2019 LGE Code Jam: Online Round 1

Problem A. Chess

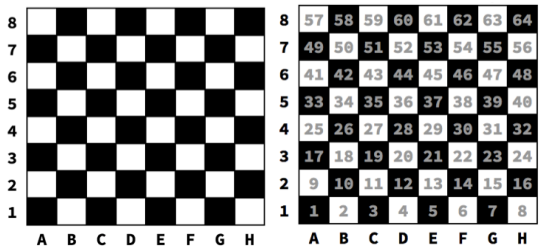
Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 sec
Memory limit:	512 MB

Consider a typical 8×8 chess board.

A chess board contains 64 cells, across 8 rows (labeled from 1 to 8) and 8 columns (labeled from A to H). Notice that there are exactly 32 black cells (e.g., A1 and h8) and 32 white cells (e.g., h1 and A8).

We can use two kinds of notation to refer to a cell.

- Notation 1: A cell's column index (in upper-case) followed by its row index (e.g., "A1", "D4", "H8" where quotes for clarity only).
- Notation 2: A cell's numeric label (a number written on each cell in the right figure below) which ranges between 1 and 64, inclusive.



Using these representations, a cell can be referred by two different ways. For instance, "A3" and "17" refer to the same cell. In another example, "38" and "F5" refer to the same cell.

Given a cell described by Notation 1 and another cell described by Notation 2, determine whether these two cells are of the same color or not.

Input

The first line will contain a positive integer T ($1 \leq T \leq 10$), the number of test cases.

Each test case will be given in a single line that will contain one string and one number. The string (in Notation 1) is given first, followed by a whitespace, and the number (in Notation 2) is given in each line.

The string is always of length 2 and its first character is an uppercase English alphabet ('A'-'H') and its second character is a digit ('1'-'8'). The number is always between 1 and 64, inclusive.

You can assume that every test case is a valid input.

Output

For each test case, you should output in a single line "YES" (quotes for clarity only) if the given two cells have the same color. Otherwise, output "NO" (quotes for clarity only).

Examples

standard input	standard output
4 A1 17 A1 1 B3 21 G5 40	YES YES NO NO

Notes

- Both “A1” and “17” are black cells.
- Both “A1” and “1” refer to the same cell (hence the same color).
- “B3” is a white cell and “21” is a black cell.
- “G5” is black cell and “40” is a white cell.

Problem B. Array Manipulation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 sec
Memory limit:	512 MB

We have a 2-dimensional square array A with N rows and N columns. We denote the value of an element in A at row r and column c by $A[r][c]$.

Now, consider an operation defined by $(r1, c1, r2, c2, val)$ with $1 \leq r1 \leq r2 \leq N, 1 \leq c1 \leq c2 \leq N$, and $-1,000 \leq val \leq 1,000$.

This operation adds val to all elements of A inside the rectangle defined by its upper left corner $(r1, c1)$ and lower right corner $(r2, c2)$.

For instance, suppose that $N = 3$ and $A = [[123][456][789]]$. Consider three operations with the following parameters:

- Operation 1: $(r1 = 1, c1 = 1, r2 = 2, c2 = 3, v = 3)$
- Operation 2: $(r1 = 2, c1 = 2, r2 = 3, c2 = 2, v = -5)$
- Operation 3: $(r1 = 1, c1 = 1, r2 = 3, c2 = 2, v = 1)$

Before any operation is performed, the array looks like this:

```
A = [ [ 1 2 3 ]
      [ 4 5 6 ]
      [ 7 8 9 ] ].
```

After applying operation 1, the six elements in the first two rows of A will be updated (emphasized in bold text):

```
A = [ [ 4 5 6 ]
      [ 7 8 9 ]
      [ 7 8 9 ] ].
```

After applying operation 2:

```
A = [ [ 4 5 6 ]
      [ 7 3 9 ]
      [ 7 3 9 ] ].
```

After applying operation 3:

```
A = [ [ 5 6 6 ]
      [ 8 4 9 ]
      [ 8 4 9 ] ].
```

After applying all 3 operations, Albert wants to compute the sum of N elements in each row and each column. In the above example, the row-sums are equal to [17, 21, 21] (from row 1 to row 3) and the column-sums are equal to [21, 14, 24] (from column 1 to column 3).

Our goal is to compute all N row sums and N column sums after applying M operations.

Input

The first line will contain the number of test cases T $(1 \leq T \leq 10)$.

