is, she thinks that the scheduling process and the schedule itself can be made better and after returning home, she immediately starts working on finding a perfect solution. And she might be right. For the sake of better beer, can you help Lea find an optimal personnel schedule?

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 integers n, m and d, with n being the number of tasks (indexed from 1 to n), m being the number of workers, and d being the deadline where all tasks need to be finished (i.e., the schedule starts at time 0 and should be finished upon timeslot d, i.e. at time d at the latest). n lines follow describing the tasks, where line i contains two integers c_i and p_i. c_i denotes the category of task i, i.e., how many workers are needed to perform task i, and p_i denotes the task that needs to be finished before the processing of task i can start. (A task with no predecessor task has $p_i = -1$.)

Output

For each test case, output one line containing "Case #i: x" where i is its number, starting at 1, and x is either: a schedule that takes at most d time units; or "impossible" if there is no such schedule. A schedule consists of a sequence $t_1\ t_2\ \ldots\ t_n$ where t_i is the timeslot task i is processed (e.g. timeslot j means that a task is started at time $j - 1$ and finished its processing at time j) and at most m workers may be working simultaneously.

Constraints

- $1 \leq t \leq 50$

- $1 \leq n \leq 13$
- $1 \leq m \leq 50$
- $1 \leq d \leq 30$
- $1 \leq c_i \leq m$ for all $1 \leq i \leq n$
- $1 \leq p_i \leq n$ for all $1 \leq i \leq n$
- $1 \leq t_i \leq d$ for all $1 \leq i \leq n$

Sample Data

Input

```
9
6 5 4
4 -1
3 1
1 2
3 -1
3 4
1 -1

5 6 6
3 -1
4 -1
4 -1
3 -1
3 -1

5 3 3
1 -1
3 1
2 2
3 -1
3 -1

3 1 6
1 -1
1 1
1 2

2 1 3
1 -1
1 1

1 1 4
1 -1

4 3 6
3 -1
1 -1
1 2
1 -1

3 8 5
5 -1
7 -1
3 -1

5 7 6
7 -1
7 1
6 2
7 -1
2 -1
```

Output

```
Case #1: 1 2 3 3 4 1
Case #2: 1 3 4 1 2
Case #3: impossible
Case #4: 1 2 3
Case #5: 1 2
Case #6: 1
Case #7: 1 2 3 2
Case #8: 1 2 1
Case #9: 1 4 5 2 3
```

D Alarm Clock

Author: Philipp Hoffmann

Lea is not a morning person. In fact, she hates getting up. Whenever she is in a particularly bad mood, she might even throw her alarm clock across the room and against the wall. She has been doing this for quite a while now and her clock is starting to malfunction.

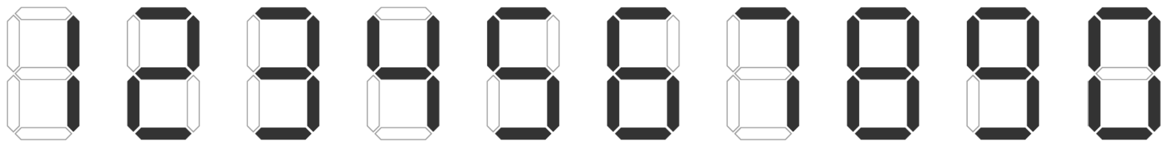

Figure 1: The seven segments of Lea's clock

Lea's clock is a 24-hour digital clock with seven segments per number (see above). She suspects that some of the segments have been damaged and never light up while others still work fine, but Lea does now know which do work and which do not. Now she has a hard time figuring out how late it is. She has already watched the clock for some time, recording what it showed every minute, but she cannot figure out the time. Help her!

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, n lines follow. Each of the following lines contains a string in the format "xx:xx" where each "x" is a digit from 0 to 9. The first line contains the time the clock showed initially, the second line the time one minute later and so on.

