A. Two Frogs

```
Input file:     standard input
Output file:    standard output
Time limit:     1 second
Memory limit:   256 megabytes
```

Roaming through the alligator-infested Everglades, Florida Man encounters a most peculiar showdown.

There are n lilypads arranged in a row, numbered from 1 to n from left to right. Alice and Bob are frogs initially positioned on distinct lilypads, a and b, respectively. They take turns jumping, starting with Alice.

During a frog's turn, it can jump either one space to the left or one space to the right, as long as the destination lilypad exists. For example, on Alice's first turn, she can jump to either lilypad $a-1$ or $a+1$, provided these lilypads are within bounds. It is important to note that each frog **must jump** during its turn and cannot remain on the same lilypad.

However, there are some restrictions:

- The two frogs cannot occupy the same lilypad. This means that Alice cannot jump to a lilypad that Bob is currently occupying, and vice versa.
- If a frog cannot make a valid jump on its turn, it loses the game. As a result, the other frog wins.

Determine whether Alice can guarantee a win, assuming that both players play optimally. It can be proven that the game will end after a finite number of moves if both players play optimally.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 500$). The description of the test cases follows.

The first and only line of each test case contains three integers n, a, and b ($2 \le n \le 100$, $1 \le a, b \le n$, $a \ne b$) — the number of lilypads, and the starting positions of Alice and Bob, respectively.

Note that there are **no** constraints on the sum of n over all test cases.

Output

For each test case, print a single line containing either "YES" or "NO", representing whether or not Alice has a winning strategy.

You can output the answer in any case (upper or lower). For example, the strings "yEs", "yes", "Yes", and "YES" will be recognized as positive responses.

Standard Input	Standard Output

5
2 1 2
3 3 1
4 2 3
5 2 4
7 6 2
	NO
	YES
	NO
	YES
	YES

Note

In the first test case, Alice has no legal moves. Therefore, Alice loses on the first turn.

In the second test case, Alice can only move to lilypad 2. Then, Bob has no legal moves. Therefore, Alice has a winning strategy in this case.

In the third test case, Alice can only move to lilypad 1. Then, Bob can move to lilypad 2. Alice is no longer able to move and loses, giving Bob the win. It can be shown that Bob can always win regardless of Alice's moves; hence, Alice does not have a winning strategy.

,

B. Crafting

```
Input file:     standard input
Output file:    standard output
Time limit:     1 second
Memory limit:   256 megabytes
```

As you'd expect, Florida is home to many bizarre magical forces, and Florida Man seeks to tame them.

There are n different types of magical materials, numbered from 1 to n. Initially, you have a_i units of material i for each i from 1 to n. You are allowed to perform the following operation:

- Select a material i (where $1 \le i \le n$). Then, spend 1 unit of **every** other material j (in other words, $j \neq i$) to gain 1 unit of material i. More formally, after selecting material i, update array a as follows: $a_i := a_i + 1$, and $a_j := a_j - 1$ for all j where $j \neq i$ and $1 \le j \le n$. Note that all a_j must remain non-negative, i.e. you cannot spend resources you do not have.

You are trying to craft an artifact using these materials. To successfully craft the artifact, you must have at least b_i units of material i for each i from 1 to n. Determine if it is possible to craft the artifact by performing the operation any number of times (including zero).

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains a single integer n ($2 \le n \le 2 \cdot 10^5$) — the number of types of materials.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^9$) — the amount of each material i that you currently hold.

The third line of each test case contains n integers $b_1, b_2, \ldots, b_n$ ($0 \le b_i \le 10^9$) — the amount of each material i needed to produce the artifact.

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

Output

For each test case, print a single line containing either "YES" or "NO", representing whether or not the artifact can be crafted.

You can output the answer in any case (upper or lower). For example, the strings "yEs", "yes", "Yes", and "YES" will be recognized as positive responses.

Standard Input	Standard Output

3
4
0 5 5 1
1 4 4 0
3
1 1 3
2 2 1
	YES
	NO
	YES

2
1 10
3 3	

Note

In the first test case, perform an operation on material 1. After doing so, we will have exactly the required resources: 1 unit of material 1, and 4 units each of materials 2 and 3.

