A. Square

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

A square of positive (strictly greater than 0) area is located on the coordinate plane, with sides parallel to the coordinate axes. You are given the coordinates of its corners, in random order. Your task is to find the area of the square.

Input

Each test consists of several testcases. The first line contains one integer t ($1 \le t \le 100$) — the number of testcases. The following is a description of the testcases.

Each testcase contains four lines, each line contains two integers x_i, y_i ($-1000 \le x_i, y_i \le 1000$), coordinates of the corners of the square.

It is guaranteed that there is a square with sides parallel to the coordinate axes, with positive (strictly greater than 0) area, with corners in given points.

Output

For each test case, print a single integer, the area of the square.

Standard Input	Standard Output
3 1 2 4 5 1 5 4 2 -1 1 1 -1 1 1 -1 -1 45 11 45 39 17 11 17 39	9 4 784

,

B. Arranging Cats

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 512 megabytes

In order to test the hypothesis about the cats, the scientists must arrange the cats in the boxes in a specific way. Of course, they would like to test the hypothesis and publish a sensational article as quickly as possible, because they are too engrossed in the next hypothesis about the phone's battery charge.

Scientists have n boxes in which cats may or may not sit. Let the current state of the boxes be denoted by the sequence $b_1, \ldots, b_n$: $b_i = 1$ if there is a cat in box number i, and $b_i = 0$ otherwise.

Fortunately, the unlimited production of cats has already been established, so in one day, the scientists can perform one of the following operations:

- Take a new cat and place it in a box (for some i such that $b_i = 0$, assign $b_i = 1$).
- Remove a cat from a box and send it into retirement (for some i such that $b_i = 1$, assign $b_i = 0$).
- Move a cat from one box to another (for some i, j such that $b_i = 1, b_j = 0$, assign $b_i = 0, b_j = 1$).

It has also been found that some boxes were immediately filled with cats. Therefore, the scientists know the initial position of the cats in the boxes $s_1, \ldots, s_n$ and the desired position $f_1, \ldots, f_n$.

Due to the large amount of paperwork, the scientists do not have time to solve this problem. Help them for the sake of science and indicate the minimum number of days required to test the hypothesis.

Input

Each test consists of several test cases. The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. This is followed by descriptions of the test cases.

Each test case consists of three lines.

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

The second line of each test case contains a string s of n characters, where the i-th character is '1' if there is a cat in the i-th box and '0' otherwise.

The third line of each test case contains a string f of n characters, where the i-th character is '1' if there should be a cat in the i-th box and '0' otherwise.

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

Output

For each test case, output a single integer on a separate line — the minimum number of operations required to obtain the desired position from the initial position. It can be shown that a solution always exists.

Standard Input	Standard Output
6 5 10010 00001 1	2 0 3 2

1 1 3 000 111 4 0101 1010 3 100 101 8 10011001 11111110	1 4

Note

In the first test case, you can first move the cat from the first box to the fifth, and then remove the cat from the fourth box.

In the second test case, there is nothing to do — the only cat is already sitting in the correct box.

In the third test case of input data, it takes three days to place a cat in each box.

,

C. Sending Messages

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

Stepan is a very busy person. Today he needs to send n messages at moments $m_1, m_2, \ldots m_n$ ($m_i < m_{i+1}$). Unfortunately, by the moment 0, his phone only has f units of charge left. At the moment 0, the phone is turned on.

The phone loses a units of charge for each unit of time it is on. Also, at any moment, Stepan can turn off the phone and turn it on later. This action consumes b units of energy each time. Consider turning on and off to be instantaneous, so you can turn it on at moment x and send a message at the same moment, and vice versa, send a message at moment x and turn off the phone at the same moment.

If at any point the charge level drops to 0 (becomes ≤ 0), it is impossible to send a message at that moment.

Since all messages are very important to Stepan, he wants to know if he can send all the messages without the possibility of charging the phone.

Input

The first line of the input contains a single integer t ($1 \leq t \leq 10^4$) — the number of test cases. This is followed by the descriptions of the test cases.

The first line of each test case contains four integers n, f, a, and b ($1 \leq n \leq 2 \cdot 10^5$, $1 \leq f, a, b \leq 10^9$) — the number of messages, the initial phone's charge, the charge consumption per unit of time, and the consumption when turned off and on sequentially.

The second line of each test case contains n integers $m_1, m_2, \ldots, m_n$ ($1 \leq m_i \leq 10^9$, $m_i < m_{i+1}$) — the moments at which messages need to be sent.

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

Output

For each test case, output "YES" if Stepan can send all the messages, and "NO" otherwise.

