The 2025 Asia Chengdu Regional Contest

Hosted by UESTC

Problems

A A Lot of Paintings

Time limit: 1s

Panda is a gallery curator. To prepare for an art exhibition, he launched a painting collection activity, and has collected a lot of paintings.

There are n people participating in the activity, numbered from 1 to n. For each person i, the number of paintings submitted is b_i. The total number of paintings is $m = \sum_{i=1}^{n} b_i > 0$. The submission percentage a_i of person i is calculated as $a_i = \text{round}\left(\frac{b_i}{m}, 2\right) \times 100\%$.

The function round$(x, 2)$ rounds the real number x to two decimal places. If the third decimal digit of x is 5 or greater, it rounds up; otherwise, it rounds down. For example, round$(1.14514, 2) = 1.15$, while round$(1.14414, 2) = 1.14$.

One day, Panda woke up to find that all the paintings had been stolen, and he also forgot the total number of paintings, m. He needs your help to restore the possible number of paintings submitted by each participant $b_1, b_2, \ldots, b_n$ using only the recorded submission percentages $a_1, a_2, \ldots a_n$. If no valid submission scheme exists, you must inform him.

In simple terms, given an array of percentages $a = [a_1, a_2, \ldots a_n]$, find a non-negative integer array $b = [b_1, b_2, \ldots, b_n]$ such that $\sum_{i=1}^{n} b_i > 0$, and for all i:

$$a_i = \text{round}\left(\frac{b_i}{\sum_{j=1}^{n} b_j}, 2\right) \times 100\%$$

or determine that no such array exists.

Input

The first line contains an integer T ($1 \le T \le 2 \times 10^5$), indicating the number of test cases.

For each test case, the first line contains an integer n ($1 \le n \le 2 \times 10^5$), representing the number of people participating in the painting activity.

The second line contains n non-negative integers $d_1, d_2, \ldots, d_n$ ($0 \le d_i \le 100$), where $a_i = \frac{d_i}{100} \times 100\%$.

It is guaranteed that the total sum of n across all test cases does not exceed 2×10^5.

Output

For each test case, if there exists a satisfying array b, you should first output a line with `Yes`, followed by a line with n non-negative integers $b_1, b_2, \ldots, b_n$ ($0 \le b_i \le 10^9$, $\sum_{i=1}^{n} b_i > 0$) separated by spaces. Any valid solution is accepted.

If there is no valid solution, simply output a line with `No`. Either `Yes` or `No` is case-insensitive, which means you can print `YeS`, `yEs`, `nO`, etc.

Sample Input 1	Sample Output 1

3
4
25 25 25 25
5
25 25 25 26 0
3
0 0 1
	Yes
	1 1 1 1
	Yes
	25 25 25 26 0
	No

This page is intentionally left blank.

B Blood Memories

Time limit: 3s

Panda is playing a game, in which he is fighting the Demon's army with a party of n characters, numbered 1 to n. Each round begins with m Ap (energy points) available to the party.

Each character i has a skill that deals a_i damage and normally costs c_i Ap. In any round, each character can choose to either use their skill once or do nothing with zero cost. The total Ap cost of all skills used in a round must not be more than m. Any remaining Ap at the end of the round is discarded, and the party is fully refreshed with m Ap for the next round.

Due to the unique mechanics of this game, if a character uses their skill in a round, its Ap cost for the next round becomes $c_i + k$. If the character continues using the skill in consecutive rounds, the cost stays at $c_i + k$ (it does not increase further). If a character does not use their skill in a round, the skill cost will reset to c_i for the very next round.

Panda wants to maximize the total damage dealt over a total of R rounds. Find the maximum possible total damage.

Input

The first line contains a single integer T ($1 \le T \le 100$), denoting the number of test cases.

For each test case, the first line contains four integers n, m, k, R ($1 \le n \le 6$, $1 \le m, k \le 10^3$, $1 \le R \le 10^9$). n is the number of characters in the party, m is the Ap gained at the start of each round, k the temporary Ap cost increase when a skill is used, and R is the total number of rounds.

For the next n lines, each line contains two integers a_i, c_i ($1 \le a_i \le 10^6$, $1 \le c_i \le m$), which are the damage and initial Ap cost for character i.

Output

Output a single integer, denoting the maximum possible total damage that can be achieved.

Sample Input 1

```
3
3 7 1 5
59 3
13 2
81 4
5 14 2 9
66 8
20 2
25 4
39 6
57 7
4 13 7 16
18 2
13 5
33 4
7 1
```

Sample Output 1

```
490
939
741
```

This page is intentionally left blank.

C Crossing River

Time limit: 1s

Panda is a boatman who operates a boat on a river. The boat can carry at most one passenger at a time.

