acm International Collegiate Programming Contest

IBM event sponsor

ACM-ICPC Asia Regional Contest

Shanghai Site

Donghua University

Shanghai
October 2009

Problems List

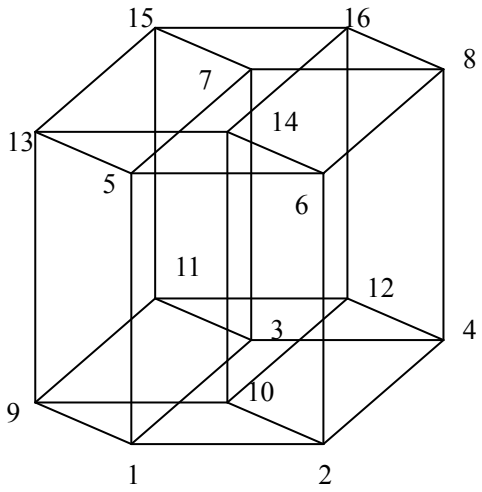
Problem A: Alice's Cube
Problem B: Brute-force Algorithm
Problem C: Compressed String
Problem D: Decrypt Messages
Problem E: Exciting Time
Problem F: Flowers Placement
Problem G: Game Simulator
Problem H: Heroes Arrangement
Problem I: Island Explorer
Problem J: Jinyuetuan Puzzle

Problem A: Alice's Cube

[Description]

Alice has received a hypercube toy as her birthday present. This hypercube has 16 vertices, numbered from 1 to 16, as illustrated below. On every vertex, there is a light bulb that can be turned on or off. Initially, eight of the light bulbs are turned on and the other eight are turned off. You are allowed to switch the states of two adjacent light bulbs with different states ("on" to "off", and "off" to "on"; specifically, swap their states) in one operation.

Given the initial state of the lights, your task is to calculate the minimum number of steps needed to achieve the target state, in which the light bulbs on the sub cube (1,2,3,4)-(5,6,7,8) are turned off, and the rest of them are turned on.



[Input]

There are multiple test cases. The first line of the input contains an integer **T**, meaning the number of the test cases. There are about 13000 test cases in total. For each test case there are 16 numbers in a single line, the i-th number is 1 meaning the light of the i-th vertex on the picture is on, and otherwise it's off.

[Output]

For every test cases output a number with case number meaning the minimum steps needed to achieve the goal. If the number is larger than 3, you should output "more".

Sample Input	Sample Output
3	Case #1: 0

0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 0 1 0 0 0 0 0 0 1 0 1 1 1 1 1 1 0 0 0 0 0 0 1 0 1 0 1 1 1 1 1 1	Case #2: 1 Case #3: more

Problem B: Brute-force Algorithm

[Description]

Professor Brute is not good at algorithm design. Once he was asked to solve a path finding problem. He worked on it for several days and finally came up with the following algorithm:

```
Function Find(integer n,function func)
    If n=1
        For i = 1 to a do func()
    Elseif n=2
        For i = 1 to b do func()
    Else Find(n-1,Find(n-2,func))

Function Main
    Find(n,funny)
```

Any fool but Brute knows that the function "funny" will be called too many times. Brute wants to investigate the number of times the function will be called, but he is too lazy to do it.

Now your task is to calculate how many times the function "funny" will be called, for the given **a**, **b** and **n**. Because the answer may be too large, you should output the answer module by **P**.

[Input]

There are multiple test cases. The first line of the input contains an integer **T**, meaning the number of the test cases.

For each test cases, there are four integers **a**, **b**, **P** and **n** in a single line. You can assume that $1 \le \mathbf{n} \le 1000000000$, $1 \le \mathbf{P} \le 1000000$, $0 \le \mathbf{a}, \mathbf{b} < 1000000$.

[Output]

For each test case, output the answer with case number in a single line.

Sample Input	Sample Output
3 3 4 10 3 4 5 13 5 3 2 19 100	Case #1: 2 Case #2: 11 Case #3: 12

Problem C: Compressed String

[Description]

Dealing with super long character strings is Chris's daily work. Unfortunately, the strings are so long that even the fastest computer in the world cannot work with them.

Chris does her work in a smart way by compressing the strings into shorter expressions. She does her compression for each string in the following way: Find a consecutive repeated substring of the original string, e.g. "ab" in "cababab".

Replace the repeating part with the bracketed repetend, followed by the times the repetend appears in the original string. e.g. Write "cababab" as "c[ab]3d". Note she can also write it as "c[ab]1ababd" or "ca[ba]2bd" and so on, although these string are not compressed as well as the first one is.

