Technische Universität München
Fakultät für Informatik
Lehrstuhl für Effiziente Algorithmen (LEA)
Prof. Dr. Ernst W. Mayr
Christian Müller, Philipp Hoffmann, Chris Pinkau, Stefan Toman

Summer Term 2015
Problem Set 6
28.05.2015

Algorithms for Programming Contests

This problem set is due by

Thursday, 11.06.2015, 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.

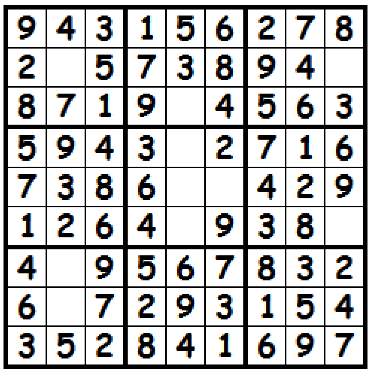
SS15N06A   Sudoku

Author: Stefan Toman

Everybody loves them and Lea is crazy about them as well: Sudokus. Recently, Lea bought the new book SUKODU 42 (Super-Ultra-Killer-Omniscient, Distracting and Ultimate Sudokus Part 42) containing lots of the hardest Sudokus on earth. Some of them are insane, so Lea decides to get some more practice before. Her training schedule contains three hours physical excercise per day as well as six hours solving Sudokus and in the evening two more hours solving other riddles.

During the last weeks Lea solved an awful amount of Sudokus. Now she thinks about creating a computer program to help her in solving them. Can you write an algorithm for her to solve arbitrary Sudokus?

In case you do not know the rules of Sudoku: There is a 9x9 board that needs to be filled with the integers from 1 to 9, one per field. Each number may occur only once per row, column and region (the board is divided into nine 3x3 square regions). Some of the fields are already filled, you need to fill the other ones. There is always a unique solution. The first test case of the sample input looks like this:



9	4	3	1	5	6	2	7	8
2		5	7	3	8	9	4	
8	7	1	9		4	5	6	3
5	9	4	3		2	7	1	6
7	3	8	6			4	2	9
1	2	6	4		9	3	8	
4		9	5	6	7	8	3	2
6		7	2	9	3	1	5	4
3	5	2	8	4	1	6	9	7

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 partly filled Sudoku grid, namely of exactly nine lines containing 9 characters each. All of these characters are non-zero integers or a question mark in case of unknown fields.

Output

For each test case, output one line containing "Case #i:" where i is its number, starting at 1. Then output a line break and the solved Sudoku in the same format as given in the input.

Constraints

- $1 \leq t \leq 250$
- There will be at most twenty question marks per Sudoku.

Sample Data

Input

```
2
943156278
2?573894?
8719?4563
5943?2716
7386??429
1264?938?
4?9567832
6?7293154
352841697

895164372
7618329?5
32??97186
5469?3?18
?82745693
93768152?
2?3459?61
65?218437
418376259
```

Output

```
Case #1:
943156278
265738941
871924563
594382716
738615429
126479385
419567832
687293154
352841697

Case #2:
895164372
761832945
324597186
546923718
182745693
937681524
273459861
659218437
418376259
```

SS15N06B Story Time

Author: Christian Müller

On another boring afternoon a few months ago, Lea decided to write a fantasy book which she will call “Game of Kings”, an epic tale about kings and queens, bishops and knights. Oh, and magic.

She has spent many nights imagining what would happen to all the awesome characters she would describe. Now, she just needs to compile the list of events into one book. For every character, she wrote a list of chapters which are centered around him or her. These have to happen in a fixed order.

Also, some chapters have a specific ordering. For example, the scene in which Queen Lena, nicknamed the “Black Queen”, has a nervous breakdown over a letter detailing the abduction of her brother, a famous knight, has to happen after the description of the abduction from her brothers point of view. Lastly, two sequential chapters should not be centered around the same character.

Can you tell her in how many possible ways she can arrange all the scenes she imagined?

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 two integers n and m, where n is the number of characters (indexed from 1 to n) and m is the number of dependencies between chapters. n lines follow. The i-th line consists of a single integer a_i, the amount of chapters that are centered around character i. m lines follow. The j-th line consists of four integers c_1, p_1, c_2, and p_2 and means that the p_1-th chapter centered around character c_1 has to happen before the p_2-th chapter centered around c_2.