One day, there are n people who want to cross from the left bank of the river to the right bank. Their arrival times at the left riverbank are $a_1, a_2, \ldots, a_n$. There are also m people who want to cross from the right bank to the left bank. Their arrival times at the right riverbank are $b_1, b_2, \ldots, b_m$. A passenger can only board the boat at or after their arrival time.

The boat takes exactly k units of time to cross the river, whether it has a passenger or not. Panda can choose the boat's starting position (left or right bank). The goal is to find a crossing schedule that minimizes the time when the very last of the $n+m$ people reaches the opposite bank destination. You should also help Panda determine the complete crossing schedule.

Input

The first line contains three integers n, m, k ($1 \le n, m \le 10^5$, $1 \le k \le 10^9$), denoting the number of people starting on the left, the number of people starting on the right, and the time for a single crossing.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$), which are the arrival times for the people on the left bank.

The third line contains m integers $b_1, b_2, \ldots, b_m$ ($1 \le b_i \le 10^9$), which are the arrival times for the people on the right bank.

Output

The first line should contain one integer T, representing the minimum possible time the last passenger reaches their destination.

The following $n+m$ lines should describe the crossing schedule **in chronological order**. The i-th line contains three integers t_i, tp_i, id_i ($0 \le t_i \le T$, $tp_i \in \{0,1\}$), indicating that at time t_i, the passenger with index id_i from the left bank ($tp_i = 0$) or the right bank ($tp_i = 1$) gets on the boat. When $tp_i = 0$, you must ensure that $1 \le id_i \le n$ and $t_i \ge a_{id_i}$; when $tp_i = 1$, you must ensure that $1 \le id_i \le m$ and $t_i \ge b_{id_i}$. Moreover, you must ensure that $t_1 < t_2 < \cdots < t_{n+m}$ and $T = \max_i(t_i + k)$.

For a valid schedule, every one of the $n+m$ people must successfully board the boat and reach the opposite bank. The time between two consecutive boarding times must be at least k, that is, $t_i - t_{i-1} \ge k$ for all $1 < i \le n+m$. If two consecutive people board from the same bank ($tp_i = tp_{i-1}$), the time between their boarding must be at least $2k$, that is, $t_i - t_{i-1} \ge 2k$.

Sample Input 1

```
5 5 2
2 1 13 19 11
12 18 19 7 8
```

Sample Output 1

```
25
5 0 2
7 1 4
9 0 1
11 1 5
13 0 5
15 1 1
17 0 3
19 1 2
21 0 4
23 1 3
```

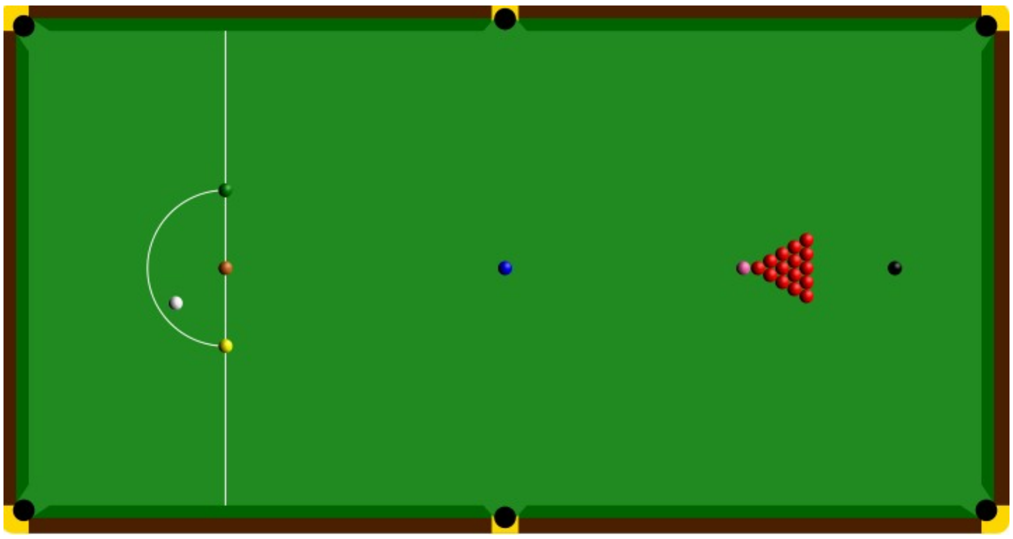
This page is intentionally left blank.

D Deductive Snooker Scoring

Time limit: 1s

Billiards is a sport where players use a cue to strike the white cue ball and pot other balls into pockets. One famous variant is Snooker. With Zhao Xintong winning the Snooker World Championship, this sport has gained more attention in China. The rules of Snooker are quite complex, and this problem focuses primarily on the scoring rules. Note that the Snooker rules described here may differ from actual Snooker rules, so please follow the description here.