In the second test case, it can be shown that no matter how the operations are performed, it is impossible to craft the artifact.

In the third test case, we can perform the operation on material 1 twice. After these operations, we will have 3 units of material 1 and 8 units of material 2, which is more than enough to craft the artifact.

C. The Trail

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

There are no mountains in Florida, and Florida Man cannot comprehend their existence. As such, he really needs your help with this one.

In the wilderness lies a region of mountainous terrain represented as a rectangular grid with n rows and m columns. Each cell in the grid is identified by its position (i, j), where i is the row index and j is the column index. The altitude of cell (i, j) is denoted by $a_{i,j}$.

However, this region has been tampered with. A path consisting of $n + m - 1$ cells, starting from the top-left corner $(1, 1)$ and ending at the bottom-right corner (n, m), has been cleared. For every cell (i, j) along this path, the altitude $a_{i,j}$ has been set to 0. The path moves strictly via downward ($\mathtt{D}$) or rightward ($\mathtt{R}$) steps.

To restore the terrain to its original state, it is known that the region possessed a magical property before it was tampered with: all rows and all columns shared the same sum of altitudes. More formally, there exists an integer x such that $\sum_{j=1}^{m} a_{i,j} = x$ for all $1 \le i \le n$, and $\sum_{i=1}^{n} a_{i,j} = x$ for all $1 \le j \le m$.

Your task is to assign new altitudes to the cells on the path such that the above magical property is restored. It can be proven that a solution always exists. If there are multiple solutions that satisfy the property, any one of them may be provided.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains two integers n and m ($2 \le n, m \le 1000$) — the number of rows and columns in the grid.

The second line of each test case contains a string s of length $n + m - 2$ ($s_i = \mathtt{D}$ or $s_i = \mathtt{R}$) — the steps the path makes from $(1, 1)$ to (n, m). The character $\mathtt{D}$ represents a downward step, and $\mathtt{R}$ represents a rightward step.

The i-th of the next n lines each contain m integers $a_{i,1}, a_{i,2}, \ldots, a_{i,m}$ ($-10^6 \le a_{i,j} \le 10^6$) — the altitude of each cell in the grid. It is guaranteed that if a cell (i, j) lies on the path, then $a_{i,j} = 0$.

It is guaranteed that the sum of $n \cdot m$ over all test cases does not exceed 10^6.

Output

For each test case, output n lines of m integers representing the restored grid of altitudes $b_{i,j}$. The altitudes must satisfy $-10^{15} \le b_{i,j} \le 10^{15}$, and additionally $a_{i,j} = b_{i,j}$ if (i, j) is not on the path. If multiple solutions exist, output any of them.

Standard Input	Standard Output

4
3 3
DRRD
	1 2 3
	2 3 1
	3 1 2

```
0 2 3
0 0 0
3 1 0
4 5
DRRRRDD
0 1 0 2 3
0 0 0 0 0
-1 0 -3 -3 0
0 0 0 -1 0
2 3
RRD
0 0 0
0 1 0
5 5
DDDDRRRR
0 25 2 9 11
0 6 13 20 22
0 17 24 1 8
0 3 10 12 19
0 0 0 0 0
```

```
-6 1 0 2 3
7 -1 3 2 -11
-1 0 -3 -3 7
0 0 0 -1 1
0 -1 1
0 1 -1
18 25 2 9 11
4 6 13 20 22
15 17 24 1 8
21 3 10 12 19
7 14 16 23 5
```

Note

In the first test case, the grid has been filled such that every row and column contains the numbers $1, 2, 3$ in some order, resulting in a common sum of 6.

In the second test case, the grid has been filled such that all rows and columns sum to 0.

,

D. Scarecrow

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

At his orange orchard, Florida Man receives yet another spam letter, delivered by a crow. Naturally, he's sending it back in the most inconvenient manner possible.

A crow is sitting at position 0 of the number line. There are n scarecrows positioned at integer coordinates $a_1, a_2, \ldots, a_n$ along the number line. These scarecrows have been enchanted, allowing them to move left and right at a speed of 1 unit per second.