You can output each letter in any case (lowercase or uppercase). For example, the strings "yEs", "yes", "Yes", and "YES" will be accepted as a positive answer.

Standard Input	Standard Output
6 1 3 1 5 3 7 21 1 3 4 6 10 13 17 20 26 5 10 1 2 1 2 3 4 5 1 1000000000 1000000000 1000000000 1000000000 3 11 9 6 6 8 10	NO YES YES NO NO YES

12 621526648 2585904 3566299 51789 61859 71998 73401 247675 298086 606959 663464 735972 806043 806459 919683	

Note

In the first test case of the example, at moment 0, the phone's charge is 3. When sending a message at moment 3 without turning it off, $(3 - 0) \cdot 1 = 3$ units of charge will be spent. In this case, the charge will drop to 0 and Stepan will not be able to send the message. When turning off and on, the phone's charge will decrease by 5, so it will not be possible to send the message in this way.

In the third test case of the example, at moment 0, the phone's charge is 10. The phone loses 1 unit of charge per unit of time, and when turned off and on, it loses 2 units of charge. To send all messages, the following actions can be taken:

- Turn off the phone at moment 0 and turn it on at moment 1, after which $10 - 2 = 8$ units of charge will remain;
- send a message at moment 1;
- send a message at moment 2, after which $8 - (2 - 1) \cdot 1 = 7$ units of charge will remain;
- Turn off the phone at moment 2 and turn it on at moment 3, after which $7 - 2 = 5$ units of charge will remain;
- send a message at moment 3;
- Turn off the phone at moment 3 and turn it on at moment 4, after which $5 - 2 = 3$ units of charge will remain;
- send a message at moment 4;
- Turn off the phone at moment 4 and turn it on at moment 5, after which $3 - 2 = 1$ unit of charge will remain;
- send a message at moment 5.

The last (sixth) test set of the example may fail if there is an integer overflow in your solution.

,

D. Very Different Array

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

Petya has an array a_i of n integers. His brother Vasya became envious and decided to make his own array of n integers.

To do this, he found m integers b_i ($m \geq n$), and now he wants to choose some n integers of them and arrange them in a certain order to obtain an array c_i of length n.

To avoid being similar to his brother, Vasya wants to make his array as different as possible from Petya's array. Specifically, he wants the total difference $D = \sum_{i=1}^{n} |a_i - c_i|$ to be as large as possible.

Help Vasya find the maximum difference D he can obtain.

Input

Each test consists of multiple test cases. The first line contains a single integer t ($1 \leq t \leq 100$) — the number of test cases. This is followed by a description of the test cases.

The first line of each test case contains two integers n and m ($1 \leq n \leq m \leq 2 \cdot 10^5$).

The second line of each test case contains n integers a_i ($1 \leq a_i \leq 10^9$). The third line of each test case contains m integers b_i ($1 \leq b_i \leq 10^9$).

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

Output

For each test case, output a single integer — the maximum total difference D that can be obtained.

Standard Input	Standard Output
9 4 6 6 1 2 4 3 5 1 7 2 3 3 4 1 1 1 1 1 1 1 5 5 1 2 3 4 5 1 2 3 4 5 2 6 5 8 8 7 5 8 2 10 2 2 4 1 9 6 4 6 8 10 6 4	16 0 12 11 10 23 15 25 7

3 10 6 1 8 9 3 5 6 5 2 1 7 9 7 2 5 5 9 10 6 3 7 5 9 2 3 9 1 6 3 2 7 10 1 1 5	

Note

In the first example, Vasya can, for example, create the array $(1, 5, 7, 2)$. Then the total difference will be $D = |6 - 1| + |1 - 5| + |2 - 7| + |4 - 2| = 5 + 4 + 5 + 2 = 16$.

In the second example, all the integers available to Vasya are equal to 1, so he can only create the array $(1, 1, 1)$, for which the difference $D = 0$.

In the third example, Vasya can, for example, create the array $(5, 4, 3, 2, 1)$. Then the total difference will be $D = |1 - 5| + |2 - 4| + |3 - 3| + |4 - 2| + |5 - 1| = 4 + 2 + 0 + 2 + 4 = 12$.

