Problem A. Boxes and Balls

Little Tom's friend Jack just showed him a great magic trick. At the beginning of the trick, there is one box on the ground with some number of balls in it. Jack then performs this operation over and over again:

1. put a new empty box down on the ground
2. move one ball from each other box into that new empty box
3. remove any boxes that are now empty
4. sort the boxes in nondecreasing order by the number of balls in them

Tom noticed that it is possible for this operation to leave the state of the boxes and balls unchanged! For example:

- Suppose that at the beginning of the trick, the one box contains 3 balls.
- In the first operation, Jack adds a new empty box, puts 1 ball from the existing box into it, and sorts the boxes, so after that operation, there will be 2 boxes on the ground, one with 1 ball and one with 2 balls.
- In the second operation, Jack adds a new empty box and puts 1 ball from each of the existing 2 boxes into it; this creates one empty box, which Jack removes, and then he sorts the boxes. So there are 2 boxes on the ground, one with 1 ball and one with 2 balls. But this is exactly the state that was present before the second operation!

Tom thought about the trick some more, and realized that for some numbers of balls, it is not possible for the operation to leave the state unchanged. For example, if there are 2 balls at the beginning, then after one operation, there will be two boxes with 1 ball each, and after 2 operations, there will be one box with 2 balls, and so on, alternating between these two states forever.

Tom looked around in his room and found infinitely many empty boxes, but only N balls. What is the maximum number of those balls that he could use to perform this trick, such that one operation leaves the state unchanged?

Input

The first line of the input gives the number of test cases, T. T lines follow.

Each line consist of one integer N, the number of balls Tom could find.

Output

For each test case, output one line containing "`Case #x: y`", where `x` is the test case number (starting from 1) and `y` is the maximum number of balls that Tom could use to perform the trick, as described above.

Limits

- $1 \leq T \leq 100$.
- $1 \leq N \leq 10^{18}$.

Sample input and output

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

Note

The trick can be performed with 1 ball or 3 balls, but not with 2 balls. So, for Case #2, Tom can use at most 1 of the 2 balls.

Problem B. Business Cycle

You just saw a TV commercial for an interesting business plan that consists of a cycle of N phases. The i^{th} phase has a value V_i, which can be positive, negative, or zero. When you enter the i^{th} phase, you add V_i to the amount of money you currently have, unless this would create a negative number, in which case your amount of money becomes 0 instead. Then you move on to the $((i+1) \bmod N)^{th}$ phase.

You start the business plan just before phase 0, with some initial amount of money (possibly 0). What is the minimum initial amount of money you need to guarantee that you will have **at least** G money after no more than P phases?

Input

The first line of the input gives the number of test cases, T; this is followed by one blank line.

T test cases follow; each consists of two lines, followed by a blank line. The first line has three integers N, G, and P, as described above. The second line has N integers; the i^{th} of these is V_i, representing the value of the ith phase.

Output

For each test case, output one line containing "`Case #x: y`", where `x` is the test case number (starting from 1) and `y` is the minimum initial amount of money needed, as described above.

Limits

- $1 \leq T \leq 100$.
- $1 \leq N \leq 10^5$.
- $1 \leq G, P \leq 10^{18}$.
- $|V_i| \leq 10^9$.

Sample input and output

Sample Input	Sample Output
3 2 10 2 3 -1 2 10 3 3 -1 1 10 10 -999	Case #1: 7 Case #2: 5 Case #3: 10

Note

In Case #1, if you start with 7 money, then after one phase (phase #0), you will have 10 money. This satisfies the condition of having at least 10 money after no more than 2 phases. No smaller amount works.

In Case #2, if you start with 5 money, then you will have 8 money after one phase (phase #0), 7 money after two phases (phase #1), and 10 money after three phases (phase #0 again). No smaller amount works.

In Case #3 (which is a terrible business plan, by the way!), the only way to have at least 10 money at any point in the process is to start with at least 10 money, and 10 is the smallest amount that works.

Problem C. Suffixes and Palindromes

At a certain algorithm cram school, the students do the following drill each day to prepare for contests:

1. Choose a string of length N made up of lowercase English letters. For example, the string might be `aba`.
2. Find all of the suffixes of the string starting from positions 0, 1, $\cdots$, $N-1$, and sort them in lexicographic order. In our example, the suffixes are `aba`, `ba`, and `a`, and the lexicographically sorted list is `a`, `aba`, `ba`.
3. Go through the sorted list in order, and write down the starting point of each suffix in the original string. In our example, the suffix `a`, which is first in the sorted list, was formed by starting from position 2 in `aba`. The suffix `aba` started from position 0; the suffix `ba` started from position 1. So the students would write down `2 0 1`, in that order.
4. Using the original string, look at every character or every position between two consecutive characters, in order. Try to use each position as the center of a palindrome, and find the length of **longest** palindrome with that center, or 0 if there are none. Then write those numbers down. In our example, the students would first try to use the first a, then the position between the first `a` and first `b`, then the first `b`, then the position between the first `b` and the second `a`, then the second a, and they would write down `1 0 3 0 1`, in that order.