First, let's familiarize ourselves with the 21 balls in Snooker. Red balls: Each worth 1 point, with 15 balls in total. Colored balls: yellow (2 points), green (3 points), brown (4 points), blue (5 points), pink (6 points), black (7 points), with only one ball of each color.



We refer to the two players in a frame of Snooker as Player A and Player B. Both start with a score of 0. Player A takes the first turn, and the process begins as follows:

1. If red balls are present, the current player must first pot a red ball. If successful, add one point to his score and proceed to step 2; if unsuccessful, switch to the opponent who starts from step 1. If there are no red balls left on the table, proceed to step 3.
2. After potting a red ball, the current player should select any one of the six colored balls and pot it. If successful, add the corresponding points of the colored ball to his score, then **put the potted colored ball back on the table** (this is the only case where a potted ball is returned to the table; in all other cases, potted balls are not returned to the table), then return to step 1; if unsuccessful, switch to the opponent who starts from step 1. Note that this step must be completed after potting a red ball, even if that red ball was the last one on the table.
3. If there are no red balls left on the table, proceed to the color clearance phase. The current player must pot the colored ball **with the lowest point** remaining on the table. If successful, add the corresponding points to his score and continue with step 3; if unsuccessful, switch to the opponent who starts from step 1 (though in practice, it would directly jump to step 3). If no balls remain on the table, the frame ends.

For convenience, this problem does not consider any fouls or special situations, including but not limited to potting multiple balls simultaneously or potting non-target balls. Only the scenarios mentioned above need to be considered.

Due to overtime work, Panda missed the beginning of a Snooker frame. When he returns home and turns on the TV, he sees that a player is currently at the table. At this moment, there are n balls (red and colored) on the table, where any balls that should be returned to the table have been put back, and the current scores of Players A and B are $a : b$. Panda wants to deduce what might have happened earlier based on the current situation.

Specifically, output any sequence of successful and failed shots (encoded as output format described below) that could lead from the start of the frame to the given current state, or determine that no such sequence exists. If multiple solutions exist, output any one of them.

Input

The first line contains an integer T ($1 \le T \le 2000$), indicating the number of test cases.

Each of the next T lines contains four integers a, b, n, p ($0 \le a, b \le 200$, $0 \le n \le 21$, $0 \le p \le 1$), representing the current scores of Player A and Player B as $a : b$, and the total number of red and colored balls currently on the table is n. If $p = 0$, Player A is currently at the table; if $p = 1$, Player B is currently at the table.

Output

For each test case, output a string representing the sequence of steps, where digits `1` to `7` indicate that the current player potted a ball of the corresponding points, and the character `/` indicates that the player failed to pot a ball, resulting in a switch to the opponent. If no valid sequence exists, output `NA` in a single line. If multiple solutions exist, output any one of them. The length of the output string for each test case should not exceed 100. Do not output extra spaces.

Sample Input 1

```
7
8 110 1 1
147 0 0 0
0 0 21 0
1 0 20 0
1 1 19 1
0 0 0 0
147 0 0 1
```

Sample Output 1

```
/16121516/17/171717171717121712142345 6
171717171717171717171717171717234567

//1//
/1/1/
NA
NA
```

The first test case in the example is the final frame where Zhao Xintong won the Snooker World Championship (Player A is Mark Williams, Player B is Zhao Xintong).

E Escaping from Trap

Time limit: 3s

This is an interactive problem.

Panda, an adventurous child, has accidentally activated a trap while exploring ruins. The trap is a **regular polygon** with N vertices, numbered $1, 2, \ldots, N$ in a counterclockwise direction. The trap's center (i.e., the centroid of the polygon) must be destroyed if he wants to escape from the trap. (A regular polygon is a convex polygon that is both equilateral, meaning all sides have the same length, and equiangular, meaning all interior angles are equal.)

Panda doesn't know the side length of the polygon or his exact location. **He might be inside, on the boundary, or outside the regular polygon**. His only tool is a measurement method: he can choose any two polygon vertices, a and b, and his tool will tell him the area of the triangle formed by his current position and these two vertices. Time is short, and he has **at most** 5 measurement attempts. Your task is to help him calculate the distance from his current position to the polygon's centroid so that he can summon a ring of fire to destroy the trap.

Interaction

This problem contains multiple sets of interactive tests. First, your program should read an integer T from standard input ($1 \le T \le 10^4$), indicating that there are T sets of interactive tests.

At the beginning of each test set, your program should read an integer N from standard input ($4 \le N \le 200$), representing the number of vertices of the regular polygon. It is guaranteed that the circumradius of this regular polygon (the distance from the polygon's centroid to any vertex) is a real number in the range $[1, 100]$, and the distance from Panda's position to the centroid of the polygon (i.e., the answer) will not exceed 10000. All hidden parameters remain fixed during each test case, including the circumradius of the polygon, Panda's position, and the exact coordinates of the polygon's vertices.

