

Problem A. Coding Contest Creation

Input file: `standard input`
Output file: `standard output`
Time limit: 30 seconds
Memory limit: 512 megabytes

You've been put in charge of creating the problems for a certain high-profile programming contest series.

The series will consist of one or more contests of exactly 4 problems each. Every problem has a difficulty rating (an integer between 1 and 100, inclusive), and the ratings of the 4 problems in each contest must be strictly increasing, but with a difference of no more than 10 between consecutive problems. In other words, if the problems in a contest have difficulties a, b, c , and d (in order), then the inequalities $a < b < c < d$, $b - a \leq 10$, $c - b \leq 10$, $d - c \leq 10$ must all hold.

You've been given an ordered list of N problems to use. Being an experienced problemsetter, you may also write some new problems to insert at any positions in the list, each with an integer difficulty between 1 and 100, inclusive. The final list of problems must still include the original N problems in their original order, though (with your new problems optionally mixed in).

Once the problem list is finalized, the first 4 problems (in order) will form a contest, the next 4 problems will form another contest, and so on. Note that the number of problems in the list must be divisible by 4, and that each of the contests formed must feature a valid ordered set of 4 problems. What's the minimum number of additional problems you must write in order to create a set of valid contests?

Input

Input begins with an integer T ($1 \leq T \leq 50$), the number of contest series you need to create. For each series, there is first a line containing the integer N ($1 \leq N \leq 10^5$), then a line containing N space-separated integers, the i -th of which is D_i ($1 \leq D_i \leq 100$), the difficulty rating of the i th existing problem.

Output

For the i -th series, print a line containing "Case #i: " followed by the minimum number of additional problems you'll need to write.

Examples

standard input	standard output
5	Case #1: 0
4	Case #2: 4
10 15 25 30	Case #3: 9
4	Case #4: 4
15 20 25 40	Case #5: 9
3	
3 3 3	
8	
60 90 61 62 63 91 92	
93	
11	
5 14 30 32 39 46 47 47	
30 58 47	

Note

In the first series, the four problems given are already a valid contest, so no new problems need to be written. In the second series, the four existing problems most certainly do not form a valid contest, due to the gap between the third and fourth ones — one possible way to salvage the situation is to add three new

problems with difficulties 30, 29, and 30 after the third problem, as well as a problem with difficulty 42 at the end, creating two valid contests: [15, 20, 25, 30] and [29, 30, 40, 42].

Problem B. Laundro, Matt

Input file: `standard input`
Output file: `standard output`
Time limit: 10 seconds
Memory limit: 512 megabytes

The original problem tests were splitted to contain at most 5 testcases.

Matt Laundro is about to engage in his favourite activity — doing laundry! He's brought L indistinguishable loads of laundry to his local laundromat, which has N washing machines and M dryers. The i -th washing machine takes W_i minutes to wash one load of laundry, and each dryer takes D minutes to dry a load of laundry. At any point in time, each machine may only be processing at most one load of laundry.

As one might expect, Matt wants to wash and then dry each of his L loads of laundry. Each load of laundry will go through the following steps in order:

1. A non-negative amount of time after Matt arrives at the laundromat, Matt places the load in an unoccupied washing machine i
2. W_i minutes later, he removes the load from the washing machine, placing it in a temporary holding basket (which has unlimited space)
3. A non-negative amount of time later, he places the load in an unoccupied dryer
4. D minutes later, he removes the load from the dryer

Matt can instantaneously add laundry to or remove laundry from a machine. Help Matt minimize the amount of time (in minutes after he arrives at the laundromat) after which he can be done drying all L loads of laundry!

Input

Input begins with an integer T ($1 \leq T \leq 5$), the number of times Matt goes to the laundromat. For each trip to the laundromat, there is first a line containing the space-separated integers L ($1 \leq L \leq 10^6$), N ($1 \leq N \leq 10^5$), M ($1 \leq M \leq 10^9$), and D ($1 \leq D \leq 10^9$) in that order. After that is a line containing N space-separated integers, the i -th of which is W_i ($1 \leq W_i \leq 10^9$).

Output

For the i -th trip, print a line containing "Case #i: " followed by the minimum time it will take Matt to finish his laundry.

Examples

standard input	standard output
5	Case #1: 1234
1 1 1 34	Case #2: 12
1200	Case #3: 5
2 3 2 10	Case #4: 22
100 10 1	Case #5: 3003
3 3 3 3	
1 2 3	
4 2 2 7	
5 8	
999 1 999 6	
3	

Note

In the first case, Matt has just one load of laundry. He washes it for 1200 minutes, and dries it for 34 minutes. In the second case, Matt uses the 1-minute washer for both loads of laundry. The second load finishes at the 2-minute mark, so it finishes drying at the 12-minute mark.

Problem C. Yachtzee

Input file: **standard input**
Output file: **standard output**
Time limit: 30 seconds
Memory limit: 512 megabytes

A rich, retired programmer has decided to invest his (possibly) massive savings into constructing custom yachts for himself.

Initially, he has a real amount of dollars, chosen uniformly at random from the range $[A, B]$. Constructing a yacht consists of N sequential steps, with the i th step requiring C_i dollars.