Output

For each test case, output one line containing "Case #i:" where i is its number, starting at 1. Output one more line formatted "xx:xx" for each possible time at which Lea could have started to watch the clock, sorted in ascending order. Under the assumption that some segments work and others never light up, the times must be consistent with all observations Lea made.

It might still be possible that the assumption is wrong and/or the clock itself is broken, and no time is consistent with all observations. In that case, output "none". It is theoretically possible that a broken clock shows a shape which does not correspond to any digit from 0 to 9, however Lea never observed such an event.

Each line of the output should end with a line break.

Constraints

- $1 \le t \le 20$
- $1 \le n \le 20$

Sample Data

Input

```
7
1
23:49

1
88:88

10
11:24
11:25
11:26
11:27
11:28
11:29
11:30
11:31
11:32
11:33

4
12:73
12:74
12:75
12:75

5
16:11
16:12
16:13
16:14
16:15

7
22:77
22:78
22:79
22:10
22:11
22:12
22:13

3
18:70
18:71
18:72
```

Output

```
Case #1:
23:48
23:49
Case #2:
none
Case #3:
00:24
01:24
03:24
04:24
07:24
08:24
09:24
10:24
11:24
13:24
14:24
17:24
18:24
19:24
Case #4:
02:03
02:33
08:03
08:33
12:03
12:33
18:03
18:33
Case #5:
06:01
06:11
06:31
06:41
08:01
08:11
08:31
08:41
16:01
16:11
16:31
16:41
18:01
18:11
18:31
18:41
Case #6:
22:07
22:37
Case #7:
08:00
08:30
18:00
18:30
```

E Planetarium Problem

Author: Chris Pinkau

As every year, the local planetarium runs a series of practical courses and internships which are free for the public. However, as the number of interested people was at a record high last year and is expected to rise yet again this time, the planetarium has decided to have trials in order to find out which people are actually taking part. Lea has always had a great interest in astronomy and astrophysics, so she likes to take part in some courses to learn more about the great mysteries of the universe. The task she is assigned is fairly simple: she is given a number of astronomical objects and has to decide in how many clusters this vast amount of objects can be divided. The restriction for the division is that all clusters must be of equal size and that no cluster can be subdivided any further into smaller clusters of equal size, except for trivial clusters of size 1. Lea has thought about the problem a lot, but cannot quite get to a point where she can decide how to do it efficiently. As usual, she asks you for help. What is your answer?

Input

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

Each test case consists of only one integer n, the number of astronomical objects considered.

Output

For each test case, output one line containing "Case #i:" where i is its number, starting at 1, and a sequence of all possible cluster sizes, in increasing order, separated by spaces.

Constraints

- $1 \leq t \leq 20$
- $1 \leq n \leq 10^{85}$

Sample Data

Input

```
9
16445
357
563233
41578
15789543
44155578
1574682
852636
19
```

Output

```
Case #1: 1 5 11 13 23
Case #2: 1 3 7 17
Case #3: 1 11 51203
Case #4: 1 2 20789
Case #5: 1 3 7 11 29 2357
Case #6: 1 2 3 37 198899
Case #7: 1 2 3 19 727
Case #8: 1 2 3 41 1733
Case #9: 1 19
```

F Garden Layout

Author: Philipp Hoffmann

Since Minecraft's huge success, many people have been inspired to bring cube layouts to the real world. For this reason, Lea has been tasked with creating a Minecraft garden. In general, such gardens are created over a fixed ground area that is divided into $1m \times 1m$ squares. On those squares, blocks of dimension $1m \times 1m \times 1m$ are stacked. You can imagine the resulting garden as a set of heights, one for each square.

For this special garden, the owner has already decided the place and height for the watering system as well as the places and heights for his plants. Lea should now decide the heights of all the other squares. The garden should fulfill a very special constraint: It should be possible that the water reaches all of the plants (starting of course from the watering system). As everyone knows, water flows from a square a to an adjacent square b if and only if the height of b is equal or less than the height of a. Whether or not a plant is planted on a square has no influence on the flow of water.