For a measurement query, output a line to standard output in the form of `? x y`, where x, y are two integers from 1 to N, indicating the indices of the points on the regular polygon you want to query. The interaction system will return a non-negative real number with at least 10 valid digits, representing the area of the triangle formed by these two vertices and Panda's position. Please note that if your program makes more than 5 queries, the interactor will output a line `-1` and stop the interaction, and your solution will receive `Wrong Answer`.

After at most 5 queries, output a line to standard output in the form of `! d` to give the answer, where d is a non-negative real number representing the distance from Panda to the centroid of the regular polygon. Your output will be considered correct if the absolute or relative error between your output p and the interaction system's computed standard answer q does not exceed 10^{-4}, i.e., $\min\left(\left|\frac{p-q}{q}\right|, |p-q|\right) \le 10^{-4}$. In this case, the interactor will output a line `Correct` and continue to the next set of data for interaction; otherwise, the interactor will output a line `Wrong` and stop the interaction, and your solution will receive `Wrong Answer`.

Note: Each output must be followed by a newline, and the standard output buffer must be flushed. To flush the buffer, you can:

- For C or C++, use `fflush(stdout)` or `cout.flush()`.
- For Java, use `System.out.flush()`.
- For Python, use `stdout.flush()`.

Read **Sample Interaction 1** **Write**

```
1
4
                                            ? 1 2
1.000000000
                                            ? 2 3
3.000000000
                                            ? 3 4
3.000000000
                                            ? 1 4
1.000000000
                                            ! 1.000000000
Correct
```

F Following Arrows

Time limit: 2s

Panda is a puzzle designer and decides to create a two-dimensional maze based on an arrow-following theme. This maze is a grid with N rows and M columns. Each cell contains an arrow: `L` (Left), `R` (Right), `U` (Up), or `D` (Down).

Cells are indexed with (x, y), where x is the row $(1 \le x \le N)$ and y is the column $(1 \le y \le M)$. The top-left corner is $(1, 1)$ and the bottom-right corner is (N, M). The player starts at position $(1, 1)$, and proceeds in discrete steps until reaching (N, M) for the first time. Each step consists of the following two actions performed in order:

1. Move: Move one step in the direction of the arrow in your current cell. You only remain in your current cell if the attempted move would take you outside the maze.
2. Flip: After the move (or non-move) from step 1, the arrow in the cell where you were before the move changes to its opposite direction (i.e., change `L` to `R`, `R` to `L`, `U` to `D`, and `D` to `U`). If you didn't move, you flip the arrow in your current cell.

Panda wants to design a maze, with a maximum size of 8×8, such that a player starting at $(1, 1)$ and following the rules reaches (N, M) in **exactly** k steps.

Input

The first line contains an integer T $(1 \le T \le 2 \times 10^4)$, denoting the number of test cases.

Each test case contains an integer k $(1 \le k \le 10^6)$ in a single line, which is the required number of steps for the player to exit the maze.

It is guaranteed that the sum of k over all the test cases does not exceed 8×10^7.

Output

For each test case, if it is impossible to design such a maze, output `-1 -1` in a single line. Otherwise, first output one line containing two integers N, M $(1 \le N, M \le 8)$, representing the size of the maze. Then output N lines, with M characters per line, representing the maze. Every character in the maze is one of `L`, `R`, `U`, `D`. You must ensure that the number of operations required to go from $(1, 1)$ to (N, M) is exactly k.

Sample Input 1

```
5
1
4
9
16
25
```

Sample Output 1

```
1 2
RR
2 2
LU
RD
3 2
LU
DR
LL
3 2
LU
UR
LL
1 6
LLRLLR
```

This page is intentionally left blank.

G GCD of Subsets

Time limit: 1s

A multiset is a type of collection that allows duplicate elements. Panda has an integer multiset S and three integers n, k, m. Initially, $S = \{1, 2, \dots, n\}$. Panda wants to perform a series of operations on S. In each operation: choose a subset of integers from the current S such that the greatest common divisor (GCD) of all integers in this chosen subset is exactly k, and then remove all the selected integers from S.

At least one integer must be selected in each operation. The greatest common divisor (GCD) of a set of integers is the largest positive integer g that divides every integer in the set. For example, $\gcd(12, 16) = 4$, $\gcd(6, 9, 12) = 3$, and $\gcd(6, 10, 15) = 1$. Remember that the GCD of a single integer is the integer itself.

Before the first operation, Panda may choose **at most** m integers in S and change their values. Each selected integer can be changed to any value between 1 and n. **Note that this modification allows S to contain duplicate values.**

You should help Panda find the maximum number of operations he can successfully perform.

Input

The first line contains an integer T $(1 \le T \le 10^4)$, denoting the number of test cases.