For each test case: The first line will contain N and M where $1 \leq N \leq 1,000$ and $1 \leq M \leq 1,000$.

The following N lines will contain N integers, each separated by a whitespace.

The i-th line is the i-th row of A, whose j-th integer is $A[i][j]$.

Each element of A will be between 1 and 1,000, inclusive.

Each of the next M lines contains five integers $r1, c1, r2, c2$, and val, separated by a whitespace, denoting a single operation.

The ranges of parameters are $1 \leq r1 \leq r2 \leq N$ and $1 \leq c1 \leq c2 \leq M$ and $-1,000 \leq v \leq 1,000$.

Output

For each test case, output two lines;

The first line should output N row sums, the second line should output N column sums, after applying all operations.

Examples

standard input	standard output
3 3 3 1 2 3 4 5 6 7 8 9 1 1 2 3 3 2 2 3 2 -5 1 1 3 2 1 2 1 10 20 30 40 1 1 2 2 -30 1 3 1000 1 1 1 1 1000 1 1 1 1 -1000 1 1 1 1 1000	17 21 21 21 14 24 -30 10 -20 0 2000 2000

Notes

The first test case was described in the problem statement.

For the second case, after the operation, the array will look like:

```
[ [ -20 -10 ]
  [   0  10 ] ].
```

Therefore the row-sums are [-30, 10] and the column-sums are [-20, 0].

For the third case, the final array contains one element of value 2000.

In most cases, execution time of your python code will be much faster using PyPy than Python, so if your submission resulted in Time Limit Exceeded, then it's recommended that you submit your code using PyPy.

Problem C. Two Arrays

Input file:	standard input
Output file:	standard output
Time limit:	2 sec
Memory limit:	512 MB

There are two arrays, A and B, each containing distinct positive integers with length n and m respectively.

Our goal is to construct array C, with length n, as follows:

For each i, $C[i]$ is an element of B that is closest to $A[i]$ (in absolute difference)

If there are multiple possible values, choose the smallest one.

For instance, consider $A = [20, 5, 14, 9]$ and $B = [16, 8, 12]$,

- $C[1] = 16$ because $B[1] = 16$ is closest to $A[1] = 20$.
- $C[2] = 8$ because $B[2] = 8$ is closest to $A[2] = 5$.
- $C[3] = 12$ because both $B[1] = 16$ and $B[3] = 12$ are equally close to $A[3] = 14$, but $B[3]$ is smaller.
- $C[4] = 8$.

Hence in this case we have $C = [16, 8, 12, 8]$.

Given two arrays A and B, construct an array C as above, and output the sum of all elements in C.

Input

The first line will contain the number of test case T ($1 \leq T \leq 10$).

A single test case consists of three lines:

The first line will contain two integers n and m ($1 \leq n, m \leq 10^6$).

The second line will contain n integers (delimited by whitespace) representing $A[1]$ to $A[n]$ (between 1 and 10^9, inclusive).

The third line will contain m integers (delimited by whitespace) representing $B[1]$ to $B[m]$ (between 1 and 10^9, inclusive).

Recall that A contains distinct integers and B contains distinct integers.

Output

For each test case, compute the array C and output the sum of elements in C.

Examples

standard input	standard output
3 4 3 20 5 14 9 16 8 12 3 4 16 8 12 20 5 14 9 3 3 1 2 3 2 3 4	44 37 7

Notes

The first case was explained in the problem statement.

In the second case, we have $A = [16, 8, 12]$ and $B = [20, 5, 14, 9]$. C is then $[14, 9, 14]$ and therefore the answer is 37.

For the third case, we have $C = [223]$ and the answer is 7.

Problem D. Number Game

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 sec
Memory limit:	512 MB

Albert is a teacher, and he's teaching n children in his math class. Everyone loves candies, so he's invented a fun game for them to play while he can get their attention by giving out candies.

Here is how the game works. Assume that the children in Albert's class are numbered from 1 to n (there are n children in his class). Child i knows of integers between 1 and $x[i]$, inclusive.

The game plays out for possibly many turns as follows.

1. At the beginning of each turn, every child picks an arbitrary number that he/she knows of, and writes it down on a piece of paper.
2. After everyone finishes writing, Albert collects n pieces of paper.
3. Albert then sorts the numbers in non-decreasing order, and obtains a sorted list of numbers.
4. If this sorted list was obtained in any of the previous turns, the list is discarded and the game continues. If it has not been obtained previously, then every child will get a candy from Albert, and the game continues.
5. The game will end when every possible sorted list of numbers has been obtained.