Repeat a) and b) several times until the string is short enough.

Chris does her compression quite well. But as you know, the work is boring and a waste of time. Chris has written a computer program to help her do the boring work. Unfortunately, there is something wrong with the program; it often outputs an incorrect result. To help her debug the program, you are ordered to write a debugger which can compare Chris's standard compressed string against the string compressed by the program.

[Input]

There are multiple test cases.

The first line of the input contains an integer **T**, meaning the number of the test cases.

For each test case, there are two lines of character strings which the first one is Chris's standard compressed string and the second one is the program's compressed string. Both string contains only lowercase letters (a-z), square brackets ([]) and numbers (0-9). The brackets must be followed with an integer indicating the times the string in the brackets repeat, note that the repeat time can be zero. The brackets can be nested.

You can assume all the compressed strings in the input are no longer than 20. See further details in the input sample.

[Output]

For each test case, output case number first. And then if the two uncompressed strings are the same, output "YES" in a single line; otherwise, output "NO" followed by the first position where the uncompressed strings differ.

Sample Input	Sample Output
5	Case #1: YES

a[a]12 [[a]3]4a [z]12 zzzzzzzzz [a[ba]2b]12 [ab]36 [a]123123123[icpc]2 [[a]123]1001001inter aismoreeasierthanc gismuchharderthanj	Case #2: NO 10 Case #3: YES Case #4: NO 123123125 Case #5: NO 1

[Hint]

For sample test case 3, the first string "[a[ba]2b]12" can be written as "[abababab]12", then "[[ab]3]12", finally we get "[ab]36".

The numbers in this task may be very large and cannot be stored in a 32 bit integer.

Problem D: Decrypt Messages

[Description]

In the game BioHazard 4, the American president's daughter has been abducted by some crazy villagers. Leon S. Kennedy, the secret agent of White House, goes to rescue her. He keeps in contact with Hunnigan, the president's secretary. But the time in their contact log has been encrypted, using the following method: Count the number of seconds from 2000.01.01 00:00:00 to that time, assume this number is **x**. Then calculate $\mathbf{x}^{\mathbf{q}}$, modulo it by a prime number **p**. The remainder **a** is the encrypted number.

Your task is to help Leon write a program to decrypt the contact log, and tell him all the possible original time.

1. Remember that if the year can be divided evenly by 4 but can't be divided evenly by 100, or it can be divided evenly by 400, this year is a leap year. The February of a leap year has 29 days, while the February of other years has 28 days.
2. In this problem, if the year modulo 10 is 5 or 8, at the end of this year, there is one "leap second", i.e., the second after 2005.12.31 23:59:59 is 2005.12.31 23:59:60, and after that second, it's 2006.01.01 00:00:00.

You may assume that from 2000.01.01 00:00:00 till that time, less than **p** seconds have passed.

[Input]

There are multiple test cases.

The first line of the input contains an integer **T**, meaning the number of the test cases.

For each test case, a single line of three integers: **p**, **q**, and **a**. ($2<\mathbf{p}\leq 1000000007$, $1<\mathbf{q}\leq 10$, $0\leq \mathbf{a}<\mathbf{p}$, **p** is always a prime.)

[Output]

The time. If there are multiple solutions, you must output all possible solutions in chronological order.

If the solution doesn't exist, output *Transmission error* instead.

See the sample output for further details.

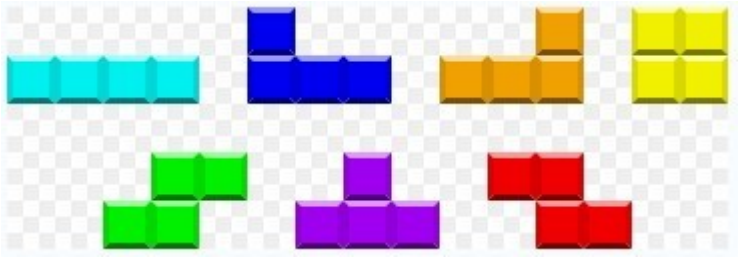
Sample Input	Sample Output
2 3 2 1 3 2 2	Case #1: 2000.01.01 00:00:01 2000.01.01 00:00:02 Case #2: Transmission error

Problem E: Exciting Time

[Description]

It's exciting time! Now let's consider the most famous video game in the world: Tetris.