For each test case, the input is one line with three integers n, k, m $(1 \le k \le n \le 10^{18}, 0 \le m \le n)$, where n is the upper limit of the initial multiset, k is the required GCD for subsets, and m is the maximum number of values that can be changed.

Output

For each test case, print an integer in one line, denoting the maximum number of operations that can be successfully performed.

Sample Input 1

```
2
4 1 0
5 3 1
```

Sample Output 1

```
2
2
```

Note

For the first test case, a possible solution involves two operations:

1. Choose subset $\{1, 4\}$ and remove them.
2. Choose subset $\{2, 3\}$ and remove them.

For the second test case, a solution is to change 1 into 3, then perform two operations:

1. Choose subset $\{3\}$ and remove it.
2. Choose subset $\{3\}$ and remove it.

This page is intentionally left blank.

H Heuristic Knapsack

Time limit: 2s

In computer science, the 0/1 Knapsack problem is a classic challenge: given a set of items, each with a weight and a value, choose a subset of items so that the total weight does not exceed a given limit, and the total value is as large as possible.

This problem is NP-hard. When the range of weights is relatively small, it can be solved exactly using the well-known dynamic programming algorithm. However, when both the weight and value ranges are very large, up to 10^9 for example, heuristic approaches are needed to obtain good (though not necessarily optimal) solutions.

Two researchers, Alice and Bob, are studying simple and fast "heuristic" algorithms that give good results for such large-scale knapsack instances. Each of them uses a greedy strategy:

- **Alice's Strategy: Lightest First**. Alice wants to keep the knapsack as light as possible. She sorts all items by weight in **ascending order**. If two items have the same weight, she breaks ties by their original index in ascending order. Then she takes a prefix of this sorted list: she picks items one by one until the next item in her order does not fit into the remaining capacity.

- **Bob's Strategy: Best Value First**. Bob wants to get the most value. He sorts all items by value in **descending order**. If two items have the same value, he breaks ties by their original index in ascending order. Then he goes through this list. For each item, if it fits in the remaining capacity, he takes it; otherwise, he skips it and continues.

Now, Alice and Bob are given a dataset of n items indexed from 1 to n and a knapsack with capacity W. However, some data is missing. For each item, its weight w_i and value r_i are given, but **at most one** of these two values might be unknown.

Your task is to assign a positive integer to each unknown value so that the set of items chosen by Alice is **exactly the same** as the set of items chosen by Bob (the order does not matter).

Input

The first line contains an integer T ($1 \le T \le 100$), denoting the number of test cases.

For each of the test cases, the first line inputs two integers n, W ($1 \le n \le 3000$, $1 \le W \le 10^9$), denoting the number of items and the knapsack capacity.

The second line contains n integers $w_1, w_2, \ldots, w_n$ ($0 \le w_i \le 10^9$), denoting the weights of the items. If $w_i = 0$, this parameter is unknown.

The third line contains n integers $r_1, r_2, \ldots, r_n$ ($0 \le r_i \le 10^9$), denoting the values of the items. If $r_i = 0$, this parameter is unknown. It is guaranteed that for each item, at most one of the two values is unknown.

It is guaranteed that $\sum n \le 3000$ over all test cases.

Output

For each test case, if there exists a valid assignment of positive integers to the unknown values that makes Alice's and Bob's chosen item sets identical, print `Yes` in the first line. Then print all w_i ($1 \le w_i \le 10^9$) in the second line separated by spaces, and print all r_i ($1 \le r_i \le 10^9$) in the third line separated by spaces, which denotes a possible valid assignment. Otherwise, print `No` in a single line.

Note that the maximum limits of w_i and r_i are both 10^9, **as same as** the input limits. Either `Yes` or `No` is case-insensitive, which means you can print `YeS`, `yEs`, `nO`, etc.

Input 1

```
6
3 5
0 2 3
2 4 0
3 5
2 3 0
4 0 2
3 1000000000
0 0 1000000000
2 3 1
5 1
0 1 1 1 0
3 8 0 7 9
5 1000000000
0 0 0 1 0
1 9 3 9 10
3 1000000000
500000001 500000000 0
1000000000 999999998 1
```

Output 1

```
Yes
3 2 3
2 4 1
Yes
2 3 2
4 1 2
Yes
1000000000 999999999 1000000000
2 3 1
Yes
1000000000 1 1 1 1000000000
3 8 1 7 9
Yes
1000000000 1000000000 1000000000 1 999999999
1 9 3 9 10
No
```

Note

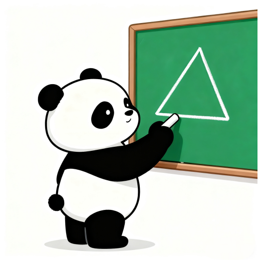
For the first test case of the example, Alice takes the second item and then the first; Bob also takes the second and then the first.

For the second test case, Alice and Bob both take the first and the third items.

For the last test case, there is no solution.