You belong to a rival algorithm school, and you sneaked into this school late at night. You found a student's desk and saw the two lists from steps 3 and 4. Can you figure out what the original string was, and show off by writing it down? If there are multiple choices, use the **lexicographically smallest** one. If there are no choices, the student must have made a mistake, and you should write Wrong calculation!

Input

The first line of the input gives the number of test cases, T, followed by a blank line.

T test cases follow; each consists of three lines followed by one blank line. The first line has N, the length of the string. The next line has N integers: a permutation of the integers from 0 to $N-1$, inclusive, as described in Step 3. The third line has $2 \times N - 1$ integers, as described in Step 4.

Output

For each test case, output one line containing "`Case #x: y`", where `x` is the test case number (starting from 1), and `y` is the lexicographical smallest string that could have produced the two lists. If there is no such string, output "`Wrong calculation!`" instead.

Limits

- $1 \leq T \leq 100$.
- $2 \leq N \leq 10^5$.

- All numbers in the second list will be between 0 and N, inclusive.

Sample input and output

Sample Input	Sample Output
2 3 2 0 1 1 0 3 0 1 3 2 0 1 1 2 3 2 1	Case #1: aba Case #2: Wrong calculation!

Note

Sample case #1 matches the example in the problem statement. Other strings (such as `bcb`) match the two lists, but `aba` is the only acceptable answer because it is lexicographically smallest.

In sample case #2, the palindrome data implies that all of the characters in the original string must be the same, but this doesn't agree with the suffix data. So the student must have made a mistake.

Problem D. Change

You currently have only a banknote/coin of A `CNY`. (`CNY` stands for Chinese yuan, or renminbi, the currency unit in People's Republic of China). You will need to pay exactly B `CNY` at some place soon. The only option you have now is to spend money at a vending machine, which has an infinite number of copies of merchandise of all possible positive prices. You have as much time as you want to use the vending machine. However, you cannot make any assumptions about the way the vending machine will make change; it is only guaranteed that it will dispense banknotes/coins that sum to the amount of change required.

What is the smallest total amount of money you will need to spend at the vending machine to make sure that you will be able to pay exactly B `CNY` later using the change? I.e. what's the smallest total amount you have to spend in the worst case, to get a banknote/coin of B `CNY`, or some smaller banknotes/coins that can combine to exactly B `CNY`?

In this problem we assume all denominations of Chinese yuan are in use, including 1 `fen` (0.01 `CNY`), 2 `fen`, 5 `fen`, 1 `jiao` (0.1 `CNY`), 2 `jiao`, 5 `jiao`, 1 `yuan` (1 `CNY`), 2 `yuan`, 5 `yuan`, 10 `yuan`, 20 `yuan`, 50 `yuan` and 100 `yuan`. These are the only possible denominations that the vending machine will accept or return.

Input

The first line of the input gives the number of test cases, T. T lines follow.

Each line contains 2 numbers A and B.

Output

For each test case, output one line containing "`Case #x: y`", where `x` is the test case number (starting from 1) and `y` is the minimum amount in `CNY` you have to spend when you have only a banknote/coin of A `CNY` and need to pay exactly B `CNY`.

Limits

- $1 \leq T \leq 78$.
- $A, B \in \{0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1, 2, 5, 10, 20, 50, 100\}$ and $A > B$.

Sample input and output

Sample Input	Sample Output
2 0.05 0.02 2 1	Case #1: 0.01 Case #2: 0.01

Note

In Case #1, the optimal solution is to buy a product that costs 1 fen. The machine might return any of the following: two 2 `fen`; one 2 `fen` and two 1 `fen`; four 1 `fen`. In all of those cases, you either have the 2 `fen` that you need, or you can combine two 1 `fen` to create the exact 2 `fen` amount.

Problem E. Colorful Floor

You are going to build a tile mosaic floor on an alien planet. You will choose a certain rectangular pattern (with R rows and C columns) of unit square tiles, and then you will choose a color for each tile from the K colors that exist on that planet; they are numbered 0 through $K-1$. You will tessellate the pattern infinitely in the horizontal and vertical directions. (That is, in the finished floor, if you choose a tile of a certain color, and you move any multiple of R cells up or down or any multiple of C cells left or right, you will end up on another tile of the same color.)

When an alien is happy, it perceives colors exactly as they are. When an alien is angry, though, it sees the i^{th} color as the P_i^{th} color, where P is a permutation of 0, 1, $\cdots$, $K-1$. (It is possible for P_i to equal i.) To avoid confusion, you must choose a pattern that is unambiguous, meaning that happy and angry aliens will see the same overall design when looking at the floor. That is, no matter where on the floor an alien in one emotional state stands, there must be another cell where an alien of the other emotional state can stand, facing in the same direction, such that they both see exactly the same floor.