Lea is overwhelmed: Which of the many many possible layouts should she pick? She does not even know how many there are! Help her by answering that second question.

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 h and w. h is the maximum height allowed in the garden, the minimum height allowed is always 0. w is the height of the watering system.

4 lines follow, each containing 4 characters, those lines describe the garden layout.

An underscore ("_") denotes something that does not belong to the garden (no water can flow here), a digit x denotes a flower at height x, an asterisk ("*") denotes the watering system and a question mark ("?") denotes a square that belongs to the garden, but has neither a plant nor the watering system on it. There will always be exactly one asterisk.

Output

For each testcase, output one line containing "Case #i: k" where i is its number, starting at 1, and k is the number of garden layouts that fulfill all the above constraints. Each line of the output should end with a line break.

Constraints

- $1 \leq t \leq 20$
- There will be at most 7 question marks in the layout.
- $0 \leq w, x \leq h \leq 6$

Sample Data

Input

```
9
1 1
*0__
____
____
____

2 2
_*__
1?1_
_1__
____

3 3
__*_
__?_
???_
121_

3 3
____
*?__
_?2_
____

2 1
1*0_
1??_
____
____

2 2
__1*
__?0
__??
__??

5 3
___3
___*
__?2
__0_

3 1
____
000_
*0??
1??_

1 0
____
?100
*0??
??__
```

Output

```
Case #1: 1
Case #2: 2
Case #3: 16
Case #4: 3
Case #5: 9
Case #6: 243
Case #7: 3
Case #8: 256
Case #9: 0
```

G Tetris

Author: Stefan Toman

Just one more level! Lea is on her way to finally beat Bea's highscore at Tetris. They play an advanced Tetris game with unusual types of pieces which is extremely hard to master. The pieces are a connected set of tiles which are of size 1×1. The pieces fall down from top of the screen until they hit another piece. The game is so fast that is is impossible to position pieces while they are falling down. Lea can just move and rotate them in the fraction of a second before they enter the board. Formally speaking, Lea can choose a horizontal offset and a rotation for her tile, then it is falling down until it shares a horizontal edge with another piece or the bottom of the board. The final position of the piece is legal if and only if the piece does not intersect with tiles already on the board and fits completely on the board. If a piece does not fit on the board the game is over: Lea keeps her score up to this piece.

If one line is fully filled with tiles it is erased and all tiles above fall down exactly one line. To make advanced strategic planning possible, the game shows the next few pieces to the player. Can you tell Lea how many lines she can fill up completely?

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 pieces that will be falling down. $n + 1$ blocks follow, the first one describing the board and the next n ones describing the pieces that will arrive in order.

Each block starts with two integers w and h, the block's width and height. h lines follow, containing w characters each. Each of these characters will be X for a used tile or a dot for an unused tile.

Output

For each test case, output one line containing "Case #i: y" where i is its number, starting at 1, and y is the maximum number of lines that Lea can eliminate.

Constraints

- $1 \leq t \leq 10$
- $1 \leq n \leq 4$
- $5 \leq w \leq 10$, $5 \leq h \leq 20$ for the board.
- $1 \leq w \leq 5$, $1 \leq h \leq 5$ for the pieces.

- All pieces will be connected and there will not be empty lines (horizontal or vertical) in the pieces falling down
- Initially, the board does not contain lines full of tiles.

Sample Data

Input

```
5
1
5 5
.....
X....
XX...
XX...
XXX..
4 1
XXXX

1
5 5
.....
.....
.....
X...X
X.X.X
2 3
XX
.X
XX

3
7 10
.......
.......
.......
.......
X......
XX..XXX
XXX.XXX
XXX.XXX
XXXXXX.
.XXX.XX
3 2
XX.
.XX
2 3
.X
.X
XX
2 2
XX
XX

1
7 6
.......
.......
.......
.......
.......
..XXXXX
3 3
.XX
.XX
XXX

1
6 6
......
......
X.X.XX
.X..X.
XXX...
...XXX
4 1
XXXX
```

Output

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