Output

For each test case, output one line containing “Case #i: x” where i is its number, starting at 1, and x is the amount of possible orderings of the chapters such that:

- Sequential chapters are about different characters.
- Chapters about a single character are in a fixed order.
- All m inter-chapter-dependencies are fulfilled.

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

Constraints

- $1 \le t \le 20$
- $1 \le n \le 6$
- $0 \le m \le 15$
- $1 \le c_i \le a_i \le 15$
- $p_1 \ne p_2$
- $1 \le p_i \le n$
- There are at most 13 chapters.

Sample Data

Input

```
2
3 0
2
2
2

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

Output

```
Case #1: 30
Case #2: 570
```

SS15N06C 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

```
2
16445
357
```

Output

```
Case #1: 1 5 11 13 23 16445
Case #2: 1 3 7 17 357
```

SS15N06D 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.

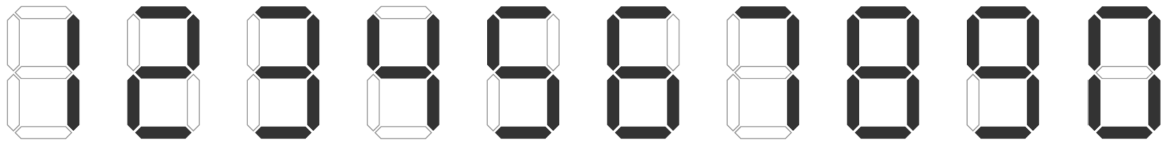


Figure 1: The seven segments of Lea's clock

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

- $1 \leq n \leq 20$

Sample Data

Input

```
2
1
23:49

1
88:88
```

Output

```
Case #1:
23:48
23:49
Case #2:
none
```

SS15N06E 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

```
2
4
..x.
....
....
....

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

Output

```
Case #1:
..x.
x...
...x
.x..
Case #2:
impossible
```

SS15N06F 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

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

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

Output

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

SS15N06G More Dimensions!

Author: Philipp Hoffmann

OCTAS (**O**verly **C**omplicated **T**o **A**nnoy **S**tudents) is an esoteric programming language that is quite capable of computing many things, but maybe not as useful as other languages. Nevertheless, Lea has decided to write a sample program in it. Sadly, her program has already run for quite some time without terminating. She is not sure whether this is due to her code being wrong or because the interpreter is very slow. Can you answer her question?

To solve this problem, you need to know how OCTAS works: OCTAS programs operate with a two-dimensional array of 16*16 unsigned bytes initially filled with zeroes and a pointer which points to an array position. The array pointer to the current cell thus has an x- and a y-coordinate and starts at $(0, 0)$. Remember that bytes can only store values from 0 to 255 and will over-/underflow as usual, that is $255 + 1 = 0$.

The code is given in a rectangular grid, reading starts as usual with reading direction "right" in the upper left corner. A program terminates as soon as the next command to be read is outside of the rectangular grid.

There are 12 commands in OCTAS, their interpretation is defined below. Additionally, the grid may contain spaces. These do not have an associated action.

ˆ	increase the y-coordinate of the array pointer
v	decrease the y-coordinate of the array pointer
>	increase the x-coordinate of the array pointer
<	decrease the x-coordinate of the array pointer
U	change reading direction of the code to "up"
D	change reading direction of the code to "down"
R	change reading direction of the code to "right"
L	change reading direction of the code to "left"
+	increase the content of the current cell by 1 (mod 256)
-	decrease the content of the current cell by 1 (mod 256)
.	output the content of the current cell interpreted as ASCII character (see e.g. `http://www.asciitable.com`)
/	if the array pointer points to a cell with content 0, skip the next (according to the current reading direction) character or space in the code

Input

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

The first line of each test case contains two integers, n and m, n lines follow containing m characters each. These n lines are the code, they contain only characters given above.

Output

For each test case, output one line containing "Case #i: x" where x is the output the program will produce, or "endless loop" if the program will not terminate. You can assume that each program will either loop endlessly or terminate after at most 50 million commands (not counting spaces and skipped commands). Each line of the output should end with a line break.

Constraints

- $1 \leq t \leq 5$
- $1 \leq m, n \leq 100$
- The array pointer will never leave the bounds of the array.

Sample Data

Input

```
2
2 27
+++++D               LR>.
     R->+++++++++++++</UU

1 4
R+-L
```

Output

```
Case #1: A
Case #2: endless loop
```