Codeforces Round #753 (Div. 3)

A. Linear Keyboard

1 second, 256 megabytes

You are given a keyboard that consists of 26 keys. The keys are arranged sequentially in one row in a certain order. Each key corresponds to a unique lowercase Latin letter.

You have to type the word s on this keyboard. It also consists only of lowercase Latin letters.

To type a word, you need to type all its letters consecutively one by one. To type each letter you must position your hand exactly over the corresponding key and press it.

Moving the hand between the keys takes time which is equal to the absolute value of the difference between positions of these keys (the keys are numbered from left to right). No time is spent on pressing the keys and on placing your hand over the first letter of the word.

For example, consider a keyboard where the letters from 'a' to 'z' are arranged in consecutive alphabetical order. The letters 'h', 'e', 'l' and 'o' then are on the positions 8, 5, 12 and 15, respectively. Therefore, it will take $|5 - 8| + |12 - 5| + |12 - 12| + |15 - 12| = 13$ units of time to type the word "hello".

Determine how long it will take to print the word s.

Input

The first line contains an integer t ($1 \le t \le 1000$) — the number of test cases.

The next $2t$ lines contain descriptions of the test cases.

The first line of a description contains a keyboard — a string of length 26, which consists only of lowercase Latin letters. Each of the letters from 'a' to 'z' appears exactly once on the keyboard.

The second line of the description contains the word s. The word has a length from 1 to 50 letters inclusive and consists of lowercase Latin letters.

Output

Print t lines, each line containing the answer to the corresponding test case. The answer to the test case is the minimal time it takes to type the word s on the given keyboard.

input

```
5
abcdefghijklmnopqrstuvwxyz
hello
abcdefghijklmnopqrstuvwxyz
i
abcdefghijklmnopqrstuvwxyz
codeforces
qwertyuiopasdfghjklzxcvbnm
qqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqqq
qwertyuiopasdfghjklzxcvbnm
abacaba
```

output

```
13
0
68
0
74
```

B. Odd Grasshopper

1 second, 256 megabytes

The grasshopper is located on the numeric axis at the point with coordinate x_0.

Having nothing else to do he starts jumping between integer points on the axis. Making a jump from a point with coordinate x with a distance d to the left moves the grasshopper to a point with a coordinate $x - d$, while jumping to the right moves him to a point with a coordinate $x + d$.

The grasshopper is very fond of positive integers, so for each integer i starting with 1 the following holds: exactly i minutes after the start he makes a jump with a distance of exactly i. So, in the first minutes he jumps by 1, then by 2, and so on.

The direction of a jump is determined as follows: if the point where the grasshopper was before the jump has an **even** coordinate, the grasshopper jumps to the **left**, **otherwise** he jumps to the **right**.

For example, if after 18 consecutive jumps he arrives at the point with a coordinate 7, he will jump by a distance of 19 to the right, since 7 is an odd number, and will end up at a point $7 + 19 = 26$. Since 26 is an even number, the next jump the grasshopper will make to the left by a distance of 20, and it will move him to the point $26 - 20 = 6$.

Find exactly which point the grasshopper will be at after exactly n jumps.

Input

The first line of input contains an integer t ($1 \le t \le 10^4$) — the number of test cases.

Each of the following t lines contains two integers x_0 ($-10^{14} \le x_0 \le 10^{14}$) and n ($0 \le n \le 10^{14}$) — the coordinate of the grasshopper's initial position and the number of jumps.

Output

Print exactly t lines. On the i-th line print one integer — the answer to the i-th test case — the coordinate of the point the grasshopper will be at after making n jumps from the point x_0.

input

```
9
0 1
0 2
10 10
10 99
177 13
10000000000 987654321
-433494437 87178291199
1 0
-1 1
```

output

```
-1
1
11
110
190
9012345679
-87611785637
1
0
```

The first two test cases in the example correspond to the first two jumps from the point $x_0 = 0$.

Since 0 is an even number, the first jump of length 1 is made to the left, and the grasshopper ends up at the point $0 - 1 = -1$.

Then, since -1 is an odd number, a jump of length 2 is made to the right, bringing the grasshopper to the point with coordinate $-1 + 2 = 1$.

C. Minimum Extraction

1 second, 256 megabytes

Yelisey has an array a of n integers.

If a has length strictly greater than 1, then Yelisei can apply an operation called *minimum extraction* to it:

1. First, Yelisei finds the minimal number m in the array. If there are

several identical minima, Yelisey can choose any of them.
2. Then the selected minimal element is removed from the array. After that, m is subtracted from each remaining element.

Thus, after each operation, the length of the array is reduced by 1.

For example, if $a = [1, 6, -4, -2, -4]$, then the minimum element in it is $a_3 = -4$, which means that after this operation the array will be equal to $a = [1-(-4), 6-(-4), -2-(-4), -4-(-4)] = [5, 10, 2, 0]$.

Since Yelisey likes big numbers, he wants the numbers in the array a to be as big as possible.

Formally speaking, he wants to make the **minimum** of the numbers in array a to be **maximal possible** (i.e. he want to maximize a minimum). To do this, Yelisey can apply the *minimum extraction* operation to the array as many times as he wants (possibly, zero). Note that the operation cannot be applied to an array of length 1.

Help him find what maximal value can the minimal element of the array have after applying several (possibly, zero) *minimum extraction* operations to the array.

Input

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

The next $2t$ lines contain descriptions of the test cases.

In the description of each test case, the first line contains an integer n ($1 \le n \le 2 \cdot 10^5$) — the original length of the array a. The second line of the description lists n space-separated integers a_i ($-10^9 \le a_i \le 10^9$) — elements of the array a.

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

Output

Print t lines, each of them containing the answer to the corresponding test case. The answer to the test case is a single integer — the maximal possible minimum in a, which can be obtained by several applications of the described operation to it.

input

```
8
1
10
2
0 0
3
-1 2 0
4
2 10 1 7
2
2 3
5
3 2 -4 -2 0
2
-1 1
1
-2
```

output

```
10
0
2
5
2
2
2
-2
```

In the first example test case, the original length of the array $n = 1$. Therefore *minimum extraction* cannot be applied to it. Thus, the array remains unchanged and the answer is $a_1 = 10$.

In the second set of input data, the array will always consist only of zeros.

In the third set, the array will be changing as follows: $[-1, 2, 0] \to [3, 1] \to [2]$. The minimum elements are highlighted with blue. The maximal one is 2.

In the fourth set, the array will be modified as $[2, 10, 1, 7] \to [1, 9, 6] \to [8, 5] \to [3]$. Similarly, the maximum of the minimum elements is 5.

The problem statement has recently been changed. View the changes. ×

D. Blue-Red Permutation

1 second, 256 megabytes

You are given an array of integers a of length n. The elements of the array can be either different or the same.

Each element of the array is colored either blue or red. There are no unpainted elements in the array. One of the two operations described below can be applied to an array in a single step:

- either you can select any blue element and decrease its value by 1;
- or you can select any red element and increase its value by 1.

Situations in which there are no elements of some color at all are also possible. For example, if the whole array is colored blue or red, one of the operations becomes unavailable.

Determine whether it is possible to make 0 or more steps such that the resulting array is a permutation of numbers from 1 to n?

In other words, check whether there exists a sequence of steps (possibly empty) such that after applying it, the array a contains in some order all numbers from 1 to n (inclusive), each exactly once.

Input

The first line contains an integer t ($1 \le t \le 10^4$) — the number of input data sets in the test.

The description of each set of input data consists of three lines. The first line contains an integer n ($1 \le n \le 2 \cdot 10^5$) — the length of the original array a. The second line contains n integers $a_1, a_2, ..., a_n$ ($-10^9 \le a_i \le 10^9$) — the array elements themselves.

The third line has length n and consists exclusively of the letters 'B' and/or 'R': ith character is 'B' if a_i is colored blue, and is 'R' if colored red.

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

Output

Print t lines, each of which contains the answer to the corresponding test case of the input. Print YES as an answer if the corresponding array can be transformed into a permutation, and NO otherwise.

You can print the answer in any case (for example, the strings yES, yes, Yes, and YES will be recognized as a positive answer).

input

```
8
4
1 2 5 2
BRBR
2
1 1
BB
5
3 1 4 2 5
RBRRB
5
3 1 3 1 3
RBRRB
5
5 1 5 1 5
RBRRB
4
2 2 2 2
BRBR
2
1 -2
BR
4
-2 -1 4 0
RRRR
```

output

```
YES
NO
YES
YES
NO
YES
YES
YES
```

In the first test case of the example, the following sequence of moves can be performed:

- choose $i = 3$, element $a_3 = 5$ is blue, so we decrease it, we get $a = [1, 2, 4, 2]$;
- choose $i = 2$, element $a_2 = 2$ is red, so we increase it, we get $a = [1, 3, 4, 2]$;
- choose $i = 3$, element $a_3 = 4$ is blue, so we decrease it, we get $a = [1, 3, 3, 2]$;
- choose $i = 2$, element $a_2 = 2$ is red, so we increase it, we get $a = [1, 4, 3, 2]$.

We got that a is a permutation. Hence the answer is YES.

The problem statement has recently been changed. View the changes. ×

E. Robot on the Board 1

2 seconds, 256 megabytes

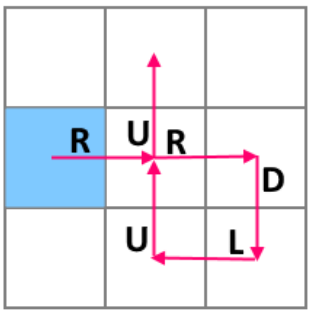
The robot is located on a checkered rectangular board of size $n \times m$ (n rows, m columns). The rows in the board are numbered from 1 to n from top to bottom, and the columns — from 1 to m from left to right.

The robot is able to move from the current cell to one of the four cells adjacent by side.

The sequence of commands s executed by the robot is given. Each command is denoted by one of the symbols 'L', 'R', 'D' or 'U', and triggers the movement to left, right, down or up, respectively.

The robot can start its movement in **any** cell. The robot executes the commands starting from the first one, strictly in the order in which they are listed in s. If the robot moves beyond the edge of the board, it falls and breaks. A command that causes the robot to break is **not considered** successfully executed.

The robot's task is to execute as many commands as possible without falling off the board. For example, on board 3×3, if the robot starts a sequence of actions s ="RRDLUU" ("right", "right", "down", "left", "up", "up") from the central cell, the robot will perform one command, then the next command will force him to cross the edge. If the robot starts moving from the cell $(2, 1)$ (second row, first column) then all commands will be executed successfully and the robot will stop at the cell $(1, 2)$ (first row, second column).



The robot starts from cell $(2, 1)$ (second row, first column). It moves right, right, down, left, up, and up. In this case it ends in the cell $(1, 2)$ (first row, second column).

Determine the cell from which the robot should start its movement in order to execute as many commands as possible.

Input

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

The next $2t$ lines contain descriptions of the test cases.

In the description of each test case, the first line contains two integers n and m ($1 \le n, m \le 10^6$) — the height and width of the field that the robot is located on. The second line of the description is a string s consisting solely of characters 'L', 'R', 'D' and 'U' — the sequence of commands the robot executes. The string has a length from 1 to 10^6 commands.

It is guaranteed that the total length of s over all test cases does not exceed 10^6.

Output

Print t lines, each of which contains the answer to the corresponding test case. The answer to the test case are two integers r ($1 \le r \le n$) and c ($1 \le c \le m$), separated by a space — the coordinates of the cell (row number and column number) from which the robot should start moving to perform as many commands as possible.

If there are several such cells, you may output any of them.

input

```
4
1 1
L
1 2
L
3 3
RRDLUU
4 3
LUURRDDLLLUU
```

output

```
1 1
1 2
2 1
3 2
```

F. Robot on the Board 2

2 seconds, 256 megabytes

The robot is located on a checkered rectangular board of size $n \times m$ (n rows, m columns). The rows in the board are numbered from 1 to n from top to bottom, and the columns — from 1 to m from left to right.

The robot is able to move from the current cell to one of the four cells adjacent by side.

Each cell has one of the symbols 'L', 'R', 'D' or 'U' written on it, indicating the direction in which the robot will move when it gets in that cell — left, right, down or up, respectively.

The robot can start its movement in any cell. He then moves to the adjacent square in the direction indicated on the current square in one move.

- If the robot moves beyond the edge of the board, it falls and breaks.
- If the robot appears in the cell it already visited before, it breaks (it stops and doesn't move anymore).

Robot can choose any cell as the starting cell. Its goal is to make the maximum number of steps before it breaks or stops.

Determine from which square the robot should start its movement in order to execute as many commands as possible. A command is considered successfully completed if the robot has moved from the square on which that command was written (it does not matter whether to another square or beyond the edge of the board).

Input

The first line contains an integer t ($1 \le t \le 10000$) — the number of test cases in the test.

Each test case's description is preceded by a blank line. Next is a line that contains integers n and m ($1 \le n \le 2000; 1 \le m \le 2000$) — the height and width of the board. This line followed by n lines, the i-th of which describes the i-th line of the board. Each of them is exactly m letters long and consists of symbols 'L', 'R', 'D' and 'U'.

It is guaranteed that the sum of sizes of all boards in the input does not exceed $4 \cdot 10^6$.

Output

For each test case, output three integers r, c and d ($1 \le r \le n$; $1 \le c \le m$; $d \ge 0$), which denote that the robot should start moving from cell (r, c) to make the maximum number of moves d. If there are several answers, output any of them.

input

```
7

1 1
R

1 3
RRL

2 2
DL
RU

2 2
UD
RU

3 2
DL
UL
RU

4 4
RRRD
RUUD
URUD
ULLR

4 4
DDLU
RDDU
UUUU
RDLD
```

output

```
1 1 1
1 1 3
1 1 4
2 1 3
3 1 5
4 3 12
1 1 4
```

G. Banquet Preparations 1

2 seconds, 256 megabytes

A known chef has prepared n dishes: the i-th dish consists of a_i grams of fish and b_i grams of meat.

The banquet organizers estimate the *balance* of n dishes as follows. The *balance* is equal to the absolute value of the difference between the total mass of fish and the total mass of meat.

Technically, the *balance* equals to $\left|\sum_{i=1}^{n} a_i - \sum_{i=1}^{n} b_i\right|$. The smaller the *balance*, the better.

In order to improve the *balance*, a taster was invited. He will eat **exactly** m grams of food from each dish. For each dish, the taster determines separately how much fish and how much meat he will eat. The only condition is that he should eat exactly m grams of each dish in total.

Determine how much of what type of food the taster should eat from each dish so that the value of the *balance* is as minimal as possible. If there are several correct answers, you may choose any of them.

Input

The first line of input data contains an integer t ($1 \le t \le 10^4$) — the number of the test cases.

Each test case's description is preceded by a blank line. Next comes a line that contains integers n and m ($1 \le n \le 2 \cdot 10^5; 0 \le m \le 10^6$). The next n lines describe dishes, the i-th of them contains a pair of integers a_i and b_i ($0 \le a_i, b_i \le 10^6$) — the masses of fish and meat in the i-th dish.

It is guaranteed that it is possible to eat m grams of food from each dish. In other words, $m \le a_i + b_i$ for all i from 1 to n inclusive.

The sum of all n values over all test cases in the test does not exceed $2 \cdot 10^5$.

Output

For each test case, print on the first line the minimal balance value that can be achieved by eating exactly m grams of food from each dish.

Then print n lines that describe a way to do this: the i-th line should contain two integers x_i and y_i ($0 \le x_i \le a_i; 0 \le y_i \le b_i$; $x_i + y_i = m$), where x_i is how many grams of fish taster should eat from the i-th meal and y_i is how many grams of meat.

If there are several ways to achieve a minimal balance, find any of them.

input

```
8

1 5
3 4

1 6
3 4

2 2
1 3
4 2

2 4
1 3
1 7

3 6
1 7
1 8
1 9

3 6
1 8
1 9
30 10

3 4
3 1
3 2
4 1

5 4
0 7
6 4
0 8
4 1
5 3
```

output

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

H. Banquet Preparations 2

3 seconds, 256 megabytes

The chef has cooked n dishes yet again: the i-th dish consists of a_i grams of fish and b_i grams of meat.

Banquet organizers consider two dishes i and j equal if $a_i = a_j$ and $b_i = b_j$ at the same time.

The banquet organizers estimate the *variety* of n dishes as follows. The *variety* of a set of dishes is equal to the number of different dishes in it. The **less** *variety* is, the **better**.

In order to reduce the *variety*, a taster was invited. He will eat **exactly** m_i grams of food from each dish. For each dish, the taster determines separately how much fish and how much meat he will eat. The only condition is that he will eat exactly m_i grams of the i-th dish in total.

Determine how much of what type of food the taster should eat from each dish so that the value of *variety* is the minimum possible. If there are several correct answers, you may output any of them.

Input

The first line of input data contains an integer t ($1 \le t \le 10^4$) — the number of test cases.

Each test case's description is preceded by a blank line. Next comes a line that contains an integer n ($1 \le n \le 2 \cdot 10^5$) — the number of dishes. Then follows n lines, i-th of which contains three integers a_i, b_i and m_i ($0 \le a_i, b_i \le 10^6$; $0 \le m_i \le a_i + b_i$) — the mass of fish in i-th dish, the mass of meat in i-th dish and how many grams in total the taster should eat in i-th dish.

The sum of all n values for all input data sets in the test does not exceed $2 \cdot 10^5$.

Output

For each test case, print on the first line the minimum value of *variety* that can be achieved by eating exactly m_i grams of food (for all i from 1 to n) from a dish i.

Then print n lines that describe a way to do this: the i-th line should contain two integers x_i and y_i ($0 \le x_i \le a_i$; $0 \le y_i \le b_i$; $x_i + y_i = m_i$), where x_i is how many grams of fish the taster should eat from i-th dish, and y_i is how many grams of meat.

If there are several ways to achieve a minimum balance, print any of them.

input

```
5

3
10 10 2
9 9 0
10 9 1

2
3 4 1
5 1 2

3
7 2 5
6 5 4
5 5 6

1
13 42 50

5
5 7 12
3 1 4
7 3 7
0 0 0
4 1 5
```

output

```
1
1 1
0 0
1 0
2
0 1
1 1
2
3 2
0 4
1 5
1
8 42
2
5 7
3 1
4 3
0 0
4 1
```