For example, suppose that the floor will consist of tiles of two colors, black and red, and that an angry alien sees black as red, and red as black. Then an all black floor would be ambiguous, since happy aliens would think it was all black, while angry aliens would think it was all red. However, a checkerboard pattern would be unambiguous, because both types of aliens would see the same overall design.

Find the number of different unambiguous patterns, modulo $1,000,000,007$ (10^9+7). Two patterns are considered the same if the floors paved by them can be matched after horizontal and/or vertical translations only (i.e., reflections and rotations are not allowed).

Input

The first line of the input gives the number of test cases, T. T test cases follow; each consists of two lines.

he first line has three integers K, R, and C, as described above. The second line has K integers; the i^{th} of these is P_i, representing the color perception permutation.

Output

For each test case, output one line containing "`Case #x: y`", where `x` is the test case number (starting from 1) and `y` is the number of different unambiguous patterns, modulo $1,000,000,007$ (10^9+7).

Limits

- $1 \leq T \leq 100$.
- $2 \leq K \leq 10^4$.
- $1 \leq R, C \leq 10^6$.

Sample input and output

Sample Input	Sample Output
6 2 1 2 1 0 2 2 2 1 0 3 2 2 1 2 0 3 1 3 1 2 0 3 3 3 0 1 2 3 3 3 0 2 1	Case #1: 1 Case #2: 3 Case #3: 0 Case #4: 2 Case #5: 2211 Case #6: 1

Note

In Case #1, the only unambiguous pattern has one cell each of the two different colors.

In Case #2, the three different unambiguous patterns are:

```
00  01  01
11  01  10
```

Any other pattern would either be ambiguous, or identical (under our definition above) to one of these three patterns.

In Case #4, the two different unambiguous patterns are `012` and `021`.

Problem F. Hungry Game of Ants

It's time for a hungry game! But, don't worry – it's only for ants.

At the start of this hungry game, there are N ants on a stick of length $N + 1$. The ants are numbered from 1 to N, and the i^{th} ant has a weight of i and stands i unit(s) from the left end of the stick. Note that the N^{th} ant is N units from the left end of the stick and 1 unit from the right end of the stick.

When the game begins, each ant selects left or right with equal probability and starts moving in that direction. All ants always move at the same speed throughout the game. Whenever an ant reaches either end of the stick, it will instantly change direction.

Whenever two ants meet, they fight, and the one with heavier weight wins. (If they are of equal weight, the one coming from the left wins.) Then, the winner eats the loser, which makes its weight permanently increase by the loser's weight. The winner then keeps moving in the direction it was moving before the fight. (This entire process happens instantly.)

The games continue until only one ant remains, and that ant is the winner!

Since each of the N ants can begin by moving left or right, there are 2^N possible scenarios. In how many of these scenarios will the K^{th} ant be the winner? Find this value modulo $1,000,000,007$ $(10^9 + 7)$.

Input

The first line of the input gives the number of test cases, T. T lines follow.

Each line consist of 2 integers N and K, the number of ants in this game and the index (starting from 1) of the ant we are interested in.

Output

For each test case, output one line containing "`Case #x: y`", where `x` is the test case number (starting from 1) and `y` is the number of scenarios, modulo $1,000,000,007$ $(10^9 + 7)$, in which the K^{th} ant will be the winner.

Limits

- $1 \leq T \leq 100$.
- $1 \leq K \leq N$.
- $2 \leq N \leq 10^6$.

Sample input and output

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

Note

In Case #1, there are 2 ants. No matter what directions they go in initially, they will meet and Ant #2 will eat Ant #1. So it is impossible for Ant #1 to be the winner.

In Case #2, there are 3 ants. In any scenario in which Ant #2 initially moves left, it will first encounter and eat Ant #1, increasing its weight to 3. Then it will encounter Ant #3; it will have the same weight as Ant #3, but Ant #2 will win because it is coming from the left. However, in any scenario in which Ant #2 initially moves right, it will first encounter Ant #3 and will be eaten. Since there are 8 scenarios, and Ant #2 initially moves left in half of them, the answer is 4.

Problem G. Legacy of the Void

After joining the Templar, Dark Templar, the Purifier and Taldarim together, Artanis's team is planing to recover Aiur (the homeland of the Protoss) and destroy Amon, the fallen XelNaga. In order to weaken Amon's force, Artanis first has to destroy the Power Structures that power the large psi-matrix under Aiur.

The Power Structure has N nodes, and they are connected by $N-1$ undirected roads, such that it is possible to use the roads to travel from any node to any other node. The nodes are numbered 1 through N. The i^{th} node has a power crystal which holds W_i units of power. When Artanis attacks the i^{th} node, he has probability P_i to successfully destroy the power crystal and make the Power Structure lose all W_i units of power that were contributed by that node. (If he does not destroy the power crystal, nothing happens to that node.)