I Inside Triangle

Time limit: 2s



A polygon P_1 contains polygon P_2 if every point in or on P_2 is also in or on P_1. A polygon P_1 **strictly** contains P_2 if every point in or on P_2 is strictly inside P_1 (no point on the boundary).

Panda has two convex polygons, P and Q. Polygon P has n vertices, and polygon Q has m vertices. It is guaranteed that Q is **strictly** contained by P.

You should help Panda find the total number of ways to choose exactly 3 distinct vertices from polygon P that form a triangle P_Δ such that P_Δ (not strictly) contains polygon Q.

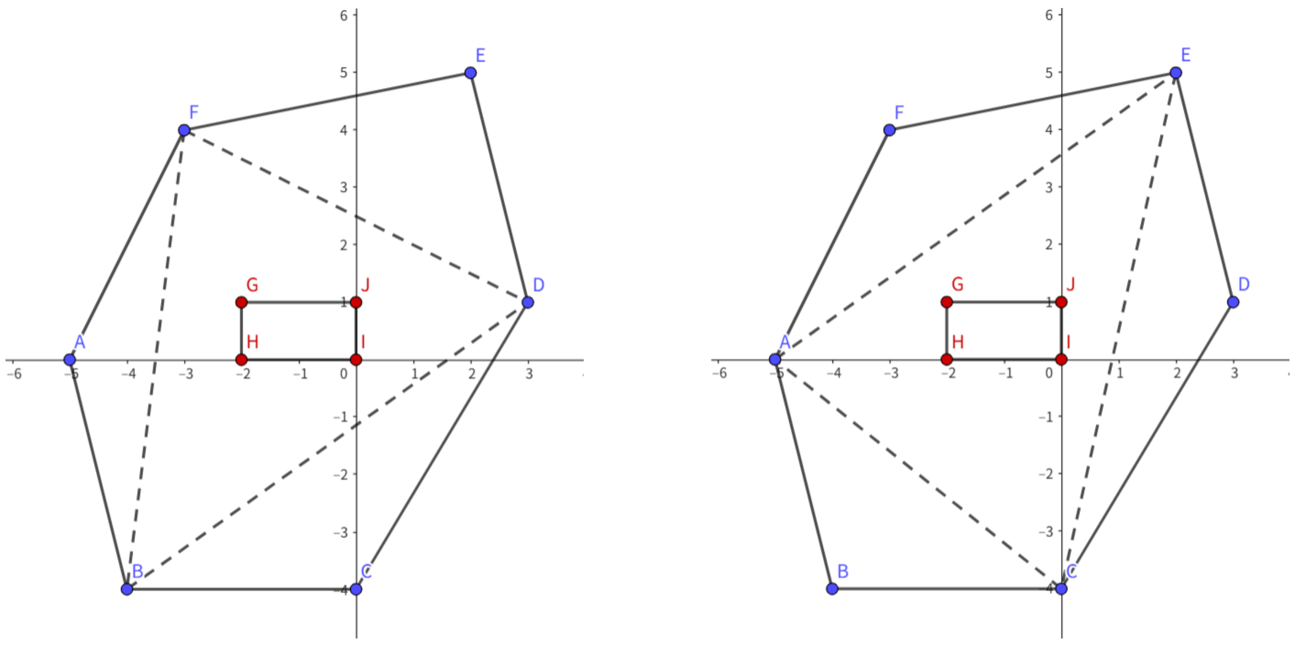


Figure I.1: Solutions of example 1.

Input

The first line contains a single integer T ($1 \le T \le 1000$), denoting the number of test cases.

For each test case, the first line contains an integer n ($3 \le n \le 3 \cdot 10^5$), denoting the number of vertices of P.

For the next n lines, each line contains two integers x, y ($|x|, |y| \le 10^9$), denoting the coordinates of the vertices of P. It is guaranteed that the vertices are given in counter-clockwise order, and no three vertices are collinear.

The next line contains an integer m ($3 \le m \le 3 \cdot 10^5$), denoting the number of vertices of Q.

For the next m lines, each line contains two integers x, y ($|x|, |y| \le 10^9$), denoting the coordinates of the vertices of Q. It is guaranteed that the vertices are given in counter-clockwise order, and no three vertices are collinear, and Q is **strictly** contained by P.

It is guaranteed that both $\sum n$ and $\sum m$ over all test cases do not exceed $5 \cdot 10^5$.

Output

For each test case, output an integer in a single line, denoting the answer.

Sample Input 1	Sample Output 1

```
3
6
-5 0
-4 -4
0 -4
3 1
2 5
-3 4
4
-2 1
-2 0
0 0
0 1
6
-5 0
-4 -4
0 -4
3 1
2 5
-3 4
4
-1 1
-1 0
0 0
0 1
7
1 -3
4 -1
5 0
1 4
0 4
-1 0
0 -2
3
0 0
2 0
0 2
```
	```
	2
	4
	6
	```

Note

For the first example, as shown in the figure, the polygon P is $ABCDEF$, and the polygon Q is $GHIJ$. The triangles $\triangle ACE$ and $\triangle BDF$ are the ones that contain Q.