Tetris is a puzzle video game originally designed and programmed by Alexey Pajitnov. It was created on June 6, 1984. The game is available for nearly every video game console and computer operating system, as well as devices such as graphing calculators, mobile phones, portable media players, PDAs and even as an Easter egg on non-media products like oscilloscopes.



Some random tetrominoes (shapes each composed of four square blocks) will fall down sequentially into the playing field. The objective of the game is to manipulate those tetrominoes, by moving each one sideways and/or rotating it, with the aim of creating a horizontal line of blocks without gaps. When such a line is created, it disappears, and any block above that line will fall until it hits an obstacle. See the sample test case for further details.

Tetris game manuals refer to the seven one-side tetrominoes in Tetris as I, J, L, O, S, T and Z (due to their resembling letters of the alphabet). All are capable of single and double clears. I, J and L are able to clear triples. Only the I tetromino has the capacity to clear four lines simultaneously, and this is referred to as a "tetris".

Wikipedia (http://en.wikipedia.org/wiki/Tetris)

The scoring formula for the majority of Tetris products is based on the idea that more difficult line clears should be awarded more points. Specifically, a single clear will be awarded 100 points, 250 points for a double clear, 400 points for a triple clear, and 1000 points for a "tetris".

Now you are given the width of the playing field and a sequence of tetrominoes with the position and degrees it rotated. The task is to calculate how many points will be awarded.

You can assume the playing field is high enough, so that tetrominoes will not overlap each other even when a tetromino is created at the top of the playing field.

[Input]

There are multiple test cases.

The first line of the input contains an integer **T**, meaning the number of the test cases.

The first line of each case contains two integers **W** and **N** (1≤**W**≤30000, 0≤**N**≤30000), where **W** is the width of the playing field and **N** is the number of tetrominoes.

Then **N** lines follow. Each line contains the description of a single tetromino. The type of the tetromino (I, J, L, O, S, T or Z) comes first, followed by an integer **ID**, meaning the id of the column of the leftmost square block in the tetromino. The columns are numbered in increasing order from left to right, beginning with zero. The last number shows the degree the tetromino should be rotated; it can only be 0, 90, 180 or 270. The one with 0-degree rotation is shown in the above picture.

Note that the rotate operations are in clockwise. And the order of tetrominoes is the same as the input sequence.

See the sample input for further details.

[Output]

For each test case you should output three lines.

The first line contains the case number.

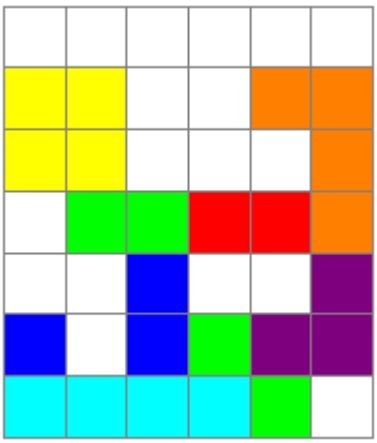
The second line contains a single integer which is the points awarded in total.

The third line contains W integers, the i-th number is the height of the highest square block in the i-th column, separated by a single whitespace.

Sample Input	Sample Output
2 6 9 I 0 0 S 3 90 T 4 270 J 0 0 J 2 90 S 0 0 Z 3 0 O 0 0 L 4 270 1 1 I 0 90	Case #1: 200 6 6 4 4 6 6 Case #2: 1000 0

[Hint]

The final state of the game field is shown below.



Problem F: Flowers Placement

[Description]

Bytetown has a long tradition in organizing an international flower exhibition. Professor Feuerbach, a true flower lover, visited the exhibition this year. He was pleased by the most beautiful flowers and other plants around the world roses, orchids, magnolias, cactuses. All flowers were nicely placed.

The flower placement that was the most appealing to him was composed of many kinds of flowers placed in a rectangular grid, such that each row of the grid contained each kind of flowers exactly once and each column of the grid contained each kind of flowers at most once.

Professor Feuerbach is a good mathematician and soon he realized that the number of columns of the grid has to be the same as the number of different kinds of flowers in the placement. The different kinds of flowers are represented by numbers **1, 2..., N**, where **N** is the number of different kinds of flowers. Soon he encountered a new problem. He would like to add one row of flowers to the placement without violating the rules stated above. (Note that he may not modify the existing rows and therefore he may not use any new kinds of flowers in the new rows.) It's quite easy, but he wants to know the K-th lexicographically valid placement.

[Input]

The input data set describes about 100 flower placements. In the first line the number of placements is given.

