A. Phone Desktop

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

Little Rosie has a phone with a desktop (or launcher, as it is also called). The desktop can consist of several screens. Each screen is represented as a grid of size 5×3, i.e., five rows and three columns.

There are x applications with an icon size of 1×1 cells; such an icon occupies only one cell of the screen. There are also y applications with an icon size of 2×2 cells; such an icon occupies a **square** of 4 cells on the screen. Each cell of each screen can be occupied by no more than one icon.

Rosie wants to place the application icons on the minimum number of screens. Help her find the minimum number of screens needed.

Input

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

The first and only line of each test case contains two integers x and y ($0 \le x, y \le 99$) — the number of applications with a 1×1 icon and the number of applications with a 2×2 icon, respectively.

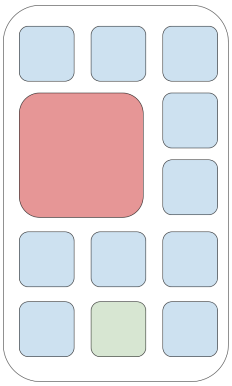
Output

For each test case, output the minimal number of required screens on a separate line.

Standard Input	Standard Output
11 1 1 7 2 12 4 0 3 1 0 8 1 0 0 2 0 15 0 8 2 0 9	1 1 2 2 1 1 0 1 1 2 5

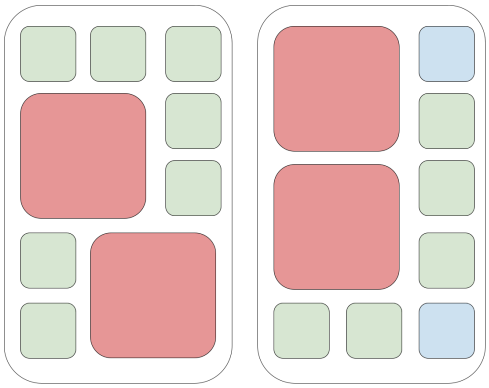
Note

The solution for the first test case can look as follows:



Blue squares represent empty spaces for icons, green squares represent 1×1 icons, red squares represent 2×2 icons

The solution for the third test case can look as follows:



,

B. Symmetric Encoding

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

Polycarp has a string s, which consists of lowercase Latin letters. He encodes this string using the following algorithm:

- first, he constructs a new auxiliary string r, which consists of all distinct letters of the string s, written in alphabetical order;
- then the encoding happens as follows: each character in the string s is replaced by its symmetric character from the string r (the first character of the string r will be replaced by the last, the second by the second from the end, and so on).

For example, encoding the string s="codeforces" happens as follows:

- the string r is obtained as "cdefors";
- the first character s_1='c' is replaced by 's';
- the second character s_2='o' is replaced by 'e';
- the third character s_3='d' is replaced by 'r';
- ...
- the last character s_{10}='s' is replaced by 'c'.

The string r and replacements for s="codeforces".

Thus, the result of encoding the string s="codeforces" is the string "serofedsoc".

Write a program that performs decoding — that is, restores the original string s from the encoding result.

Input

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

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

The second line of each test case contains a string b of length n, consisting of lowercase Latin letters — the result of encoding the original string s.

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

Output

For each test case, output the string s from which the encoding result b was obtained.

Standard Input	Standard Output
5 10 serofedsoc	codeforces fft algorithm

3 ttf 9 tlrhgmaoi 1 w 15 hnndledmnhlttin	w meetinthemiddle

,

C. Beautiful Triple Pairs

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

Polycarp was given an array a of n integers. He really likes triples of numbers, so for each j ($1 \le j \le n - 2$) he wrote down a triple of elements $[a_j, a_{j+1}, a_{j+2}]$.

Polycarp considers a pair of triples b and c *beautiful* if they differ in exactly one position, that is, one of the following conditions is satisfied:

- $b_1 \ne c_1$ and $b_2 = c_2$ and $b_3 = c_3$;
- $b_1 = c_1$ and $b_2 \ne c_2$ and $b_3 = c_3$;
- $b_1 = c_1$ and $b_2 = c_2$ and $b_3 \ne c_3$.

Find the number of *beautiful* pairs of triples among the written triples $[a_j, a_{j+1}, a_{j+2}]$.

Input

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

The first line of each test case contains a single integer n ($3 \le n \le 2 \cdot 10^5$) — the length of the array a.

The second line of each test case contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^6$) — the elements of the array.

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

Output

For each test case, output a single integer — the number of *beautiful* pairs of triples among the pairs of the form $[a_j, a_{j+1}, a_{j+2}]$.

Note that the answer may not fit into 32-bit data types.

Standard Input	Standard Output
8 5 3 2 2 2 3 5 1 2 1 2 1 8 1 2 3 2 2 3 4 2 4 2 1 1 1 8 2 1 1 2 1 1 1 1 7 2 1 1 1 1 1 1 6 2 1 1 1 1 1 5	2 0 3 1 8 4 3 2

2 1 1 1 1	

Note

In the first example, $a = [3, 2, 2, 2, 3]$, Polycarp will write the following triples:

1. $[3, 2, 2]$;
2. $[2, 2, 2]$;
3. $[2, 2, 3]$.

The beautiful pairs are triple 1 with triple 2 and triple 2 with triple 3.
In the third example, $a = [1, 2, 3, 2, 2, 3, 4, 2]$, Polycarp will write the following triples:

1. $[1, 2, 3]$;
2. $[2, 3, 2]$;
3. $[3, 2, 2]$;
4. $[2, 2, 3]$;
5. $[2, 3, 4]$;
6. $[3, 4, 2]$;

The beautiful pairs are triple 1 with triple 4, triple 2 with triple 5, and triple 3 with triple 6.

,

D. Ingenuity-2

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

Let's imagine the surface of Mars as an infinite coordinate plane. Initially, the rover Perseverance-2 and the helicopter Ingenuity-2 are located at the point with coordinates $(0, 0)$. A set of instructions s consisting of n instructions of the following types was specially developed for them:

- N: move one meter north (from point (x, y) to $(x, y + 1)$);
- S: move one meter south (from point (x, y) to $(x, y - 1)$);
- E: move one meter east (from point (x, y) to $(x + 1, y)$);
- W: move one meter west (from point (x, y) to $(x - 1, y)$).

Each instruction must be executed either by the rover or by the helicopter. Moreover, each device must execute **at least one** instruction. Your task is to distribute the instructions in such a way that after executing all n instructions, the helicopter and the rover end up at the same point, or determine that this is impossible.

Input

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

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

The second line of each test case contains a string s of length n consisting of the characters 'N', 'S', 'E', 'W' — the sequence of instructions.

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

Output

For each test case, if the required distribution of instructions exists, output a string p of length n consisting of the characters 'R', 'H'. If the i-th operation should be executed by the rover, then $p_i = \mathrm{R}$, if the i-th operation should be executed by the helicopter, then $p_i = \mathrm{H}$. If there are multiple solutions, output any of them.

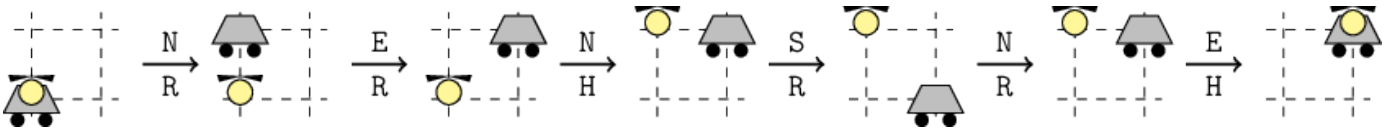
Otherwise, output NO.

Standard Input	Standard Output
10 6 NENSNE 3 WWW 6 NESSWS 2 SN 2 WE 4 SSNN	RRHRRH NO HRRHRH NO NO RHRH RRHH RH RRRH RRHH

4 WESN 2 SS 4 EWNN 4 WEWE	

Note

Let's consider the first example: the string $S =$ NENSNE. One of the possible solutions, shown in the figure below, is $p =$ RRHRRH, using which both the rover and the helicopter will end up one meter north and one meter east.



For WWW, the solution is impossible.

,

E. Money Buys Happiness

Input file: standard input
Output file: standard output
Time limit: 3 seconds
Memory limit: 256 megabytes

Being a physicist, Charlie likes to plan his life in simple and precise terms.

For the next m months, starting with no money, Charlie will work hard and earn x pounds per month. For the i-th month $(1 \leq i \leq m)$, there'll be a single opportunity of paying cost c_i pounds to obtain happiness h_i.

Borrowing is not allowed. Money earned in the i-th month can only be spent in a later j-th month ($j > i$).

Since physicists don't code, help Charlie find the maximum obtainable sum of happiness.

Input

The first line of input contains a single integer t ($1 \leq t \leq 1000$) — the number of test cases.

The first line of each test case contains two integers, m and x ($1 \leq m \leq 50$, $1 \leq x \leq 10^8$) — the total number of months and the monthly salary.

The i-th of the following m lines contains two integers, c_i and h_i ($0 \leq c_i \leq 10^8$, $1 \leq h_i \leq 10^3$) — the cost and happiness on offer for the i-th month. Note that some happiness may be free ($c_i = 0$ for some i's).

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

Output

For each test case, print a single integer, the maximum sum of happiness Charlie could obtain.

Standard Input	Standard Output
7 1 10 1 5 2 80 0 10 200 100 3 100 70 100 100 200 150 150 5 8 3 1 5 3 3 4 1 5 5 3 2 5 1 5 2 1 5 3 2 5	0 10 200 15 1 9 9

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

Note

In the first test case, Charlie only gets paid at the end of the month, so is unable to afford anything.

In the second test case, Charlie obtains the free happiness in the first month.

In the third test case, it's optimal for Charlie to buy happiness in the second month. Even with money left at the end, Charlie could not go back in time to obtain the happiness on offer in the first month.

,

F. Cutting Game

Input file: standard input
Output file: standard output
Time limit: 3 seconds
Memory limit: 256 megabytes

Alice and Bob were playing a game again. They have a grid of size $a \times b$ ($1 \leq a, b \leq 10^9$), on which there are n chips, with at most one chip in each cell. The cell at the intersection of the x-th row and the y-th column has coordinates (x, y).

Alice made the first move, and the players took turns. On each move, a player could cut several (but not all) rows or columns from the beginning or end of the remaining grid and earn a point for each chip that was on the cut part of the grid. Each move can be described by the character 'U', 'D', 'L', or 'R' and an integer k:

- If the character is 'U', then the first k remaining rows will be cut;
- If the character is 'D', then the last k remaining rows will be cut;
- If the character is 'L', then the first k remaining columns will be cut;
- If the character is 'R', then the last k remaining columns will be cut.

Based on the initial state of the grid and the players' moves, determine the number of points earned by Alice and Bob, respectively.

Input

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

The first line of each test case contains four integers a, b, n, and m ($2 \leq a, b \leq 10^9$, $1 \leq n, m \leq 2 \cdot 10^5$) — the dimensions of the grid, the number of chips, and the number of moves.

Each of the next n lines contain two integers x_i and y_i ($1 \leq x_i \leq a$, $1 \leq y_i \leq b$) — the coordinates of the chips. All pairs of coordinates are distinct.

Each of the next m lines contain a character c_j and an integer k_j — the description of the j-th move. It is guaranteed that k is **less than** the number of rows/columns in the current grid. In other words, a player cannot cut the entire remaining grid on their move.

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

Output

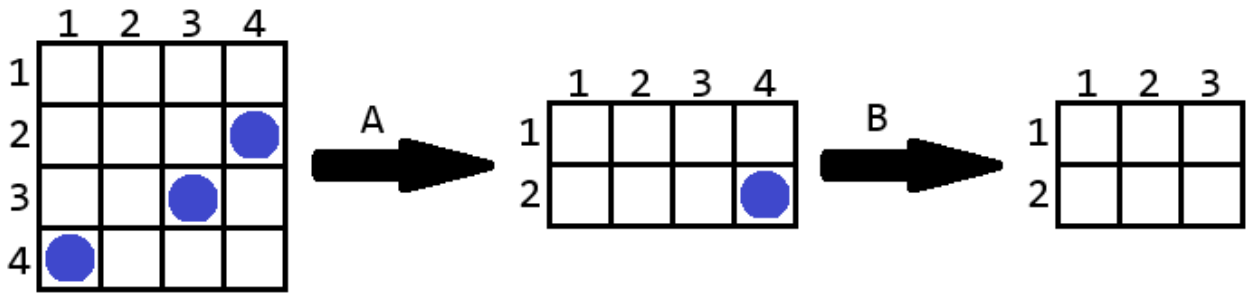
For each test case, output two integers — the number of points earned by Alice and Bob, respectively.

Standard Input	Standard Output
6 4 4 3 2 4 1 3 3 2 4 D 2 R 1 4 4 3 3 4 1	2 1 2 0 0 3 1 1 2 0 0 1

```
3 2
2 3
D 1
L 1
U 2
3 5 3 2
1 3
2 2
3 3
R 2
R 2
6 4 4 2
1 4
2 3
5 3
1 1
R 1
U 1
9 3 2 1
6 1
3 3
D 8
10 10 2 5
7 5
9 1
R 1
L 2
D 1
U 4
D 1
```

Note

Below is the game from the first example:



On her turn, Alice cut 2 rows from the bottom and scored 2 points, then Bob cut 1 column from the right and scored one point. Note that if Bob had cut 1 row from the bottom, he would have also scored 1 point.

,

G. Money Buys Less Happiness Now

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

You can never buy enough happiness, so here we go again! In this version, you can only buy $h_i = 1$ unit of happiness each month, but the number of months is hugely increased. We are in the realm of quantum happiness and time dilation.

Being a physicist, Charlie likes to plan his life in simple and precise terms.

For the next m months, starting with no money, Charlie will work hard and earn x pounds per month. For the i-th month $(1 \leq i \leq m)$, there'll be a single opportunity of paying cost c_i pounds to obtain one unit of happiness. You cannot buy more than one unit each month.

Borrowing is not allowed. Money earned in the i-th month can only be spent in a later j-th month ($j > i$).

Since physicists don't code, help Charlie find the maximum reachable units of happiness.

Input

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

The first line of each test case contains two integers, m and x ($1 \leq m \leq 2 \cdot 10^5$, $1 \leq x \leq 10^3$) — the total number of months and the monthly salary.

The second line of each test case contains m integers $c_1, c_2, \ldots, c_m$ ($1 \leq c_i \leq 10^3$) — the cost of one unit of happiness for each month.

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

Output

For each test case, output one integer — the maximal amount of happiness Charlie can get.

Standard Input	Standard Output
6 3 3 2 2 2 6 5 2 2 8 2 6 8 6 4 4 10 3 8 6 10 2 1 1 1 4 1 4 1 3 1 4 2 1 3 4 3	2 4 3 1 2 1