The programmer blindly pays for the steps in order, until he's either completed all of them, or can't afford the cost of the next step. If the former occurs, he puts his completed yacht aside and restarts the process from the first step with his remaining money — he wants as many yachts as possible! Otherwise, in the latter case, he immediately stops his project entirely, without spending any additional money on other steps.

What's the expected amount of money which the programmer will be left with once he stops spending it on yachts? Your output should have at most 10^{-6} absolute or relative error.

Input

Input begins with an integer T ($1 \leq T \leq 50$), the number of times the programmer embarks on a yacht-creation spree. For each spree, there is first a line containing the space-separated integers N ($1 \leq N \leq 10^5$), A , and B ($0 \leq A < B \leq 10^9$) in that order, then a line containing N space-separated integers, the i -th of which is C_i ($1 \leq C_i \leq 10^9$).

Output

For the i th spree, print a line containing "Case #i: " followed by the expected amount of money the programmer will have left.

Examples

standard input	standard output
5	Case #1: 1.166666667
1 5 8	Case #2: 3.500000000
2	Case #3: 4.277777.000
1 0 777777777	Case #4: 3.227272727
7	Case #5: 4.400000000
1 777777 7777777	
777777777	
2 9 20	
8 2	
5 40 140	
4 9 1 12 7	

Note

In the first case, the programmer starts with between 5 and 8 dollars. The programmer's initial amount of money will fall into one of the ranges $[5, 6)$, or $[6, 8)$, or $[8, 8]$. If it falls into the first range, the programmer will end with $[1, 2)$ dollars. If it falls into the second range, the programmer will end with $[0, 2)$ dollars. If the programmer starts with exactly 8 dollars, he'll end with 0 dollars.

Problem D. Boomerang Tournament

Input file: **standard input**
Output file: **standard output**
Time limit: 5 seconds
Memory limit: 512 megabytes

The original problem tests were splitted to contain at most 5 testcases.

This weekend, the long-awaited BIT (Boomerang Invitational Tournament) will be taking place! N of the finest boomerangists will be competing in a randomly-seeded single-elimination bracket.

For those unfamiliar with this tournament format, the process can be modelled as follows:

1. The N competitors are arranged in a queue (an ordered list), in some order
2. If the queue currently contains only 1 competitor, the tournament ends with them as the champion
3. Otherwise, the first 2 competitors in the front of the queue are removed, and they play a match against one another
4. The winner of that match is re-inserted into the queue, at the back
5. Repeat from step 2

The one-on-one matches in this tournament are, of course, boomerang duels to the death. If the i -th and j -th competitors face off against one another, the i th competitor will win if $W_{i,j} = 1$. Otherwise, if $W_{i,j} = 0$, the j -th competitor will win. Note that, for all $(1 \leq i, j \leq N)$, $W_{i,j} = 0$ or 1, and $W_{i,i} = 0$ (no one will play against themselves anyway), and $W_{i,j} \neq W_{j,i}$ (if $i \neq j$). Those are the only constraints. It's possible that, for example, competitor A can beat B , B can beat C , and C can beat A .

Once the tournament is over, each boomerangist is given a placing (even if they didn't survive the competition). A given competitor c 's placing is an integer one greater than the number of competitors who won strictly more matches than c did.

For each boomerangist, you'd like to know both the best (smallest) and the worst (largest) placing they could possibly end up with, given that the initial ordering of the competitors (in step 1 of the tournament) is unknown.

Input

Input begins with an integer T ($1 \leq T \leq 5$), the number of tournaments. For each tournament, there is first a line containing the integer N ($N = 2^K$ where K is an integer and $0 \leq K \leq 4$). Then follow N lines, the i -th of which contains the space-separated integers $W_{i,1}$ through $W_{i,N}$.

Output

For the i th tournament, print a line containing "Case #i: " followed by N lines that each contain two space-separated integers. The first integer on the i -th line should be the best possible placing for the i -th competitor, and the second should be the worst possible placing.

Examples

standard input	standard output
5	Case #1:
1	1 1
0	Case #2:
2	1 1
0 1	2 2
0 0	Case #3:
4	1 1
0 1 1 1	2 3
0 0 1 1	2 3
0 0 0 1	3 3
0 0 0 0	Case #4:
4	3 3
0 0 0 0	1 3
1 0 0 1	1 3
1 1 0 0	1 3
1 0 1 0	Case #5:
8	5 5
0 0 0 0 0 0 0 0	3 5
1 0 1 0 0 0 0 0	3 5
1 0 0 0 0 0 0 0	1 5
1 1 1 0 0 1 1 0	1 5
1 1 1 1 0 1 0 1	2 5
1 1 1 0 0 0 0 1	1 5
1 1 1 0 1 1 0 1	1 5
1 1 1 1 0 0 0 0	

Note

In the second tournament, the first competitor will always beat the second competitor, so the first competitor will finish in 1st place, and the other in 2nd place. In the third tournament, the first competitor never loses, so they will finish in 1st place. The fourth competitor never wins, so they will finish tied for 3rd place with the other competitor who loses their initial match. The other two competitors will either lose their first match (if initially paired with the first competitor) or their second match (if initially paired with the fourth competitor), so they can each finish in 2nd place, or tied for 3rd place.