Each flower placement description begins with three positive integers **N** ($1 \le N \le 200$), **M** ($0 \le M \le N$), **K** ($1 \le K \le 200$) representing the number of columns and rows of the grid. The following **M** lines contain **N** integers each representing the kinds of flowers in one row of the placement.

[Output]

For each flower placement output one line containing N numbers, describing the added row. The numbers in the output file should be separated by exactly one space. If there are less than K valid placements, output -1 instead.

Sample Input	Sample Output
2 3 2 2 3 2 1 1 3 2 4 2 2 1 4 3 2 2 1 4 3	Case #1: -1 Case #2: 4 3 2 1

Problem G: Game Simulator

The teams who solve this problem will receive two decks of poker as prize.

[Description]

"Tractor" is a very popular poker game in China. There are four players in the game. Suppose their names are Alice, Bob, Charles and David, in clockwise order. A judge is needed for this game. The players are divided into two teams, Alice and Charles are in team 1, and the other two are in team 2. The prop they use to play the game are two decks of pokers, including 108 cards in total. A simplified rule of the game is described below.

The whole game contains a number of rounds. In each round, one team is called **"Declarers" (CT)**; the other team is called **"Defenders" (FT)**. Each team has a **current rank (CR)**. The goal of the player is to increase his own team's CR as much as possible.

A certain round has a **Main Suit** (Heart - H, Spade - S, Club - C, Diamond - D, or None - no main suit in this round) and its CR. The CR in this round is the CR of the CT, and Main Suit will be given. The Main Suit and CR will be used to determine the order of the cards.

Cards ranked 5, 10, King value 5, 10, 10 pts (points) respectively, all other cards value 0 pts. In one round, we only consider the FT's pts. The rules of getting pts for FT will be discussed later.

If the FT gets less than 80 pts in one round, they will hold be FT in the next round. This situation is called **"make"**. Otherwise, they become CT in the next round and the original CT become FT instead. This situation is called **"down"**.

If the FT gets 0 pts, the CR of the current CT will be increased by 3, for example, if the CR of the CT is 9, it will become Queen (12). Otherwise, if the FT gets less than 40 pts, the CR of the CT will be increased by 2. Otherwise, if the FT gets less than 80 pts, the CR of the CT will be increased by 1. Otherwise, if the FT gets not less than $80 + k * 40$ pts and less than $120 + k * 40$ pts, the CR of the current FT will be increased by k. For example, if the FT gets 255 pts in a round, the CR of the current FT will be increased by 4; and if the FT gets 80 pts, both teams' CR remain unchanged. If a team's CR becomes beyond Ace, this team is considered the WINNER of the whole game.

During a round, one of the players in CT is called the **dealer**. If "make", the pard (teammate) of the dealer becomes the next round's dealer. Otherwise ("down"), the player on dealer's right-hand side becomes the dealer of the next round. For example, if the dealer of the current round is Alice and her team (CT) is "down", the dealer of the next round should be Bob (on Alice's right-hand side).

At the start of a round, each of the players except the dealer gets exactly 25 cards; the dealer gets all the remaining 33 cards. After that, the dealer chooses 8 of his cards and gives them to the judge, and these cards are called **"hidden cards"**.

Now each player has exactly 25 cards. A round consists of several **tricks**. In

the first trick, the dealer plays one or more cards (called **"lead"**), then, in clockwise order, players play the same number of cards as the first player one by one (called **"follow"**). The winner of the current trick leads cards during the next trick, and so on. If the winner of the current trick is a member of FT, then the FT gets the sum of the cards' pts played in this trick.

Now we start to describe how to determine the winner of a trick. After the main suit and the CR of the current round are fixed, we can determine the **"trumps"** which are cards with main suit or CR (Current Rank), and the jokers. All other cards are **"not-trumps"**.

We can have an order among all the cards according the following rules:

(1) "Trumps" are ordered higher than "not-trumps".

(2) For the trumps, the order are listed below:

Red Joker

Black Joker

card with main suit and CR (if exists)

other card with CR

other trumps ordered by their ranks(i.e., A, K, Q, J, T, 9, 8, 7, ..., 3, 2)

(3) For the "not-trumps", they are ordered by their ranks.

Assume in all the description below, in the current round, the CR of the CT is 7.