With the help of Karax, Artanis will run some simulations on a virtual Power Structure in his warship, Spear of Adun, to help him decide on the best strategy. Artanis will run a total of Q simulations. In each simulation, he will select one of the connected blocks in the Power Structure, uniformly at random from the set of all connected blocks. (A block of nodes is connected if any node in that block can be reached from any other node in that block without traveling through any nodes not in that block. As a special case, any single node is also a connected block.) Then, he will try to attack all the nodes in this block, as described above, and the simulation will report the total amount of units of power lost by the Power Structure. (The virtual Power Structure resets between simulations.)

Artanis will have Q integers (each representing a total amount of power lost) after running all of the simulations. To analyze the data, he will sort them in nondecreasing order.

What is the expected value of the K^{th} integer in this sorted array? (K counts starting from 1.)

Input

The first line of the input gives the number of test cases, T. T test cases follow.

Each test case has the following format, in this order:

1. One line with three integers N, Q, and K, as described above.
2. $N-1$ lines. Each of these has two integers U_i and V_i, which means there is one road between node U_i and a **different** node V_i. (All of these pairs of nodes are different.)
3. One line with N integers W_i, each of which gives the number of units of power stored in the crystal of the i^{th} node.
4. One line with N integers P_i, each of which gives the percent chance of success when Artanis attacks the i^{th} node.

Output

For each test case, output one line containing "`Case #x: y`", where `x` is the test case number (starting from 1) and `y` is the expectation of the K^{th} number in the sorted results list. `y` will be considered correct if it is within an absolute or relative error of 10^{-6} of the correct answer.

Limits

- $1 \leq T \leq 30$.
- $1 \leq N \leq 200$.
- $1 \leq Q \leq 50$.
- $1 \leq K \leq Q$.
- $1 \leq U_i, V_i \leq N$.
- $0 \leq W_i \leq 50000$.
- $0 \leq \sum_{i=1}^{N} W_i \leq 50000$.
- $0 \leq P_i \leq 100$.

Sample input and output

Sample Input	Sample Output
3 1 2 2 1 50 3 1 1 3 2 1 2 1 1 1 100 100 100	Case #1: 0.7500000000 Case #2: 1.6666666667

Note

In Case #1, there is only one node, which holds 1 unit of power. An attack on this node will destroy it with 50% probability. Since this one node is the only connected block, every simulation will always select it. The possible outcomes of the two simulations, and the sorted lists of values, are:

- success, success: 1 1
- success, failure: 0 1
- failure, success: 0 1
- failure, failure: 0 0

All of these outcomes are equally likely in this case, so the expected value for position 2 is $\frac{1+1+1+0}{4} = 0.75$.

In Case #2, there are three nodes, connected `1-2-3`, and each holds 1 unit of power and is certain to be destroyed if Artanis attacks it. The connected blocks are: `{1}`, `{2}`, `{3}`, `{1, 2}`, `{2, 3}`, and `{1, 2, 3}`. (Note that `{1, 3}` is **NOT** a connected block since you cannot get from 1 to 3 without using a node not in the block.) The simulation will choose one of these uniformly at random. Since the numbers of units of power destroyed in each, respectively, are `1`, `1`, `1`, `2`, `2`, and `3`, the expected value of the first (and only) element of the list is $\frac{10}{6}$.

Problem H. Open Face Chinese Poker

Decks of Cards

In a standard 52-card deck of playing cards, every card has *a rank* and *a suit*. The thirteen possible ranks, in order from weakest to strongest, are `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9`, `10`, `Jack`, `Queen`, `King`, and `Ace`. We will abbreviate the last five of those as `T`, `J`, `Q`, `K`, and `A`. The four suits are clubs, diamonds, hearts, and spades, which we will abbreviate as `C`, `D`, `H`, `S`. A card is represented by its rank followed by its suit, so the 52 cards in the deck are:

2C	3C	4C	5C	6C	7C	8C	9C	TC	JC	QC	KC	AC
2D	3D	4D	5D	6D	7D	8D	9D	TD	JD	QD	KD	AD
2H	3H	4H	5H	6H	7H	8H	9H	TH	JH	QH	KH	AH
2S	3S	4S	5S	6S	7S	8S	9S	TS	JS	QS	KS	AS

Five Card Hands

In the game of Open Face Poker, here are the possible ***five card hands*** that can be made with five cards, in order from strongest to weakest. If a set of cards matches more than one of these, take the strongest hand. For example, `7H 8H 9H TH JH` matches the definitions of *Straight Flush*, *Flush*, *Straight*, and *High Card*, but it is considered a *Straight Flush*.

Royal Flush: `T`, `J`, `Q`, `K`, and `A` of the same suit. Example: `TD JD QD KD AD`.

Straight Flush: Five consecutive cards of the same suit. (In Straight Flush, `A` can be considered as `1` (a card weaker than `2`).) Examples: `7H 8H 9H TH JH`; `AC 2C 3C 4C 5C`.

Four of a Kind: Four cards of the same rank, plus any other card. Example: `8C 8D 8H 8S JD`.

