Educational Codeforces Round 102 (Rated for Div. 2)

A. Replacing Elements

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

You have an array $a_1, a_2, \ldots, a_n$. All a_i are positive integers.

In one step you can choose three distinct indices i, j, and k ($i \neq j$; $i \neq k$; $j \neq k$) and assign the sum of a_j and a_k to a_i, i. e. make $a_i = a_j + a_k$.

Can you make all a_i lower or equal to d using the operation above any number of times (possibly, zero)?

Input

The first line contains a single integer t ($1 \leq t \leq 2000$) — the number of test cases.

The first line of each test case contains two integers n and d ($3 \leq n \leq 100$; $1 \leq d \leq 100$) — the number of elements in the array a and the value d.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 100$) — the array a.

Output

For each test case, print YES, if it's possible to make all elements a_i less or equal than d using the operation above. Otherwise, print NO.

You may print each letter in any case (for example, YES, Yes, yes, yEs will all be recognized as positive answer).

Example

input
3 5 3 2 3 2 5 4 3 4 2 4 4 5 4 2 1 5 3 6
output
NO YES YES

Note

In the first test case, we can prove that we can't make all $a_i \leq 3$.

In the second test case, all a_i are already less or equal than $d = 4$.

In the third test case, we can, for example, choose $i = 5$, $j = 1$, $k = 2$ and make $a_5 = a_1 + a_2 = 2 + 1 = 3$. Array a will become $[2, 1, 5, 3, 3]$.

After that we can make $a_3 = a_5 + a_2 = 3 + 1 = 4$. Array will become $[2, 1, 4, 3, 3]$ and all elements are less or equal than $d = 4$.

B. String LCM

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

Let's define a multiplication operation between a string a and a positive integer x: $a \cdot x$ is the string that is a result of writing x copies of a one after another. For example, "abc" $\cdot$ 2 = "abcabc", "a" $\cdot$ 5 = "aaaaa".

A string a is divisible by another string b if there exists an integer x such that $b \cdot x = a$. For example, "abababab" is divisible by "ab", but is not divisible by "ababab" or "aa".

LCM of two strings s and t (defined as $LCM(s, t)$) is the shortest non-empty string that is divisible by both s and t.

You are given two strings s and t. Find $LCM(s, t)$ or report that it does not exist. It can be shown that if $LCM(s, t)$ exists, it is unique.

Input
The first line contains one integer q ($1 \le q \le 2000$) — the number of test cases.

Each test case consists of two lines, containing strings s and t ($1 \le |s|, |t| \le 20$). Each character in each of these strings is either 'a' or 'b'.

Output
For each test case, print $LCM(s, t)$ if it exists; otherwise, print -1. It can be shown that if $LCM(s, t)$ exists, it is unique.

Example

input
3 baba ba aa aaa aba ab
output
baba aaaaaa -1

Note
In the first test case, "baba" = "baba" $\cdot$ 1 = "ba" $\cdot$ 2.

In the second test case, "aaaaaa" = "aa" $\cdot$ 3 = "aaa" $\cdot$ 2.

C. No More Inversions

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

You have a sequence a with n elements $1, 2, 3, \ldots, k-1, k, k-1, k-2, \ldots, k-(n-k)$ ($k \le n < 2k$).

Let's call as inversion in a a pair of indices $i < j$ such that $a[i] > a[j]$.

Suppose, you have some permutation p of size k and you build a sequence b of size n in the following manner: $b[i] = p[a[i]]$.

Your goal is to find such permutation p that the total number of inversions in b doesn't exceed the total number of inversions in a, and b is *lexicographically maximum*.

Small reminder: the sequence of k integers is called a permutation if it contains all integers from 1 to k exactly once.

Another small reminder: a sequence s is *lexicographically smaller* than another sequence t, if either s is a prefix of t, or for the first i such that $s_i \neq t_i$, $s_i < t_i$ holds (in the first position that these sequences are different, s has smaller number than t).

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

The first and only line of each test case contains two integers n and k ($k \le n < 2k$; $1 \le k \le 10^5$) — the length of the sequence a and its maximum.

It's guaranteed that the total sum of k over test cases doesn't exceed 10^5.

Output
For each test case, print k integers — the permutation p which maximizes b lexicographically without increasing the total number of inversions.

It can be proven that p exists and is unique.

Example

input
4 1 1 2 2 3 2 4 3
output
1 1 2 2 1 1 3 2

Note

In the first test case, the sequence $a = [1]$, there is only one permutation $p = [1]$.