Suppose the main suit is H, the cards can be arranged in this order (as an example):

S2 , C2 , D2
< S3 , C3 , D3
< S4 , C4 , D4
< S5 , C5 , D5
< S6 , C6 , D6
< S8 , C8 , D8
< S9 , C9 , D9
< ST , FT , CT (T - 10)
< SJ , CJ , DJ (J - Jack)
< SQ , CQ , DQ (Q - Queen)
< SK , CK , DK (K - King)
< SA , CA , DA (A - Ace)
< H2
< H3
< H4
< H5
< H6
< H8
< H9

< HT

< HJ

< HQ

< HK

< HA

< S7 = C7 = D7

< H7

< BJ (the Black Joker)

< RJ (the Red Joker)

If "None" during this round, then the pokers can be arranged in this order:

H2 , S2 , C2 , D2

< H3 , S3 , C3 , D3

< H4 , S4 , C4 , D4

< H5 , S5 , C5 , D5

< H6 , S6 , C6 , D6

< H8 , S8 , C8 , D8

< H9 , S9 , C9 , D9

< HT , ST , FT , CT

< HJ , SJ , CJ , DJ

< HQ , SQ , CQ , DQ

< HK , SK , CK , DK

< HA , SA , CA , DA

< H7 = S7 = C7 = D7

< BJ

< RJ

In these two tables, cards written in italic are "trumps", and cards written in boldface are "not-trumps".

In each trick, the **lead cards** (played by the player leading this trick) must be either all "trumps", or all "not-trumps" with same suit.

The possible **structures** of the cards are listed below (assume the main suit is H and main rank is 7 for the example):

Single. A single card, such as D9.

Pair. Two same cards, such as D9D9. But D7S7 is not a pair although their orders are the same.

Tractor. Two or more consecutive-ordered pairs, satisfying the condition that they are all "trumps", or all "not-trumps" with same suit, such as SJSJSQSQSKSKSASA, H7H7S7S7HAHA or RJRJBJBJ. But, these are not tractors: S7S7C7C7 (their orders are the same), C7C7C6C6 (they are not consecutive-ordered), DADAD2D2 (Ace is not "one", so they are not consecutive-ordered), H2H2H4H4, or D2D2D3. Be careful: if "None" in this round, H7H7S7S7HAHA is not a tractor (H7 and S7 are same-ordered

because of "None").

Throw. The combination of the structures above, satisfying the condition that they are all "trumps", or all "not-trumps" with same suit. Each of the Single, Pair or Tractor in a Throw is one of the Throw's component. In the original tractor game, in some situation, the throw will be rejected. But, to keep the rule simple, we assume in this problem all the throws are accepted. For example, RJRJBJBJH7H7HQHQHJHJH9H9H6H6HAH2 contains six components: two tractors, two pairs and two single cards (RJRJBJBJH7H7-HQHQHJHJ-H9H9-H6H6-HA-H2); CACAC8C8CK contains three components: two pairs and one single card(CACA-C8C8-CK).

A throw can be treated as different list of components, for example, H2H2H3H3H4H4H5H5H6H6 can also be treated as H2H2H3H3-H4H4H5H5H6H6, or H2H2-H3H3H4H4H4H5H5-H6H6, and so on. For the lead cards, each time we choose the longest component (choose the one with the highest order to break the tie) to construct a list of components, this list is the structure of the lead cards, also **the structure of the trick**. So that the structure of the trick is unique.

After the first player lead his or her cards, other players follow cards one by one in clockwise order, as mentioned above.

An important part of the game is to determine the winner of a trick:

If one's follow cards contain both "trumps" and "not-trumps", or all "not-trumps" but with different suits, this player can't be winner of this trick.

Otherwise, if the lead cards are all "not-trumps" and one's follow cards contain "not-trumps" with different suit from the lead cards, this follow player can't be the winner of this trick.

Else, if one's follow cards can't be constructed as the same structure of lead cards, this player can't be the winner of this trick either.

Otherwise, if the structure of this trick is not "throw", the one who played the highest-ordered card wins this trick. If more than one player played the same highest-ordered card, the winner of this trick will be the one who plays the highest-ordered card first.

Now let's consider the "throw" situation. We construct the follow cards into the structure of the lead cards, so that the order of the highest-ordered card in all the longest components of the "throw" is as high as possible (this card is called "**honor card**"). Note that tractor can be treated as several pairs or shorter tractors, and pair can be treated as two single cards. The winner of this trick is the one who plays the highest-ordered "honor card". Similarly, if more than one player played the same highest-ordered "honor card" the winner of this trick will be the one who plays the highest-ordered "honor card" first.

There are many hair-raising rules about lead and follow cards; fortunately, they're not related to this problem, the only thing we care about is: when someone leads a "not-trump" "throw" the only possible way to beat it is to "throw" the same structure of "trumps". And it's impossible to beat a leading "trump" "throw".