Full House: Three cards of one rank and two cards of another rank. Example: `3C 3H 3S 5C 5D`.

Flush: Five cards of the same suit. Example: `4D 7D 8D QD KD`.

Straight: Five cards with consecutive ranks. (In Straight, `A` is a special card that can be considered as `A` (a card stronger than `K`) or 1 (a card weaker than 2).) Example: `6C 7D 8C 9H TH`.

Three of a Kind: Three cards of the same rank, plus any two other different cards. Example: `JC JH JS 2H KH`.

Two Pair: Two cards of one rank and two cards of another rank, plus any other different card. Example: `8C 8H AC AD TH`.

Pair: Two cards of one rank, plus any other three different cards. Example: `9H 9S 3D 4D QS`.

High Card: Any hand that matches none of the above. Example: `3S 5C 6C 7C TS`.

Three Card Hands

Here are the possible three card hands that can be made with three cards, in order from strongest to weakest:

Three of a Kind: Three cards of the same rank. Example: `TD TH TS`.

Pair: Two cards of the same rank. Example: `5C 5S 8H`.

High Card: Any hand that matches neither of the above. Example: `JC QC KC`.

Note: a three card hand cannot make a *Straight Flush*, *Flush*, or *Straight*, even though the three cards might be of the same suit and/or consecutive.

Comparing Hands

When comparing two hands, the hand with the stronger type wins. If they have the same type, you must use tiebreaker rules.

Two "Royal Flush"es: The hands tie.

Two "Straight Flush"es: Look only at the final card in each consecutive sequence. The hand with the card with the highest rank wins. If they have the same highest rank, they tie. Note that this means that a `23456` straight flush beats an `A2345` straight flush, since `A` is regarded as 1 here.

Two "Four of a Kind"s: The hand with the higher rank making up the four of a kind wins.

Two "Full House"s: Compare the three of a kinds within the hands. The hand with the higher rank making up the three of a kind wins.

Two "Flush"es: Sort each hand by rank. Compare the highest-ranking cards; if one hands highest-ranking card beats the other, that hand wins. If tie, compare the second-highest-ranking cards, and so on. If all five cards tie, the hands tie.

Two "Straight"s: Look only at the final card in each consecutive sequence. The hand with the card with the highest rank wins. If they have the same highest rank, they tie.

Two "Three of a Kind"s: The hand with the higher rank making up the three of a kind wins.

Two "Two Pair"s: Compare the highest-ranking pair from each hand, then the lowest-ranking pair from each hand, then the remaining cards, then declare a tie.

Two "Pair"s: Compare the ranks of the pairs, then the ranks of the highest remaining cards, then the ranks of the next highest remaining cards, then the ranks of the final remaining cards, then declare a tie.

When comparing a five-card "pair" hand to a three-card "pair" hand, if the pair and the highest remaining card tie, then the five-card hand wins.

Two "High Cards"s: Sort each hand by rank, then compare the highest-ranked card from each hand, then the next-highest-ranked, and so on.

When comparing a five-card "high card" hand to a three-card "high card" hand, if the three highest ranked cards tie, then the five-card hand wins.

Open Face Chinese Poker

In our variant of Open Face Chinese Poker, the player is given fourteen different cards from the deck. The player must discard one card and then rearrange the remaining thirteen into one three-card *front hand*, a five-card *middle hand*, and a five-card *back hand*. The middle hand must beat or tie with the front hand, and the back hand must beat or tie with the middle hand.

For example, this would not be a legal set of hands:

Front: `7D 7S 8H`
Middle: `2H 2D 5C 5S 6H`
Back: `2C 2S 5H 5D 4D`

because the back hand does not beat or tie the middle hand.

The player earns separate scores for the front, middle, and back hands as follows. If none of the scoring options for a type of hand are matched, then the player earns 0 points for that hand. The players total score is the sum of the scores for the three hands.

Front Hand Scoring

Pair

66	77	88	99	TT	JJ	QQ	KK	AA
1	2	3	4	5	6	7	8	9

Three of a kind

222	333	444	555	666	777	888	999	TTT	JJJ	QQQ	KKK	AAA
10	11	12	13	14	15	16	17	18	19	20	21	22

Middle Hand Scoring

Three of a Kind	Straight	Flush	Full House	Four of a Kind	Straight Flush	Royal Flush
2	4	8	12	20	30	50

(Except for three of a kind, these values are twice the values for back hand scoring, below.)

Back Hand Scoring

Straight	Flush	Full House	Four of a Kind	Straight Flush	Royal Flush
2	4	6	10	15	25

These point values do not determine which hands beat which other hands. They are only used to determine players' final scores.

This is a legal set of hands:

Front: `7D 7S 8H`
Middle: `2H 2D 5C 5S 4D`
Back: `2C 2S 5H 5D 6H`

and it scores 2 points for the front, 0 points for the middle, and 0 points for the back, for a total of 2 points.