The crow is afraid of scarecrows and wants to stay at least a distance of k ahead of the nearest scarecrow positioned **at or before** it. To do so, the crow uses its teleportation ability as follows:

- Let x be the current position of the crow, and let y be the largest position of a scarecrow such that $y \le x$. If $x - y < k$, meaning the scarecrow is too close, the crow will instantly teleport to position $y + k$.

This teleportation happens instantly and continuously. The crow will keep checking for scarecrows positioned at or to the left of him and teleport whenever one gets too close (which could happen at non-integral times). Note that besides this teleportation ability, the crow will not move on its own.

Your task is to determine the minimum time required to make the crow teleport to a position greater than or equal to ℓ, assuming the scarecrows move optimally to allow the crow to reach its goal. For convenience, you are asked to output **twice the minimum time** needed for the crow to reach the target position ℓ. It can be proven that this value will always be an integer.

Note that the scarecrows can start, stop, or change direction at any time (possibly at non-integral times).

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains three integers n, k, ℓ ($1 \le n \le 2 \cdot 10^5$, $1 \le k \le \ell \le 10^8$) — the number of scarecrows, the teleportation distance, and the target position of the crow, respectively.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($0 \le a_1 \le a_2 \le \ldots \le a_n \le \ell$) — the initial positions of the n scarecrows.

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

Output

For each test case, output a single integer representing the **twice the minimum time** required for the crow to teleport to a position greater than or equal to ℓ.

Standard Input	Standard Output

9
1 3 5
0
	4
	5
	20

```
3 2 5
2 5 5
1 10 10
10
10 1 10
0 1 2 3 4 5 6 7 8 9
2 1 2
0 0
2 1 2
0 2
2 1 3
0 2
2 2 4
1 1
9 12 54
3 3 8 24 25 27 29 34 53
```

```
0
2
1
2
2
7
```

Note

In the first test case, the crow instantly teleports to position 3 due to the scarecrow at position 0. This scarecrow may then move to position 2, causing the crow to continuously move from position 3 to position 5, completing the trip in 2 seconds. Therefore, the output is 4.

In the second test case, scarecrow 1 and scarecrow 2 can move to positions 0 and 3, respectively, in 2 seconds, while scarecrow 3 remains at position 5. The crow teleports to position 2 due to scarecrow 1. Then, scarecrow 1 moves to the right while scarecrow 2 and scarecrow 3 move to the left for 0.5 seconds. This causes the crow to continuously move from position 2 to position 2.5 due to scarecrow 1 moving right from position 0. After this half second, the scarecrows will be at positions $0.5, 2.5, 4.5$. Scarecrow 2, now at position 2.5, causes the crow to instantly teleport to position 4.5, and scarecrow 3 at position 4.5 causes it to instantly teleport to position 6.5, which exceeds $\ell = 5$. Therefore, the crow finishes the trip in just 2.5 seconds, and the output is 5.

It can be shown that these are the minimum possible times for both test cases.

E. Haystacks

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

On the next new moon, the universe will reset, beginning with Florida. It's up to Florida Man to stop it, but he first needs to find an important item.

There are n haystacks labelled from 1 to n, where haystack i contains a_i haybales. One of the haystacks has a needle hidden beneath it, but you do not know which one. Your task is to move the haybales so that each haystack is emptied at least once, allowing you to check if the needle is hidden under that particular haystack.

However, the process is not that simple. Once a haystack i is emptied for the first time, it will be assigned a height limit and can no longer contain more than b_i haybales. More formally, a move is described as follows:

- Choose two haystacks i and j. If haystack i has not been emptied before, or haystack i contains strictly less than b_i haybales, you may move exactly 1 haybale from haystack j to haystack i.

Note: Before a haystack is emptied, it has no height limit, and you can move as many haybales as you want onto that haystack.

Compute the minimum number of moves required to ensure that each haystack is emptied at least once, or report that it is impossible.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains a single integer n ($2 \le n \le 5 \cdot 10^5$) — the number of haystacks.

The i-th of the next n lines contains two integers a_i and b_i ($1 \le a_i, b_i \le 10^9$) — the initial number of haybales in the i-th haystack, and the height limit that it is assigned after it is emptied for the first time.