In the second test case, the sequence $a = [1, 2]$. There is no inversion in a, so there is only one permutation $p = [1, 2]$ which doesn't increase the number of inversions.

In the third test case, $a = [1, 2, 1]$ and has 1 inversion. If we use $p = [2, 1]$, then $b = [p[a[1]], p[a[2]], p[a[3]]] = [2, 1, 2]$ and also has 1 inversion.

In the fourth test case, $a = [1, 2, 3, 2]$, and since $p = [1, 3, 2]$ then $b = [1, 3, 2, 3]$. Both a and b have 1 inversion and b is the lexicographically maximum.

D. Program

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

You are given a program that consists of n instructions. Initially a single variable x is assigned to 0. Afterwards, the instructions are of two types:

- increase x by 1;
- decrease x by 1.

You are given m queries of the following format:

- query l r — how many distinct values is x assigned to if all the instructions between the l-th one and the r-th one inclusive are ignored and the rest are executed without changing the order?

Input

The first line contains a single integer t ($1 \le t \le 1000$) — the number of testcases.

Then the description of t testcases follows.

The first line of each testcase contains two integers n and m ($1 \le n, m \le 2 \cdot 10^5$) — the number of instructions in the program and the number of queries.

The second line of each testcase contains a program — a string of n characters: each character is either '+' or '-' — increment and decrement instruction, respectively.

Each of the next m lines contains two integers l and r ($1 \le l \le r \le n$) — the description of the query.

The sum of n over all testcases doesn't exceed $2 \cdot 10^5$. The sum of m over all testcases doesn't exceed $2 \cdot 10^5$.

Output

For each testcase print m integers — for each query l, r print the number of distinct values variable x is assigned to if all the instructions between the l-th one and the r-th one inclusive are ignored and the rest are executed without changing the order.

Example

input
2 8 4 -+--+--+ 1 8 2 8 2 5 1 1 4 10 +-++ 1 1 1 2 2 2 1 3 2 3 3 3 1 4 2 4 3 4 4 4
output
1 2 4 4 3 3 4 2 3 2 1

```
2
2
2
```

Note

The instructions that remain for each query of the first testcase are:

1. empty program — x was only equal to 0;
2. "-" — x had values 0 and -1;
3. "---+" — x had values 0, -1, -2, -3, -2 — there are 4 distinct values among them;
4. "+--+--+" — the distinct values are 1, 0, -1, -2.

E. Minimum Path

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

You are given a weighted undirected connected graph consisting of n vertices and m edges. It is guaranteed that there are no self-loops or multiple edges in the given graph.

Let's define the weight of the path consisting of k edges with indices $e_1, e_2, \ldots, e_k$ as $\sum\limits_{i=1}^{k} w_{e_i} - \max\limits_{i=1}^{k} w_{e_i} + \min\limits_{i=1}^{k} w_{e_i}$, where w_i — weight of the i-th edge in the graph.

Your task is to find the minimum weight of the path from the 1-st vertex to the i-th vertex for each i ($2 \le i \le n$).

Input

The first line contains two integers n and m ($2 \le n \le 2 \cdot 10^5$; $1 \le m \le 2 \cdot 10^5$) — the number of vertices and the number of edges in the graph.

Following m lines contains three integers v_i, u_i, w_i ($1 \le v_i, u_i \le n$; $1 \le w_i \le 10^9$; $v_i \ne u_i$) — endpoints of the i-th edge and its weight respectively.

Output

Print $n - 1$ integers — the minimum weight of the path from 1-st vertex to the i-th vertex for each i ($2 \le i \le n$).

Examples

input
5 4 5 3 4 2 1 1 3 2 2 2 4 2
output
1 2 2 4

input
6 8 3 1 1 3 6 2 5 4 2 4 2 2 6 1 1 5 2 1 3 2 3 1 5 4
output
2 1 4 3 1

input
7 10 7 5 5 2 3 3 4 7 1 5 3 6 2 7 6 6 2 6 3 7 6 4 2 1 3 1 4 1 7 4
output
3 4 2 7 7 3

F. Strange Set

time limit per test: 4 seconds
memory limit per test: 32 megabytes
input: standard input
output: standard output

Note that the memory limit is unusual.

You are given an integer n and two sequences $a_1, a_2, \ldots, a_n$ and $b_1, b_2, \ldots, b_n$.