Because everyone knows of only a finite number of integers, the game will eventually end, but Albert needs to figure out how many candies he needs to prepare in the worst case. Given n (the number of children) and an array $x[]$ (where $x[i]$ is the largest number child i knows of), compute the maximum number of candies Albert will need to give out if the game plays out until every sorted list has been obtained. This number can be quite large, so you only need to compute this number modulo 1,000,000,007.

For instance, suppose that there are two children Alice and Elise in Albert's class and Alice knows 3 integers (1, 2, and 3) and Elise knows 3 integers (1, 2, and 3). In this case, there are six sorted lists of two integers that can be obtained during the game: 1, 1, 1, 2, 1, 3, 2, 2, 2, 3, and 3, 3. Each time one of these lists is obtained for the first time, Albert will need to give out two candies, and therefore Albert will need 12 candies.

Input

The first line will contain T ($1 \leq T \leq 10$), the number of test cases.

Each test case will consist of two lines. The first line will contain n ($1 \leq n \leq 200$), the number of children in Albert's math class. The second line will contain n integers (delimited by a whitespace) that describe the array $x[]$. Assume that $1 \leq x[i] \leq 200$ for all i.

Output

For each test case, output the maximum number of candies Albert needs to give out.

Examples

standard input	standard output
4 1 3 2 3 3 3 1 2 3 6 95 96 97 98 99 100	3 12 15 627383157

Notes

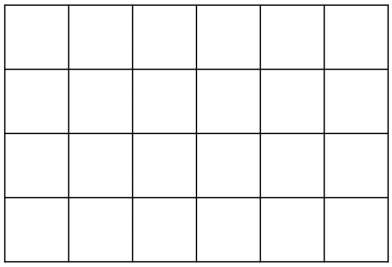
- Case 1: With only one child who knows 3 integers, the answer is trivially 3. The three sorted lists are: 1, 2, and 3 and each time one of these lists is obtained for the first time, Albert will need to give out one candy.
- Case 2: This is described in the problem statement.
- Case 3: The number of sorted lists is five: 1, 1, 1, 1, 1, 2, 1, 1, 3, 1, 2, 2, 1, 2, 3. Since there are three children, Albert will need to prepare 15 candies.
- Case 4: Recall that you will need to compute the answer modulo 1,000,000,007 because the answer can become quite large. There are 1,604,563,870 sorted lists in this case, and there are six children (hence Albert needs to prepare 9,627,383,220 candies). 9,627,383,220 modulo 1,000,000,007 is equal to 627,383,157.

Problem E. Cleaning Robot

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 sec
Memory limit:	512 MB

Albert's warehouse is rectangle-shaped, and it consists of R rows and C columns; each cell contains a massive shelf that can only be cleaned by a robot.

The example figure below shows a table with 4 rows and 6 columns.



Albert will place his smart cleaning robot at row r and column c, and the robot will first scan the entire warehouse, and will determine whether it will be possible to visit (and clean up) each shelf exactly once while only moving by up/down/left/right.

For instance, suppose that $R = 4, C = 6, r = 3$, and $c = 2$. Starting from the cell at row 3 and column 2, Albert's robot can visit all 24 cells only by moving Up/Down/Left/right, as shown in the figure below. Alternatively, this order can be concisely described by the direction at which the robot moves: "URDDRRRULLURRULLLLLDDDR" (23 moves).

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

However, it is not always possible to visit all cells of the board. For instance, when $R = 3, C = 3, r = 1$, and $c = 2$, there is no way to visit all nine cells exactly once.

Given R, C, r, and c, you must help Albert by determining whether the robot will be able to clean up each shelf exactly once. If it is possible, then you should also output how it can be done. The robot can only move up/down/left/right and it cannot go out of the warehouse.

Input

The first line will contain an integer T ($1 \leq T \leq 100$), the number of test cases.

Each of the following T lines will contain four integers separated by a white space, representing R, C, r, and c in this order. You can assume that $1 \leq R, C \leq 100$ and $1 \leq r \leq R$ and $1 \leq c \leq C$. You can further assume that $R \times C > 1$ (i.e., the warehouse always contains more than one shelf).

Output

For each test case, you must output one string in a single line.