This is another legal set of hands:

Front: `9C 9D 9S`
Middle: `TS JS QS KS AS`
Back: `TH JH QH KH AH`

and it scores 17 points for the front, 50 points for the middle, and 25 points for the back, for a total of 92 points.

Given a set of fourteen different cards (remember, you must discard one), what is the greatest number of points you can earn, if you choose the hands optimally? Of course, the hands must be a legal set of hands.

Input

The first line contain an integer T, indicating the number of test cases. T lines follow.

Each line contains 14 space-separated descriptions of cards. Each card is represented by two characters. The first character is the rank and second character is the suit, as described in the Decks of Cards section above.

Output

For each test case, output one line containing "`Case #x: y`", where x is the test case number (starting from 1) and y the maximum score you can get.

Limits

- $1 \leq T \leq 100$.
- all cards in a case are distinct.

Sample input and output

Sample Input	Sample Output
1 9C 9D 9S 9H TS TH JS JH QS QH KS KH AS AH	Case #1: 92

Note

The hand with maximum score is the one described above, i.e:

Front: `9C 9D 9S`
Middle: `TS JS QS KS AS`
Back: `TH JH QH KH AH`

and it scores 17 points for the front, 50 points for the middle, and 25 points for the back, for a total of 92 points.

Problem I. Champions League

Every year, 32 top football clubs from different European countries get together to compete for the best of best football club. This is the famous UEFA (Union of European Football Associations) Champions League.

During the group stage of this tournament, the 32 teams will be assigned into 8 groups, marked from `A` to `H`, and each group contains 4 teams. The teams within each group will compete with each other and only 2 of them will advance to the knockout phase. UEFA don't want the top teams to compete with each other too early, so they have divided the teams into 4 levels (from level 1 to level 4) with 8 teams each. The teams in the same level will be assigned to different groups, thus each group will have exactly one team from each level. The group assignment is done by levels: UEFA will first assign the 8 teams from level 1 into different groups, then the teams from level 2, and so on. For each level, the teams are assigned one by one: each time an unassigned team is randomly picked and it will be assigned into a random group which it can be assigned based on all the requirements below.

Since each country has at most 5 teams advanced to the Champions League, UEFA don't want the teams from the same country to fight with each other too early, teams from the same country should not be assigned to the same group.

Each round of the games will be played on two different days: groups `A-D` play on the first day and groups `E-H` play on the second day. To make the broadcast schedule more friendly, the matches that involve teams from the same country should be distributed as evenly as possible. To meet this requirement, when a specific team T is about to be assigned, UEFA will first check the teams that have already been assigned from T's country. If the country has the same number of assigned teams in groups `A-D` as groups E-H, they can assign team T to either day; otherwise they have to assign team T to a group from the day with fewer teams.

As there are so many rules, you know what we want to ask: how many different ways can UEFA assign these teams? Two assignments are considered different if one or more teams are assigned to different groups.

Input

The first line of the input gives the number of test cases, T. T test cases follow.

Each case contains four lines which represents level 1 to level 4. Each line contains 8 teams in the corresponding level. Each team is represented as "`X`", where `X` is the country where this team comes from.

Output

For each test case, output one line containing "`Case #x: y`", where `x` is the test case number (starting from 1) and `y` is the number of ways UEFA could assign teams.

Limits

- $1 \leq T \leq 20$.
- `X` only contains three upper-case letters.

Sample input and output

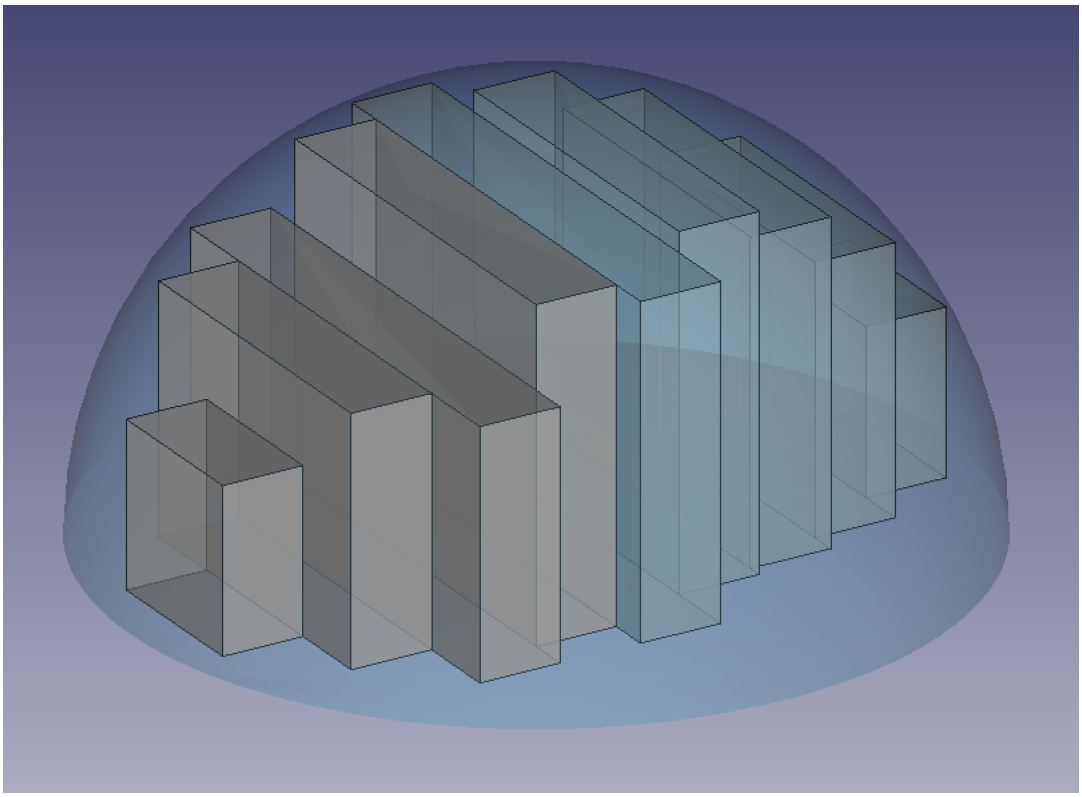
Sample Input	Sample Output
1 ESP GER ENG ITA POR FRA RUS NED ESP ESP POR ENG ENG ESP ENG GER UKR ESP FRA UKR GRE RUS TUR ITA BLS GER GER CRO ISR BEL SWE KAZ	Case #1: 1370850443919360