Let's call a set of integers S such that $S \subseteq \{1, 2, 3, \ldots, n\}$ *strange*, if, for every element i of S, the following condition is met: for every $j \in [1, i-1]$, if a_j divides a_i, then j is also included in S. An empty set is always *strange*.

The *cost* of the set S is $\sum\limits_{i \in S} b_i$. You have to calculate the maximum possible *cost* of a *strange* set.

Input

The first line contains one integer n ($1 \le n \le 3000$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 100$).

The third line contains n integers $b_1, b_2, \ldots, b_n$ ($-10^5 \le b_i \le 10^5$).

Output

Print one integer — the maximum *cost* of a *strange* set.

Examples

input
9 4 7 3 4 5 6 7 8 13 -2 3 -19 5 -6 7 -8 9 1
output
16

input
2 42 42 -37 13
output
0

input
2 42 42 13 -37
output
13

Note

The *strange* set with the maximum *cost* in the first example is $\{1, 2, 4, 8, 9\}$.

The *strange* set with the maximum *cost* in the second example is empty.

G. Tiles

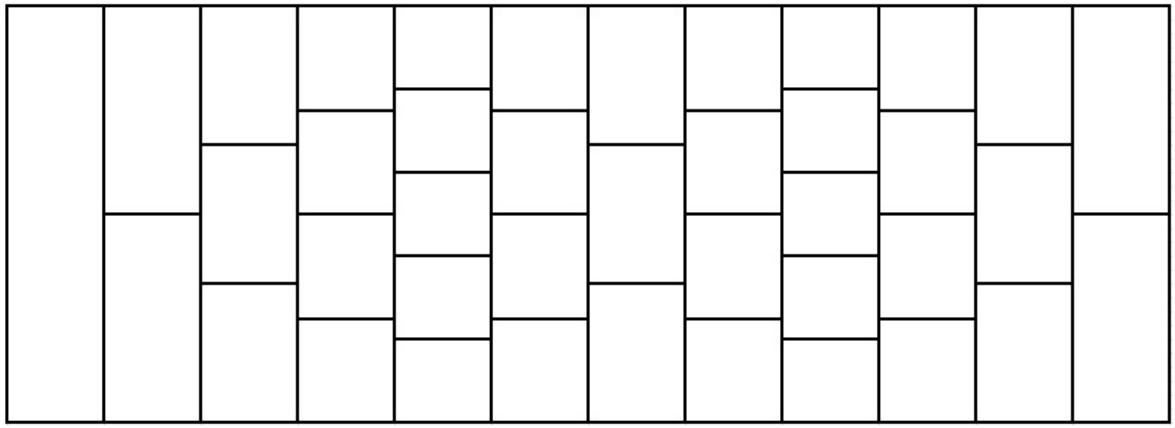
time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Consider a road consisting of several rows. Each row is divided into several rectangular tiles, and all tiles in the same row are equal. The first row contains exactly one rectangular tile. Look at the picture below which shows how the tiles are arranged.

The road is constructed as follows:

- the first row consists of 1 tile;
- then a_1 rows follow; each of these rows contains 1 tile greater than the previous row;
- then b_1 rows follow; each of these rows contains 1 tile less than the previous row;
- then a_2 rows follow; each of these rows contains 1 tile greater than the previous row;
- then b_2 rows follow; each of these rows contains 1 tile less than the previous row;
- ...
- then a_n rows follow; each of these rows contains 1 tile greater than the previous row;

- then b_n rows follow; each of these rows contains 1 tile less than the previous row.



An example of the road with $n = 2$, $a_1 = 4$, $b_1 = 2$, $a_2 = 2$, $b_2 = 3$. Rows are arranged from left to right.

You start from the only tile in the first row and want to reach the last row (any tile of it). From your current tile, you can move to any tile in the next row which touches your current tile.

Calculate the number of different paths from the first row to the last row. Since it can be large, print it modulo 998244353.

Input

The first line contains one integer n ($1 \le n \le 1000$).

Then n lines follow. The i-th of them contains two integers a_i and b_i ($1 \le a_i, b_i \le 10^5$; $|a_i - b_i| \le 5$).

Additional constraint on the input: the sequence of a_i and b_i never results in a row with non-positive number of tiles.

Output

Print one integer — the number of paths from the first row to the last row, taken modulo 998244353.

Examples

input
2 4 2 2 3
output
850

input
3 4 1 2 3 3 1
output
10150

input
8 328 323 867 868 715 718 721 722 439 435 868 870 834 834 797 796
output
759099319