Special attention on the examples below. In these examples, Alice always leads

cards. And assume in all the following examples, the CR is 7, and the main suit is H.

Alice	Bob	Charles	David	Winner	Comments
SA	S2	ST	S5	Alice	Alice plays the highest-ranked card SA.
SA	S2	ST	SA	Alice	Alice plays the first SA.
SA	S2	ST	H2	David	David plays the first only "trump".
SA	H2	C7	D7	Charles	Charles plays "trump" C7, while David plays D7 with the same order of C7.
C2C2	C3C4	C7D7	RJBJ	Alice	The structure of this trick is "pair", while all players except Alice play two single cards.
D3D3	DTDT	SKSK	H2H3	Bob	Bob plays a pair with order higher than Alice's, while Charles discard a pair with wrong suit.
D3D3	DTDT	SKSK	H2H2	David	David plays the only "trump" pair.
D6D6D8D8	DJDJDKDK	DTDTD2D3	HTHTBJBJ	Alice	Alice plays the only tractor (because the CR is 7).
H6H6H8H8	H7H7BJBJ	C2C2C3C4	HKHKRJRJ	Bob	Bob's tractor is higher-ordered than Alices'.
H6H6H8H8	H7H7D7D7	C2C2C3C4	HKHKRJRJ	Bob	Bob also plays a tractor!

SASK	STST	C2H3	S7SK	Alice	Alice makes a throw.
SASK	HKH3	HAH2	S7SK	Charles	Both Bob and Charles can beat Alice.
SASK	HAH2	HAH3	S7SK	Bob	Both Bob and Charles can beat Alice, but Bob's HA comes first.
S2S2S3S3SA	H3H3H4H4RJ	D7D7H7H7H2	S7S7SQSJS6	Charles	Both Bob and Charles can beat Alice.

There is a special rule about "hidden cards": if the winner of the last trick of a certain round is a member of FT, then, in addition, the FT gets the sum of the hidden cards' pts, multiplied by 2^{w} (2 to the power of **w**). When the structure of the final trick is not "throw", then **w** is the number of lead cards of the last trick of this round. If the structure is "throw" instead, **w** is the length of the longest components, in the example RJRJBJBJH7H7HQHQHJHJH6H6HA, the **w** is 6 because the length of RJRJBJBJH7H7 (the longest components of the "throw") is 6.

To make the problem easier, you are only to write a single-round tractor game simulator.

[Input]

Multiple test cases, the number of them **T** is given in the very first line. For each test case:

The first line contains the main suit of this round (H, S, C, D, O; O denotes "None" in this round), the dealer of this round, the CR of team 1, the CR of team 2, separated by single spaces. Each of the rest lines contains 4 strings: the lead cards and the cards played by the second, third and last player. In one string, the cards can be given in any order. Each player will play exactly 25 cards in one round.

You may assume the input is always valid.

There is a blank line between consecutive test cases, and a blank line also appears between **T** and the first test case.

[Output]

For each test case:

The first line contains the case number.

The second line contains the pts get by the FT in this round.

If a team wins the whole game after this round, output "Winner: Team **X**"(without quotes, **X** should be either 1 or 2) in the second line. If no team wins, output

the new CR of team 1, the new CR of team 2 after this round, followed by the name of the dealer of the next round, separated by single spaces.

See the example for further details.

Sample Input	Sample Output
1 O Charles 2 2 S6S6S7S7 SASKSJST STS8S4S4 S3S5SJSQ S9S9 H3D3 S3DT SAD3 DA DQ DK D4 SKS8S5S3 RJC2D2H2 C6C8CJD9 H3CKDTD5 H7H7 H6H4 HJHQ H9H9 DJDJ DKH5 D5D4 D6D6 D8D8 C4C3 HTH5 D9D7 C5C5 C6CT H8HQ C7C4 H8 C7 HA HA H2 RJ BJ CK DA BJ C8 HK S2S2C2 CQCAD2 HTHJHK C9CQCA	Case #1: 50 3 2 Alice

Problem H: Heroes Arrangement

[Description]

There are **N** heroes in the Kingdom of Heroes, each hero has a special range of activity, this "range" is a delta-shaped region (triangle region including the boundary; it is guaranteed that all triangles will neither degenerate into a segment nor a point using the King's angle of view), and heroes can appear in any point in his activity range. The king, standing at the Origin (0, 0, 0), is observing the heroes. No range of activity will contain the king's position.