If it is impossible to visit each shelf exactly once, then you should output "IMPOSSIBLE" (without quotes). Otherwise, you should output a string of length $(R \times C) - 1$, which describes how the robot should move (each character must be one of U, D, L, R describing the direction at which the robot moves). U means going up (by decrementing the row number), D means going down (by incrementing the row number), L means going left (by decrementing the column number), and R means going right (by incrementing the column number).

Subtask 1 (10 points)

- $1 \leq R, C \leq 5$

Subtask 2 (21 points)

- $1 \leq R, C \leq 100$

Examples

standard input	standard output
6 2 2 1 1 3 3 1 2 3 3 2 2 5 5 3 2 4 6 3 2 1 3 1 2	RDL IMPOSSIBLE LURRDDLL IMPOSSIBLE RDLLUURRRDDRRULURULLLLL IMPOSSIBLE

Notes

For each test case, you must output one string in a single line.

If it is impossible to visit each shelf exactly once, then you should output "IMPOSSIBLE" (without quotes). Otherwise, you should output a string of length (R*C)-1, which describes how the robot should move (each character must be one of U, D, L, R describing the direction at which the robot moves). U means going up (by decrementing the row number), D means going down (by incrementing the row number), L means going left (by decrementing the column number), and R means going right (by incrementing the column number).

Problem F. Operation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 sec
Memory limit:	512 MB

Consider an array with 0-based index. Let us write $A[i]$ to refer to the value of this array at index i. Initially, $A[i] = 0$ for all valid i.

There are three operations given as follows:

> $(key, value)$: adds $A[key]$ by value, returns the sum of all elements of A afterwards. (key): assigns 0 to $A[key]$, again, returns the sum of all elements of A afterwards. $(key1, key2)$: Returns the sum of elements of A between indices $\min(key1, key2)$ and $\max(key1, key2)$, inclusive. Note that this operation does not mutate the array.

Example:

- Addition $(key = 27, value = 30) \rightarrow 30$
- Addition $(key = 25, value = 40) \rightarrow 70$
- Purge $(key = 17) \rightarrow 70$
- Addition $(key = 17, value = 20) \rightarrow 90$
- Addition $(key = 5, value = 50) \rightarrow 140$
- Sum $(key1 = 10, key2 = 20) \rightarrow 20$
- Sum $(key1 = 25, key2 = 30) \rightarrow 70$
- Purge $(key = 25) \rightarrow 100$
- Purge $(key = 17) \rightarrow 80$
- Addition $(key = 27, value = 20) \rightarrow 100$
- Sum $(key1 = 10, key2 = 20) \rightarrow 0$
- Sum $(key1 = 25, key2 = 30) \rightarrow 50$

Given n operations, output returned values for each operation.

Input

The first line contains n number of operations.

Each of the following n lines will describe a single operation.

- A line begins with 1 denotes an addition operation, followed by a key and a value, delimited by a whitespace.
- A line begins with 2 denotes a sum operation, followed by two keys, delimited by a whitespace.
- A line begins with 3 denotes a purge operation, followed by a key.

Output

Output returned values of each operation.

Subtask 1 (6 points)

- $1 \leq n \leq 1,000$ and all keys will be between 0 and 1,000-1, inclusive. All values will be between -10^9 and 10^9, inclusive.

Subtask 2 (26 points)

- $1 \leq n \leq 100,000$ and all keys will be between 0 and $(10^9 9) - 1$, inclusive. All values will be between -10^9 and 10^9, inclusive.

Examples

standard input	standard output
12 1 27 30 1 25 40 3 17 1 17 20 1 5 50 2 10 20 2 25 30 3 25 3 17 1 27 20 2 10 20 2 25 30	30 70 70 90 140 20 70 100 80 100 0 50
8 1 2 10 1 4 10 1 15 10 2 2 15 2 3 14 3 4 3 10 2 4 15	10 20 30 30 10 20 20 10

Notes

Your submission may receive Time Limit Exceeded (TLE) error if input and/or output is large.

C++: If you want to use `cin/cout`, then you should use both `cin.tie(NULL)` and `sync_with_stdio(false)`, and must use "\n" (newline character) instead of `endl`. If you do so, you should not use `scanf/printf/puts/getchar/putchar` (I/O methods for C).

Java: You should use `BufferedReader` and `BufferedWriter` instead of `Scanner` and `System.out`. You only need to call BufferedWriter.flush once at the end of your program.

Python: You should use `sys.stdin.readline` instead of input. This will also read the newline character ("\n") from each line, so it is recommended that you add `.rstrip()` at the end `sys.stdin.readline`.