It is guaranteed that the sum of n over all test cases does not exceed $5 \cdot 10^5$.

Output

For each test case, print a single integer — the minimum number of moves required to ensure that each haystack is emptied at least once. If it is not possible to empty each haystack at least once, output `-1`.

Standard Input	Standard Output

7
2
3 5
2 4
2
10 1
1 10
3
1 3
	8
	-1
	8
	9
	14
	15
	19

```
4 3
1 1
3
5 4
2 4
1 10
6
2 1
3 3
5 4
1 5
1 6
1 8
5
3 2
1 2
1 1
1 3
6 5
2
5 10
7 12
```

Note

In the first test case, we can do the following sequence of moves:

- Move 3 haybales from haystack 1 to haystack 2. Haystack 1 is now emptied, and is assigned a height limit of 5.
- Move 5 haybales from haystack 2 to haystack 1. Haystack 2 is now emptied, and is assigned a height limit of 4.

The above sequence requires $3+5=8$ moves. It is not possible to use less than 8 moves as the following sequence of moves is invalid:

- Move 2 haybales from haystack 2 to haystack 1. Haystack 2 is now emptied, and is assigned a height limit of 4.
- Move 4 haybales from haystack 1 to haystack 2. Haystack 1 now has 1 haybale, while haystack 2 has 4 haybales.
- Haystack 1 cannot be emptied as haystack 2 is already at its height limit of 4, so no more haybales can be moved from haystack 1 to haystack 2.

In the second test case, the task is impossible. This is because the height limits of both haystacks are too small that once one of the haystacks is emptied, the other haystack cannot be emptied due to the small height limits.

In the third test case, the following sequence of moves can be shown to be optimal:

- Move 1 haybale from haystack 1 to haystack 3. Haystack 1 is now emptied, and is assigned a height limit of 3.
- Move 3 haybales from haystack 2 to haystack 1.
- Move 1 haybale from haystack 2 to haystack 3. Haystack 2 is now emptied and is assigned a height limit of 3.

- Move 3 haybales from haystack 3 to haystack 2. Haystack 3 is now emptied, and is assigned a height limit of 1.

The above sequence requires $1 + 3 + 1 + 3 = 8$ moves.

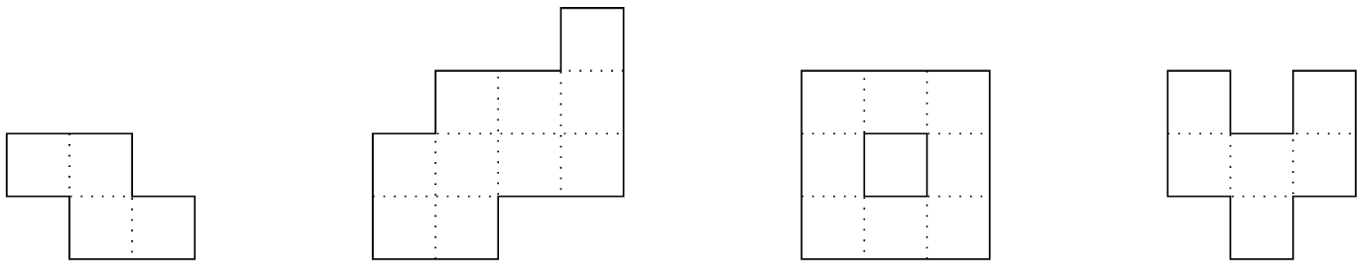
,

F. Cosmic Divide

```
Input file:     standard input
Output file:    standard output
Time limit:     4 seconds
Memory limit:   256 megabytes
```

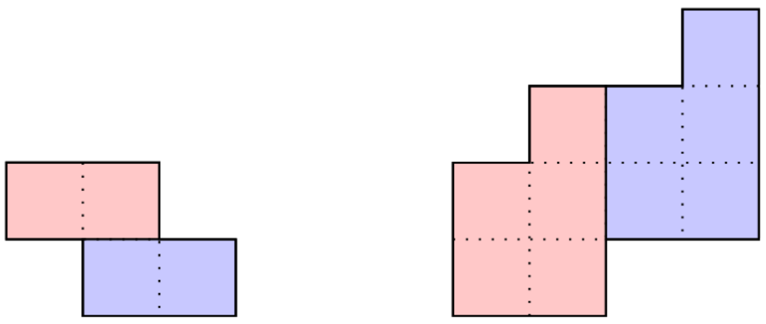
With the artifact in hand, the fabric of reality gives way to its true master — Florida Man.