You may assume that no hero could meet others, that means these "range" have no common point. Your task is to find number **K**, indicating that the king can choose at most **K** heroes such that no pairs of chosen heroes cause one may block the king's view line to observe another. In other words, no hero can appear on the segment between another hero and the King.

[Input]

There are multiple test cases, the number of them, **T**, is given in the very first line of the input, followed by **T** cases.

For each test case:

First line contains an integer **N**, the number of heroes, $1 \le N \le 40$. Then **N** lines follow, each line contains nine integers **x1 y1 z1 x2 y2 z2 x3 y3 z3**, denoting the coordinates of the delta-shaped regions' vertex respectively, $-100 \le x1, y1, z1, x2, y2, z2, x3, y3, z3 \le 100$.

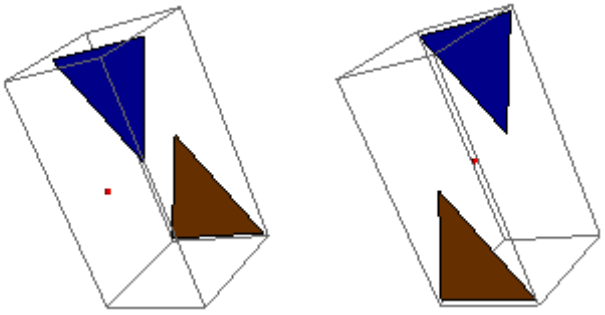
[Output]

Output a single line with a number **K** with case number, the maximum number of heroes king can choose.

Sample Input	Sample Output
2 2 0 0 1 1 2 1 -1 2 1 0 0 2 1 -2 2 -1 -2 2 2 0 0 1 1 2 1 -1 2 1 0 0 -1 1 -2 -1 -1 -2 -1	Case #1: 1 Case #2: 2

[Hint]

The sample image of the sample input is drawn below, the small sphere is the position of the king.



Problem I: Island Explorer

[Description]

A group of explorers has found a solitary island. They land on the island and explore it along a straight line. They build a lot of campsites while they advance. So the campsites are laid on the line.

Coincidently, another group of explorers land on the island at the same time. They also build several campsites along another straight line. Now the explorers meet at the island and they decide to connect all the campsites with telegraph line so that they can communicate with each other wherever they are.

Simply building segments that connect a campsite to another is quite easy, but the telegraph line is rare. So they decide to connect all the campsites with as less telegraph line as possible. Two campsites are connected if they are directly connected with telegraph line or they are both connected to another campsite.

[Input]

There are multiple test cases.

The number of the test cases is in the first line of the input.

For each test case, first line contains two integers **N** and **M** (0≤**N, M**≤10000), which **N** is the number of the campsites of the first group of explorers and **M** is the number of the campsites of the second group of explorers. And there exist at least one campsite.

The next two lines contain eight integers **Ax, Ay, Bx, By, Cx, Cy, Dx, Dy**. Their absolute values are less than 1000. The integers are the coordinates of four points **A, B, C** and **D**. The exploring path of the first group is begin with the first point **A** and end with the second point **B**, and the path of the second group is from the third point **C** to the fourth point **D**. Every pair of points is distinct.

The last two lines of the test case contain N and M real numbers; they indicate the positions of the campsites. Suppose the i-th real number in the first line is **t**. It means the x-coordinate of the i-th campsite is **Ax * t + Bx * (1-t)**, and the y-coordinate is **Ay * t + By * (1-t)**. Equally, the campsite on the second straight line is **C * t + D * (1-t)**. You can assume that there are at most four digits in the decimal part, and the numbers are always between 0 and 1.

[Output]