,

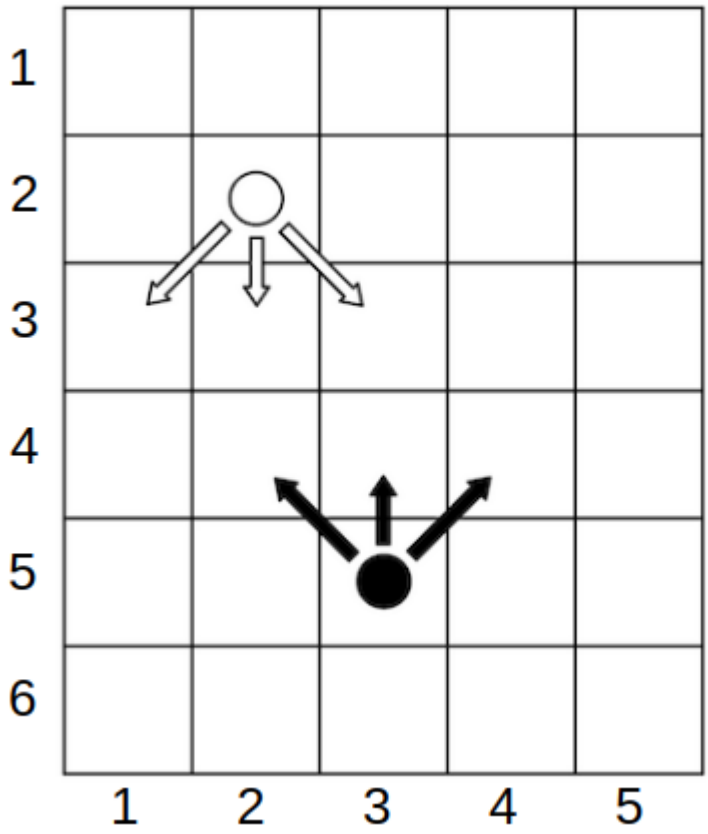
E. Eat the Chip

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

Alice and Bob are playing a game on a checkered board. The board has h rows, numbered from top to bottom, and w columns, numbered from left to right. Both players have a chip each. Initially, Alice's chip is located at the cell with coordinates (x_a, y_a) (row x_a, column y_a), and Bob's chip is located at (x_b, y_b). It is guaranteed that the initial positions of the chips do not coincide. Players take turns making moves, with Alice starting.

On her turn, Alice can move her chip one cell down or one cell down-right or down-left (diagonally). Bob, on the other hand, moves his chip one cell up, up-right, or up-left. It is not allowed to make moves that go beyond the board boundaries.

More formally, if at the beginning of Alice's turn she is in the cell with coordinates (x_a, y_a), then she can move her chip to one of the cells $(x_a + 1, y_a)$, $(x_a + 1, y_a - 1)$, or $(x_a + 1, y_a + 1)$. Bob, on his turn, from the cell (x_b, y_b) can move to $(x_b - 1, y_b)$, $(x_b - 1, y_b - 1)$, or $(x_b - 1, y_b + 1)$. The new chip coordinates (x', y') must satisfy the conditions $1 \le x' \le h$ and $1 \le y' \le w$.



Example game state. Alice plays with the white chip, Bob with the black one. Arrows indicate possible moves.

A player immediately wins if they place their chip in a cell occupied by the other player's chip. If either player cannot make a move (Alice—if she is in the last row, i.e. $x_a = h$, Bob—if he is in the first row, i.e. $x_b = 1$), the game immediately ends in a draw.

What will be the outcome of the game if both opponents play optimally?

Input

Each test consists of multiple test cases. The first line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases. This is followed by the description of the test cases.

Each test case consists of a single line containing six integers h, w, x_a, y_a, x_b, y_b ($1 \le x_a, x_b \le h \le 10^6$, $1 \le y_a, y_b \le w \le 10^9$) — the dimensions of the board and the initial positions of Alice's and Bob's chips. It is guaranteed that either $x_a \neq x_b$ or $y_a \neq y_b$.

It is guaranteed that the sum of h over all test cases does not exceed 10^6.

Output

For each test case, output `"Alice"` if Alice wins, `"Bob"` if Bob wins, and `"Draw"` if neither player can secure a victory. You can output each letter in any case (lowercase or uppercase). For example, the strings `"bOb"`, `"bob"`, `"Bob"`, and `"BOB"` will be accepted as Bob's victory.

Standard Input	Standard Output
12 6 5 2 2 5 3 4 1 2 1 4 1 1 4 1 3 1 1 5 5 1 4 5 2 4 4 1 1 4 4 10 10 1 6 10 8 10 10 2 6 10 7 10 10 9 1 8 1 10 10 8 1 10 2 10 10 1 1 2 1 10 10 1 3 4 1 10 10 3 1 1 1	Alice Bob Draw Draw Draw Alice Draw Draw Bob Alice Alice Draw

,

F. Sum of Progression

Input file: standard input
Output file: standard output
Time limit: 2 seconds
Memory limit: 1024 megabytes

You are given an array a of n numbers. There are also q queries of the form s, d, k.