Note

Sample is the real data of Champion League 2015.

Problem J. Dome and Steles

Archaeologists recently discovered a dome built by an ancient civilization. Its inner surface was a perfect hemisphere, and many steles were stored under the dome. Those steles are cuboids, and they all have the same thickness. They were placed vertically on the flat ground and parallel to each other, as shown in the picture below. Note that, assuming that your line of sight is parallel to the steles (viewing from the lower right of the picture), no stele is allowed to block any part of another one.



A war broke out before more information could be retrieved. The dome was completely destroyed, and the steles were scattered around. However, all the steles somehow remained undamaged, and were collected by archaeologists after the war. Now the archaeologists are planning to rebuild the dome, and they want to know the minimum radius of its inner surface.

Input

The first line of the input gives the number of test cases, T. T test cases follow.

Each test case starts with a number N, the number of steles. Each of the next N lines contains 2 float numbers with 4 digits after the decimal point, a_i and b_i, representing two of the dimensions of stele i besides the thickness. All the steles have the same thickness, which is 1. Note that the steles and their dimensions are not in any particular order as their original placements were lost.

Output

For each test case, output one line containing "`Case #x: y`", where `x` is the test case number (starting from 1) and `y` is the minimum radius of the dome. `y` will be considered correct if it is within an absolute or relative error of 10^{-6} of the correct answer.

Limits

- $1 \leq T \leq 100$.
- $1 \leq N \leq 10^5$.
- $1 \leq a_i, b_i \leq 10^5$.

Sample input and output

Sample Input	Sample Output
2 9 2.0000 2.0000 3.0000 4.0000 3.0000 6.0000 4.0000 5.0000 4.0000 6.0000 4.2500 4.2500 3.8800 3.8800 3.2200 3.2200 2.0000 2.0000 3 2.0000 4.0000 2.0000 2.0000 4.0000 2.0000	Case #1: 5.0249378189 Case #2: 3.0000000000

Note

The first test case is illustrated in the picture above, from lower-left to upper-right.

For the second test case, note that the steles were scattered around during the war so their original placements were lost. The archaeologists only want to know the minimum inner radius.

Problem K. Convex Polyhedron

Harold lives on the Tropic of Cancer, and he just noticed that the sun was directly overhead, which is very rare! He was so excited that he took out one of his toys, which was a convex polyhedron, and started playing with it. The sun shone down on the toy and made a shadow of the toy on the flat ground. Harold found that as he rotated the toy in three-dimensional space, the area of the shadow might change. He wondered what the maximal area of this shadow could be.

More formally: given a convex polyhedron, find out the maximal possible area of its projection onto a plane. According to Wikipedia, a convex polyhedron is a three-dimensional solid with flat polygonal faces, straight edges, and sharp corners or vertices, for which the surface (comprising its faces, edges and vertices) does not intersect itself and the line segment joining any two points of the polyhedron is contained in the interior or surface.

Input

The first line of the input gives the number of test cases, T, and is followed by a blank line.

T test cases follow. Each test case starts with one line with an integer N, the number of points on the convex polyhedron. N more lines follow; each line consists of 3 real numbers, X_i, Y_i, and Z_i, representing the coordinates in 3-dimensional space of the i^{th} point of the convex polyhedron. All of these points will be different.

Output

For each test case, output one line containing "`Case #x: y`", where `x` is the test case number (starting from 1) and `y` is the maximal area of the projection. `y` will be considered correct if it is within an absolute or relative error of 10^{-6} of the correct answer.

Limits

- $1 \leq T \leq 100$.
- $4 \leq N \leq 50$.
- All X_i, Y_i, and Z_i will be between -10^9 and 10^9, inclusive.
- The points are guaranteed to form a convex polyhedron.
- There isn't a plane that contains all the points.

Sample input and output

Sample Input	Sample Output
1 4 0.0 0.0 0.0 0.0 0.0 1.0 0.0 1.0 0.0 1.0 0.0 0.0	Case #1: 0.8660254038

Problem L. Multiplication Table

Peter learned about multiplication in school yesterday. He was so excited about it that he stayed up late writing out a very large multiplication table, which looks like this:

1	2	3	4	5	6	7	8	9	10	$\cdots$
2	4	6	8	10	12	14	16	18	20	$\cdots$
3	6	9	12	15	18	21	24	27	30	$\cdots$
4	8	12	16	20	24	28	32	36	40	$\cdots$
5	10	$\cdots$								
6	12	$\cdots$								
7	14	$\cdots$								
$\cdots$										

The value of the j^{th} column of the i^{th} row in this table is $i \times j$ (assuming that rows and columns are indexed starting from 1). You may assume that the table has infinitely many rows and columns.

Last night, even after Peter finally fell asleep, he kept on dreaming about the table. He thought he saw some part of the table as a rectangle with R rows and C columns. (Peter's rectangle did not necessarily start from the first row or first column, but it was in the same orientation as the table.) After he woke up, he wrote down the entire rectangle, using '`?`' for the numbers that he didn't remember.

Now Peter would like to know whether the rectangle of numbers from his dream could have really existed in the table or not. Can you help him determine this?

Input

The first line of the input gives the number of test cases, T. T test cases follow.

Each test case begins with one line with 2 numbers R and C, the numbers of rows and columns in Peter's rectangle.

Then, R more lines follow. Each line consists of C space-separated values, each of which may be either a number or '`?`'. A '`?`' indicates that Peter could not remember the number in that cell.

Output

For each test case, output one line containing "`Case #x: y`", where `x` is the test case number (starting from 1) and `y` is "`Yes`" if it matches part of the table, and "`No`" otherwise.

Limits

- $1 \leq T \leq 100$.
- $1 \leq R, C \leq 1000$.
- All numbers that Peter remembers are between 1 and 10^9.

Sample input and output

Sample Input	Sample Output
5 3 3 4 ? 8 ? 9 ? ? ? ? 3 4 ? ? ? ? ? ? ? ? 3 6 8 12 1 2 1000000000 ? 1 2 ? 1 2 1 ? ?	Case #1: Yes Case #2: No Case #3: Yes Case #4: No Case #5: Yes

Note

In Case #1, the rectangle could match the part of the table starting from the second column of the second row:

$$\begin{matrix} 4 & 6 & 8 \\ 6 & 9 & 12 \\ 8 & 12 & 16 \end{matrix}$$

In Case #2, it is impossible for the rectangle to match part of the table. (In particular, no row of the table contains 3, 6, 8, 12 as consecutive numbers.)

Note that in Case #3, the unknown value must be greater than 10^9 (for example, it could be 2×10^9). The limit of 10^9 is only on the numbers that Peter remembers.

In Case #4, no value can come before 1 in the table.

Problem M. November 11^{th}

It's November 11^{th}, which is Singles' Day! On this day, a certain cinema is only allowing singles to watch movies there. Couples are forbidden!

There are R rows in this cinema, numbered 0, 1, $\cdots$, $R-1$. In each row, there are S seats, numbered 0, 1, $\cdots$, $S-1$.

Singles refuse to sit directly beside each other. Two seats are considered beside each other if they are in the same row and they have consecutive seat numbers.

There are a total of B broken seats in the cinema, and nobody can sit in a broken seat.

The cinema owner has asked you to find two values:

- The maximum possible number of singles that could sit in this cinema
- The minimum number of singles needed to occupy the cinema so that no more singles can sit

Input

The first line of the input gives the number of test cases, T. T test cases follow.

Each test case starts with one empty line and then 2 integers R and S, the number of rows and the number of seats per row.

The next line consists of a number B. Then B lines follow; each has two 2 integers r_i and s_i, indicating that in row r_i, seat s_i is broken. All of the broken seats will be different.

Output

For each test case, output one line containing "`Case #x: y z`", where `x` is the test case number (starting from 1), `y` is the maximum possible number of singles that could sit in this cinema, and `z` is the minimum possible number of singles that could occupy the cinema.

Limits

- $1 \leq T \leq 100$.
- $1 \leq R, S \leq 1000$.
- $0 \leq B \leq 1000$.
- $0 \leq r_i \leq R-1$.
- $0 \leq s_i \leq S-1$.

Sample input and output

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

Note

In Case #1, up to four singles can fit in the cinema:

```
SBS
S.S
```

However, it is possible for three singles to occupy the cinema:

```
SBS
.S.
```