J Judging Papers

Time limit: 1s

Panda, a highly respected professor, has submitted n papers to the *International Collegiate Programming Conference*. Each paper is judged independently by m anonymous reviewers who give it a score.

Reviews have seven levels, which correspond to scores:

- `Strong Reject`: -3
- `Reject`: -2
- `Weak Reject`: -1
- `Borderline`: 0
- `Weak Accept`: 1
- `Accept`: 2
- `Strong Accept`: 3

A paper is accepted if its total score (the sum of all m reviewer scores) is greater than or equal to a threshold k; otherwise, it is rejected.

After the reviewers submit their scores, there is a rebuttal phase. Panda can select a maximum of b distinct papers during this phase. A rebuttal is an argument aimed at improving the reviewers' evaluations for a selected paper. However, reviewers react differently, leading to these score changes:

- Reviewers who gave a high score (1 or higher) will react negatively to the rebuttal; their score is decreased by one level (e.g., 3 becomes 2, 1 becomes 0).
- Reviewers who gave a low score (0 or lower) will react positively to the rebuttal; their score is increased by one level (e.g., -3 becomes -2, 0 becomes 1).

Panda wants to strategically select **at most** b papers for a rebuttal to maximize the final number of accepted papers after all rebuttals are applied.

Input

The first line contains a single integer T ($1 \le T \le 1000$), denoting the number of test cases.

For each test case, the first line contains four integers n, m, k, b ($1 \le n \le 10^5$, $1 \le m \le 10$, $-30 \le k \le 30$, $0 \le b \le n$). n is the number of papers, m is the number of reviewers per paper, k is the acceptance score threshold, and b is the maximum number of papers that can be selected for a rebuttal.

The i-th of the next n lines contains m space-separated integers s_{ij} ($-3 \le s_{ij} \le 3$), representing the initial scores given by the j-th reviewer for the i-th paper.

It is guaranteed that $\sum n \le 2 \cdot 10^5$ over all test cases.

Output

For each test case, output an integer in one line, denoting the maximum number of papers that can be possibly accepted.

Sample Input 1

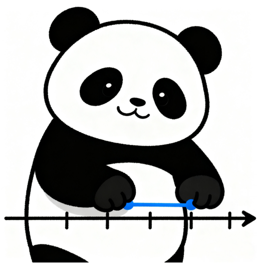
```
2
5 3 2 1
-3 0 3
2 -2 -1
1 1 1
0 0 0
-1 -1 -1
3 2 -1 1
-1 -2
-3 -3
1 -3
```

Sample Output 1

```
2
1
```

K K-Coverage

Time limit: 1s



Panda has a one-dimensional axis with n segments placed on its non-negative side. The left endpoint of the i-th segment is l_i. Every segment has the same length, L. This means the i-th segment covers the interval $[l_i, l_i + L - 1]$ (inclusive).

Now, Panda wants you to find the maximum number of integer points on the non-negative side of the axis that can be covered **exactly** k times after moving **at most one** segment. Specifically, you are allowed to select at most one segment and change its left endpoint to **any non-negative integer** (with no upper bound). The segment's length, L, remains the same.

Input

The first line contains an integer T $(1 \le T \le 2 \times 10^5)$, denoting the number of test cases.

For each test case, the first line contains three positive integers n, L, k $(1 \le n \le 2 \times 10^5, 1 \le L \le n, 1 \le k \le n)$. n is the total number of segments, L is the fixed length of every segment, and k is the required coverage count.

The second line contains n non-negative integers. The i-th integer is l_i $(0 \le l_i \le 2 \times n)$ representing the left endpoint of the i-th segment.

It is guaranteed that the sum of n over all test cases does not exceed 2×10^5.

Output

For each test case, output one line with a single integer, denoting the maximum number of integer points that can be covered exactly k times after moving at most one segment.

Sample Input 1

```
3
3 2 1
2 6 2
3 3 2
6 2 0
5 1 3
5 6 7 8 9
```

Sample Output 1

```
6
3
0
```

Note

For the first test case in the example, initially, the three segments are $[2, 3]$, $[6, 7]$, and $[2, 3]$. You can change the third segment's left endpoint to 114514. After this move, the segments are $[2, 3]$, $[6, 7]$, and $[114514, 114515]$. This results in 6 integer points covered exactly once, which is the maximum achievable.

For the second test case, you can move the first segment's left endpoint to 3 to maximize the number of integer points that are covered exactly twice.

This page is intentionally left blank.

L Label Matching

Time limit: 2s

Panda has a rooted tree with n nodes, where node 1 is the root. Each node i in the tree has two labels: a_i and b_i. Some of these values are designated as **wildcards**, which is represented by 0. Specifically, two values x and y are considered equal if $x = y$ or if at least one of them is a wildcard 0.

For a node i, let T_i denote the subtree rooted at node i, which consists of node i itself and all of its descendants in the tree. For each node i from 1 to n, Panda asks you the following question, and you must answer each one independently:

- You can perform any number of swap operations (including zero). In each operation, you can choose any two nodes u and v within the subtree T_i and swap a_u and a_v. Determine whether it is possible, using some sequence of swaps, to make a_k equal to b_k for all nodes k in the subtree T_i.

Note that each query is independent. The swaps you consider are only for that specific query and do not affect the initial state of the tree for subsequent queries.

Input

The first line contains a single integer T ($1 \le T \le 2 \times 10^5$), representing the number of test cases.

For each test case the first line contains a positive integer n ($1 \le n \le 2 \times 10^5$), indicating the number of nodes.

The second line contains n integers, the i-th of which is a_i ($0 \le a_i \le n$). If $a_i = 0$, it represents a wildcard.

The third line contains n integers, the i-th of which is b_i ($0 \le b_i \le n$). If $b_i = 0$, it represents a wildcard.

The next $n - 1$ lines describe the structure of the tree. Each line contains two integers u, v ($1 \le u, v \le n$, $u \ne v$), representing an edge between node u and node v. It is guaranteed that the given $n - 1$ edges form a tree.

The sum of n over all test cases is guaranteed to not exceed 2×10^5.

Output

For each test case, you must print a single line with a binary string s of length n. The i-th character of the string, s_i, should be 1 if a valid swap scheme exists for the subtree T_i, and 0 otherwise.

Sample Input 1

```
3
6
1 5 0 3 4 0
0 3 4 5 2 0
1 2
2 3
2 4
1 5
5 6
5
2 2 3 0 4
4 1 4 2 0
1 2
2 3
3 4
4 5
3
1 2 3
3 2 1
1 2
2 3
```

Sample Output 1

```
111011
01111
100
```

This page is intentionally left blank.

M Meeting for Meals

Time limit: 2s

In a beautiful campus, Panda and his friends are meeting for a meal. They enjoy spending time together on the way, as they can chat freely with each other.

The campus is modeled as a connected, undirected graph with n nodes and m edges with lengths. The meeting spot is the mall, which is located at node 1. k friends start from distinct nodes and travel to the mall. All friends move at the same speed (1 unit distance per unit time), and are allowed to change direction at any position along an edge (see the example explanation).

Let t_{meet} be the earliest possible time such that all friends can reach the mall by that time. Based on this earliest possible meeting time, for each friend individually, find the maximum total time they can spend with company (meaning at least one other friend is at the exact same location on the graph, which could be a node or a position along an edge) before the final gathering at the mall.

When considering a specific friend i, the paths and movement strategies of all other friends $j \neq i$ are optimized to maximize the company time for friend i, as long as all friends arrive at the mall no later than t_{meet}.

Input

The first line contains an integer T ($1 \le T \le 10^4$), denoting the number of test cases.

For each test case, the first line contains three integers n, m, k ($2 \le n \le 3 \times 10^5$, $n - 1 \le m \le 10^6$, $2 \le k \le n$), representing the number of nodes, edges, and friends, respectively.

The second line contains k distinct integers $a_1, a_2, \ldots, a_k$ ($1 \le a_i \le n$), denoting the starting nodes of the friends.

The next m lines each contain three integers u, v, w ($1 \le u, v \le n$, $u \neq v$, $1 \le w \le 10^9$), indicating an edge between node u and v with length w. The graph is connected and has no duplicate edges or self-loops.

The sum of n over all test cases does not exceed 3×10^5, and the sum of m over all test cases does not exceed 10^6.

Output

For each test case, output a line containing k floating-point numbers separated by spaces, each formatted to **exactly one decimal place**, representing the maximum time with company for each friend.

Sample Input 1

```
3
4 3 2
3 4
1 2 3
3 2 1
4 2 1
2 1 2
1 2
1 2 3
3 2 3
1 2 3
1 2 3
1 3 5
```

Sample Output 1

```
3.0 3.0
1.5 1.5
3.5 3.5 2.5
```

Note

For the third test case in the example, the earliest meeting time at the mall is 5 units of time.

- The friends from nodes 1 and 2 can meet midway on edge $(1, 2)$ in 1.5 units, travel together back to the mall in another 1.5 units, and then wait 2 units at the mall, totaling 3.5 units of companionship.
- The friend from node 3 meets the friends from node 1 midway on edge $(1, 3)$ in 2.5 units and returns with them in 2.5 units, totaling 2.5 units of companionship.