A *polyomino* is a connected[*] figure constructed by joining one or more equal 1×1 unit squares edge to edge. A polyomino is *convex* if, for any two squares in the polyomino that share the same row or the same column, all squares between them are also part of the polyomino. Below are four polyominoes, only the first and second of which are convex.



You are given a convex polyomino with n rows and an even area. For each row i from 1 to n, the unit squares from column l_i to column r_i are part of the polyomino. In other words, there are $r_i - l_i + 1$ unit squares that are part of the polyomino in the i-th row: $(i, l_i), (i, l_i + 1), \ldots, (i, r_i - 1), (i, r_i)$.

Two polyominoes are *congruent* if and only if you can make them fit exactly on top of each other by translating the polyominoes. **Note that you are not allowed to rotate or reflect the polyominoes.** Determine whether it is possible to partition the given convex polyomino into two disjoint connected polyominoes that are congruent to each other. The following examples illustrate a valid partition of each of the two convex polyominoes shown above:



The partitioned polyominoes do not need to be convex, and each unit square should belong to exactly one of the two partitioned polyominoes.

[*] A polyomino is connected if and only if for every two unit squares $u \neq v$ that are part of the polyomino, there exists a sequence of distinct squares $s_1, s_2, \ldots, s_k$, such that $s_1 = u$, $s_k = v$, s_i are all part of the polyomino, and s_i, s_{i+1} share an edge for each $1 \leq i \leq k - 1$.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first line of each test case contains a single integer n ($1 \le n \le 2 \cdot 10^5$) — the number of rows of the polyomino.

The i-th of the next n lines contains two integers l_i and r_i ($1 \le l_i \le r_i \le 10^9$) — the range of columns that are part of the polyomino in the i-th row.

It is guaranteed that the area of the polyomino is even. In other words, $\sum_{i=1}^{n} r_i - l_i + 1 \equiv 0 \pmod 2$.

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

Output

For each test case, print a single line containing either "YES" or "NO", representing whether or not the polyomino can be partitioned as described in the problem.

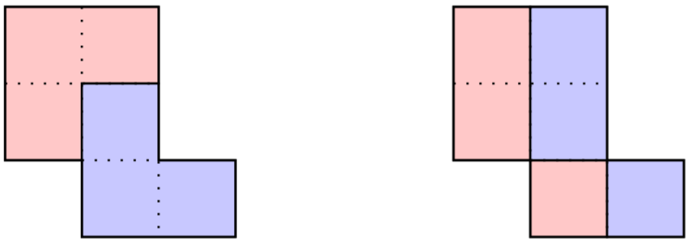
You can output the answer in any case (upper or lower). For example, the strings "yEs", "yes", "Yes", and "YES" will be recognized as positive responses.

Standard Input	Standard Output
7	YES
2	YES
1 2	NO
2 3	NO
4	NO
4 4	NO
2 4	YES
1 4	
1 2	
3	
1 2	
1 2	
2 3	
2	
1 3	
3 3	
2	
1 3	
2 2	
3	
1 2	
1 3	
1 3	
4	
8 9	
6 8	
6 8	
5 6	

Note

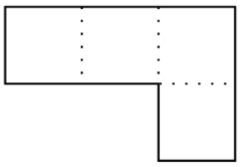
The first and second test cases are the polyominoes depicted in the problem statement and can be partitioned as shown.

The polyomino in the third test case, shown below, can be shown to be impossible to partition. None of the following partitions are valid:



The partition on the left does not use polyominoes that are translations of each other, and the partition on the right does not use connected polyominoes.

The polyomino in the fourth test case, shown below, can be shown to be impossible to partition.



Note that while you can partition it into two 1×2 rectangles, these rectangles are not translations of each other.