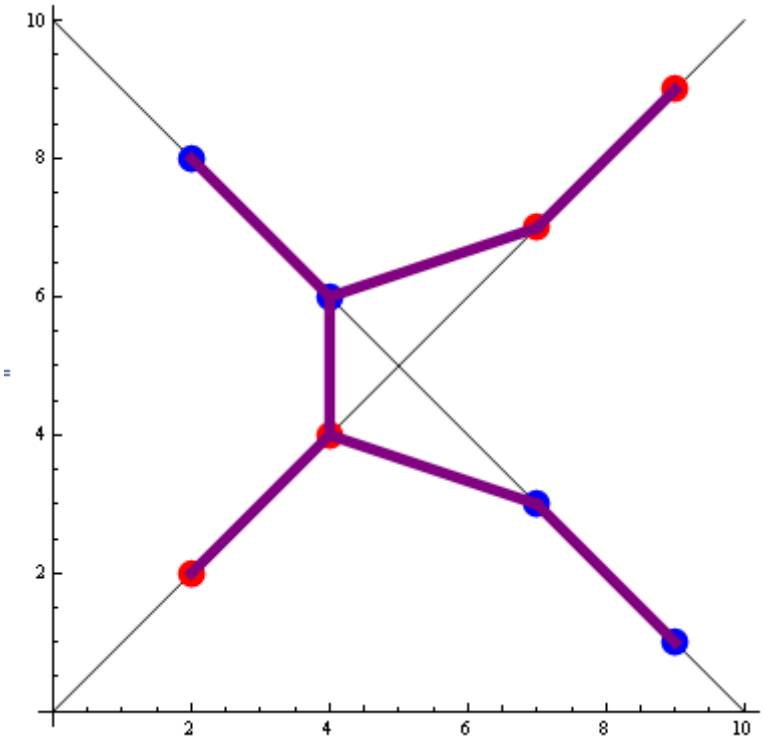
For each test case, output contains only a real number rounded to 0.001.

Sample Input	Sample Output
1 4 4 0 0 10 10 0 10 10 0	Case #1: 19.638

0.1 0.3 0.6 0.8 0.1 0.3 0.6 0.8	

[Hint]

The graph below shows the solution of the sample test.



Problem J: Jinyuetuan Puzzle

[Description]

JinYueTuan is a famous online game which has been in vogue for a long time. Large number of players put themselves in it day after days…

JinYueTuan is a simple game with these rules: Only seven keys on the keyboard will be used in games, each key are assigned to one of the seven sound tracks. During the game, a series of notes may fall in each sound track irregularly. When notes fall down in one sound track, player should press the assigned key at once. If so, then got a "Cool", or get "Miss" otherwise. There are two types of "Note": "single note" and "strip note". Just as the name implies, "single note" means a single note, press right key at the right time once, then you can get a "Cool", otherwise "Miss"; "strip note" needs you keeping press the corresponding key from the start time to the end time of the note. Press key at the begin time you may get a "Cool", and release at the end time to get another "Cool", you will "Miss" either early or late. If "Miss" at the begin time, then "Miss" the other (the end time) immediately.

People lament about the fact that such an easy game is limited by the hardware design, the so-called "keys confliction". "keys confliction" means some key combinations cannot be pressed. For example, when Track 1,4,5,6 fall a note at the same time respectively, we should press the four keys at the same time, but now, keys confliction may happen, and cause this to fail, then you will "Miss" all these four notes. Keys confliction meets the relation of inclusion, in other words, if keys set S would cause the confliction, so would keys set T when S is included in T. Note that, if at a certain moment in time, some keys should be pressed and some pressing keys should be released, then you can release and press at the same time, and released keys won't cause any keys confliction at all.

Because of the keys confliction, we have to give up some notes to save as more other notes as possible. More details will be explained in the sample data. Now you know the falling time and type (single or strip) of each note in each sound track, and all keys-assembling that may cause confliction. Your task is to calculate the maximum number of "Cool" that player can get.

[Input]

There are multiple test cases, the number of them **T** is given in the very first line, followed by **T** cases.

For each test case:

First line contains an integer **N**, the length of the music (**1≤N≤1000**).

The next 7 lines will contain description of each sound track. The first number in each line, **C**, denote the number of notes in that sound track. Following **C** description of each note: A single number "**A**" denotes an single note at Time **A**(**1≤A≤N**); A pair of integers separating by '-',"**A-B**" denotes a strip note start at Time **A** and end at Time **B**(**1≤A<B≤N**). In each sound track, no superposition exists, no strip starts at another's end time, no single notes puts in a strip(includes begin and end point).

The next line contain a single integer **K** (**0≤K<2**^{7}), denotes **K** descriptions of Keys Confliction are following.

Each description is a single line containing a string of seven characters **S0S1S2…S6**, if **Sk1,Sk2,…,Sks** were character '1' and others were '0', that means key combination **Sk1Sk2…Sks** may cause Keys Confliction.

[Output]

For each test case, output one line containing integer **P** with case number, the maximum number of "Cool" players can get.

Sample Input	Sample Output
3 6 3 1 3 4 2 1 2-5 2 3 6 0 0 0 0 1 1110000 6 3 1 3 5 2 1 2-5 2 5 6 0 0 0 0 1 1110000 6	Case #1: 7 Case #2: 8 Case #3: 3

1 1 1 1-3 1 1 0 0 0 0 1 1110000	