For each query q, find the sum of elements $a_s + a_{s+d} \cdot 2 + \cdots + a_{s+d \cdot (k-1)} \cdot k$. In other words, for each query, it is necessary to find the sum of k elements of the array with indices starting from the s-th, taking steps of size d, multiplying it by the serial number of the element in the resulting sequence.

Input

Each test consists of several testcases. The first line contains one integer t ($1 \le t \le 10^4$) — the number of testcases. Next lines contain descriptions of testcases.

The first line of each testcase contains two numbers n, q ($1 \le n \le 10^5, 1 \le q \le 2 \cdot 10^5$) — the number of elements in the array a and the number of queries.

The second line contains n integers $a_1, \ldots a_n$ ($-10^8 \le a_1, \ldots, a_n \le 10^8$) — elements of the array a.

The next q lines each contain three integers s, d, and k ($1 \le s, d, k \le n$, $s + d \cdot (k - 1) \le n$).

It is guaranteed that the sum of n over all testcases does not exceed 10^5, and that the sum of q over all testcases does not exceed $2 \cdot 10^5$.

Output

For each testcase, print q numbers in a separate line — the desired sums, separated with space.

Standard Input	Standard Output
5 3 3 1 1 2 1 2 2 2 2 1 1 1 2 3 1 -100000000 -100000000 -100000000 1 1 3 5 3 1 2 3 4 5 1 2 3 2 3 2 1 1 5 3 1 100000000 100000000 100000000 1 1 3 7 7 34 87 5 42 -44 66 -32 2 2 2	5 1 3 -600000000 22 12 55 600000000 171 42 118 66 -108 23 2

4 3 1 1 3 2 6 2 1 5 2 2 2 5 2 6 1 2	

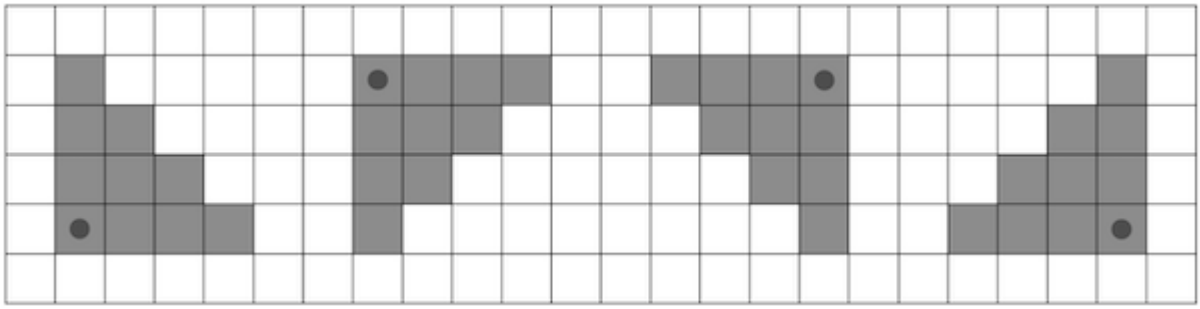
,

G. Mischievous Shooter

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

Once the mischievous and wayward shooter named Shel found himself on a rectangular field of size $n \times m$, divided into unit squares. Each cell either contains a target or not.

Shel only had a lucky shotgun with him, with which he can shoot in one of the four directions: right-down, left-down, left-up, or right-up. When fired, the shotgun hits all targets in the chosen direction, the Manhattan distance to which does not exceed a fixed constant k. The Manhattan distance between two points (x_1, y_1) and (x_2, y_2) is equal to $|x_1 - x_2| + |y_1 - y_2|$.



Possible hit areas for $k = 3$.

Shel's goal is to hit as many targets as possible. Please help him find this value.

Input

Each test consists of several test cases. The first line contains a single integer t ($1 \le t \le 1000$) — the number of test cases. Then follows the description of the test cases.

The first line of each test case contains field dimensions n, m, and the constant for the shotgun's power k ($1 \le n, m, k \le 10^5, 1 \le n \cdot m \le 10^5$).

Each of the next n lines contains m characters — the description of the next field row, where the character '`.`' means the cell is empty, and the character '#' indicates the presence of a target.

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

Output

For each test case, output a single integer on a separate line, which is equal to the maximum possible number of hit targets with one shot.

Standard Input	Standard Output
4 3 3 1 .#. ### .#. 2 5 3 ###.. ...## 4 4 2	3 4 5 2

..## ###. #..# #### 2 1 3 # #	

Note

Possible optimal shots for the examples in the statement: