Educational Codeforces Round 112 (Rated for Div. 2)

A. PizzaForces

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

PizzaForces is Petya's favorite pizzeria. PizzaForces makes and sells pizzas of three sizes: small pizzas consist of 6 slices, medium ones consist of 8 slices, and large pizzas consist of 10 slices each. Baking them takes 15, 20 and 25 minutes, respectively.

Petya's birthday is today, and n of his friends will come, so he decided to make an order from his favorite pizzeria. Petya wants to order so much pizza that each of his friends gets at least one slice of pizza. The cooking time of the order is the total baking time of all the pizzas in the order.

Your task is to determine the minimum number of minutes that is needed to make pizzas containing at least n slices in total. For example:

- if 12 friends come to Petya's birthday, he has to order pizzas containing at least 12 slices in total. He can order two small pizzas, containing exactly 12 slices, and the time to bake them is 30 minutes;
- if 15 friends come to Petya's birthday, he has to order pizzas containing at least 15 slices in total. He can order a small pizza and a large pizza, containing 16 slices, and the time to bake them is 40 minutes;
- if 300 friends come to Petya's birthday, he has to order pizzas containing at least 300 slices in total. He can order 15 small pizzas, 10 medium pizzas and 13 large pizzas, in total they contain $15 \cdot 6 + 10 \cdot 8 + 13 \cdot 10 = 300$ slices, and the total time to bake them is $15 \cdot 15 + 10 \cdot 20 + 13 \cdot 25 = 750$ minutes;
- if only one friend comes to Petya's birthday, he can order a small pizza, and the time to bake it is 15 minutes.

Input

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

Each testcase consists of a single line that contains a single integer n ($1 \le n \le 10^{16}$) — the number of Petya's friends.

Output

For each testcase, print one integer — the minimum number of minutes that is needed to bake pizzas containing at least n slices in total.

Example

input
6 12 15 300 1 9999999999999999 3

output
30 40 750 15 25000000000000000 15

B. Two Tables

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

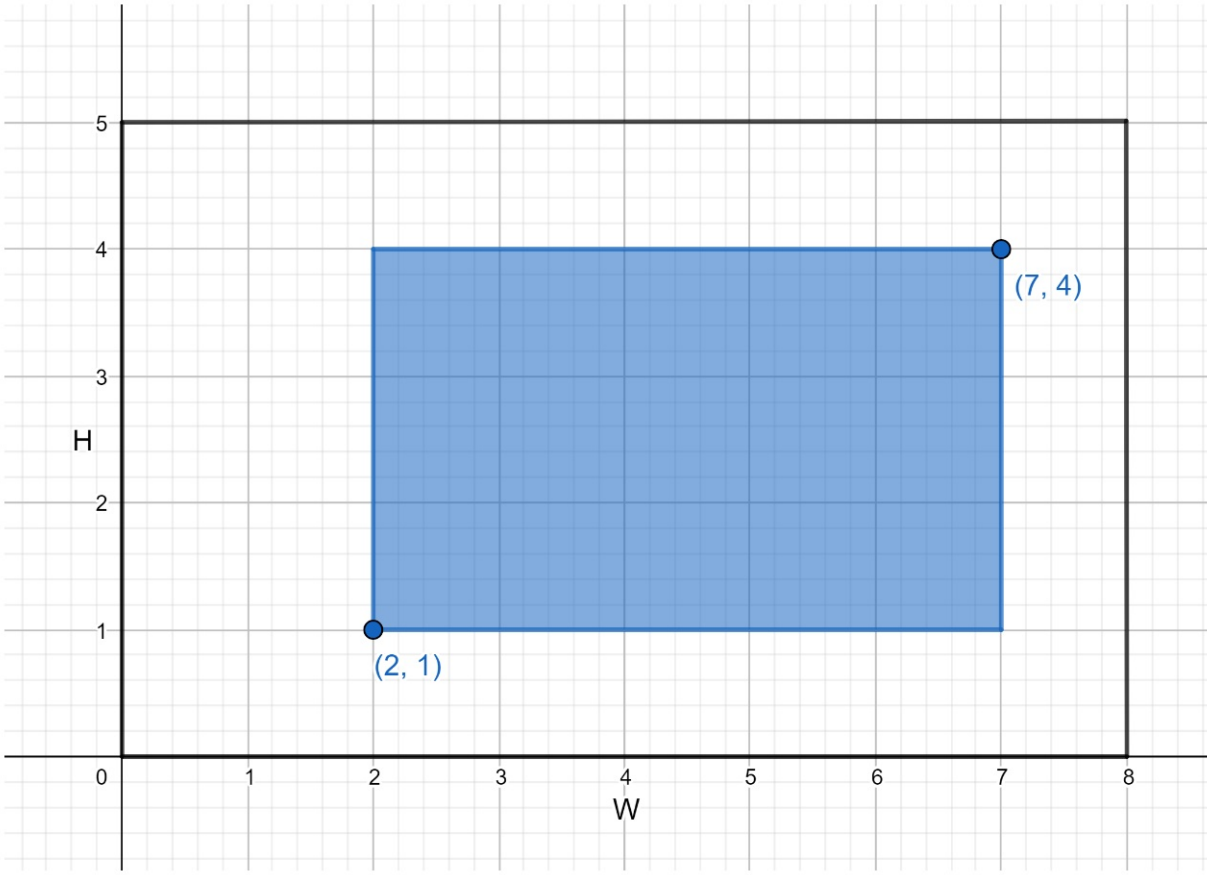
You have an axis-aligned rectangle room with width W and height H, so the lower left corner is in point $(0, 0)$ and the upper right corner is in (W, H).

There is a rectangular table standing in this room. The sides of the table are parallel to the walls, the lower left corner is in (x_1, y_1), and the upper right corner in (x_2, y_2).

You want to place another rectangular table in this room with width w and height h with the width of the table parallel to the width of the room.

The problem is that sometimes there is not enough space to place the second table without intersecting with the first one (there are no problems with tables touching, though).

You **can't rotate** any of the tables, but you can move the first table inside the room.



Example of how you may move the first table.

What is the minimum distance you should move the first table to free enough space for the second one?

Input

The first line contains the single integer t ($1 \le t \le 5000$) — the number of the test cases.

The first line of each test case contains two integers W and H ($1 \le W, H \le 10^8$) — the width and the height of the room.

The second line contains four integers x_1, y_1, x_2 and y_2 ($0 \le x_1 < x_2 \le W$; $0 \le y_1 < y_2 \le H$) — the coordinates of the corners of the first table.

The third line contains two integers w and h ($1 \le w \le W$; $1 \le h \le H$) — the width and the height of the second table.

Output

For each test case, print the minimum distance you should move the first table, or -1 if there is no way to free enough space for the second table.

Your answer will be considered correct if its absolute or relative error doesn't exceed 10^{-6}.

Example

input
5 8 5 2 1 7 4 4 2 5 4 2 2 5 4 3 3 1 8 0 3 1 6 1 5 8 1 3 0 6 1 5 1 8 10 4 5 7 8 8 5

output
1.000000000 -1 2.000000000 2.000000000 0.000000000

Note

The configuration of the first test case is shown in the picture. But the movement of the first table is not optimal. One of the optimal movement, for example, is to move the table by $(0, -1)$, so the lower left corner will move from $(2, 1)$ to $(2, 0)$. Then you can place the second table at $(0, 3) - (4, 5)$.

In the second test case, there is no way to fit both tables in the room without intersecting.

In the third test case, you can move the first table by $(0, 2)$, so the lower left corner will move from $(0, 3)$ to $(0, 5)$.

C. Coin Rows

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Alice and Bob are playing a game on a matrix, consisting of 2 rows and m columns. The cell in the i-th row in the j-th column contains $a_{i,j}$ coins in it.

Initially, both Alice and Bob are standing in a cell $(1, 1)$. They are going to perform a sequence of moves to reach a cell $(2, m)$.

The possible moves are:

- Move right — from some cell (x, y) to $(x, y + 1)$;
- Move down — from some cell (x, y) to $(x + 1, y)$.

First, Alice makes **all her moves** until she reaches $(2, m)$. She collects the coins in all cells she visit (including the starting cell).

When Alice finishes, Bob starts his journey. He also performs the moves to reach $(2, m)$ and collects the coins in all cells that he visited, **but Alice didn't**.

The score of the game is the total number of coins Bob collects.

Alice wants to minimize the score. Bob wants to maximize the score. What will the score of the game be if both players play optimally?

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

Then the descriptions of t testcases follow.

The first line of the testcase contains a single integer m ($1 \le m \le 10^5$) — the number of columns of the matrix.

The i-th of the next 2 lines contain m integers $a_{i,1}, a_{i,2}, \ldots, a_{i,m}$ ($1 \le a_{i,j} \le 10^4$) — the number of coins in the cell in the i-th row in the j-th column of the matrix.

The sum of m over all testcases doesn't exceed 10^5.

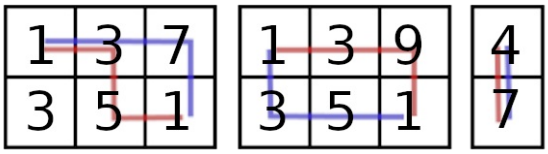
Output
For each testcase print a single integer — the score of the game if both players play optimally.

Example

input
3 3 1 3 7 3 5 1 3 1 3 9 3 5 1 1 4 7

output
7 8 0

Note
The paths for the testcases are shown on the following pictures. Alice's path is depicted in red and Bob's path is depicted in blue.



D. Say No to Palindromes

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Let's call the string **beautiful** if it does not contain a substring of length at least 2, which is a palindrome. Recall that a palindrome is a string that reads the same way from the first character to the last and from the last character to the first. For example, the strings a, bab, acca, bcabcbacb are palindromes, but the strings ab, abbbaa, cccb are not.

Let's define **cost** of a string as the minimum number of operations so that the string becomes beautiful, if in one operation it is allowed to change any character of the string to one of the first 3 letters of the Latin alphabet (in lowercase).

You are given a string s of length n, each character of the string is one of the first 3 letters of the Latin alphabet (in lowercase).

You have to answer m queries — calculate the cost of the substring of the string s from l_i-th to r_i-th position, inclusive.

Input
The first line contains two integers n and m ($1 \le n, m \le 2 \cdot 10^5$) — the length of the string s and the number of queries.

The second line contains the string s, it consists of n characters, each character one of the first 3 Latin letters.

The following m lines contain two integers l_i and r_i ($1 \le l_i \le r_i \le n$) — parameters of the i-th query.

Output
For each query, print a single integer — the cost of the substring of the string s from l_i-th to r_i-th position, inclusive.

Example

input
5 4 baacb 1 3 1 5 4 5 2 3

output
1 2 0 1

Note
Consider the queries of the example test.

- in the first query, the substring is baa, which can be changed to bac in one operation;
- in the second query, the substring is baacb, which can be changed to cbacb in two operations;
- in the third query, the substring is cb, which can be left unchanged;
- in the fourth query, the substring is aa, which can be changed to ba in one operation.

E. Boring Segments

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given n segments on a number line, numbered from 1 to n. The i-th segments covers all integer points from l_i to r_i and has a value w_i.

You are asked to select a subset of these segments (possibly, all of them). Once the subset is selected, it's possible to travel between two integer points if there exists a selected segment that covers both of them. A subset is good if it's possible to reach point m starting from point 1 in arbitrary number of moves.

The cost of the subset is the difference between the maximum and the minimum values of segments in it. Find the minimum cost of a good subset.

In every test there exists at least one good subset.

Input
The first line contains two integers n and m ($1 \le n \le 3 \cdot 10^5$; $2 \le m \le 10^6$) — the number of segments and the number of integer points.

Each of the next n lines contains three integers l_i, r_i and w_i ($1 \le l_i < r_i \le m$; $1 \le w_i \le 10^6$) — the description of the i-th

segment.

In every test there exists at least one good subset.

Output
Print a single integer — the minimum cost of a good subset.

Examples

input
5 12 1 5 5 3 4 10 4 10 6 11 12 5 10 12 3
output
3

input
1 10 1 10 23
output
0

F. Good Graph

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have an undirected graph consisting of n vertices with weighted edges.

A simple cycle is a cycle of the graph without repeated vertices. Let the *weight* of the cycle be the XOR of weights of edges it consists of.

Let's say the graph is *good* if all its *simple* cycles have weight 1. A graph is bad if it's not good.

Initially, the graph is empty. Then q queries follow. Each query has the next type:

- u v x — add edge between vertices u and v of weight x if it doesn't make the graph bad.

For each query print, was the edge added or not.

Input
The first line contains two integers n and q ($3 \le n \le 3 \cdot 10^5$; $1 \le q \le 5 \cdot 10^5$) — the number of vertices and queries.

Next q lines contain queries — one per line. Each query contains three integers u, v and x ($1 \le u, v \le n$; $u \neq v$; $0 \le x \le 1$) — the vertices of the edge and its weight.

It's guaranteed that there are no multiple edges in the input.

Output
For each query, print YES if the edge was added to the graph, or NO otherwise (both case-insensitive).

Example

input
9 12 6 1 0 1 3 1 3 6 0 6 2 0 6 4 1 3 4 1 2 4 0 2 5 0 4 5 0 7 8 1 8 9 1 9 7 0
output
YES YES YES YES YES NO YES YES NO YES YES NO