Educational Codeforces Round 124 (Rated for Div. 2)

A. Playoff

time limit per test: 2 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Consider a playoff tournament where 2^n athletes compete. The athletes are numbered from 1 to 2^n.

The tournament is held in n stages. In each stage, the athletes are split into pairs in such a way that each athlete belongs exactly to one pair. In each pair, the athletes compete against each other, and exactly one of them wins. The winner of each pair advances to the next stage, the athlete who was defeated gets eliminated from the tournament.

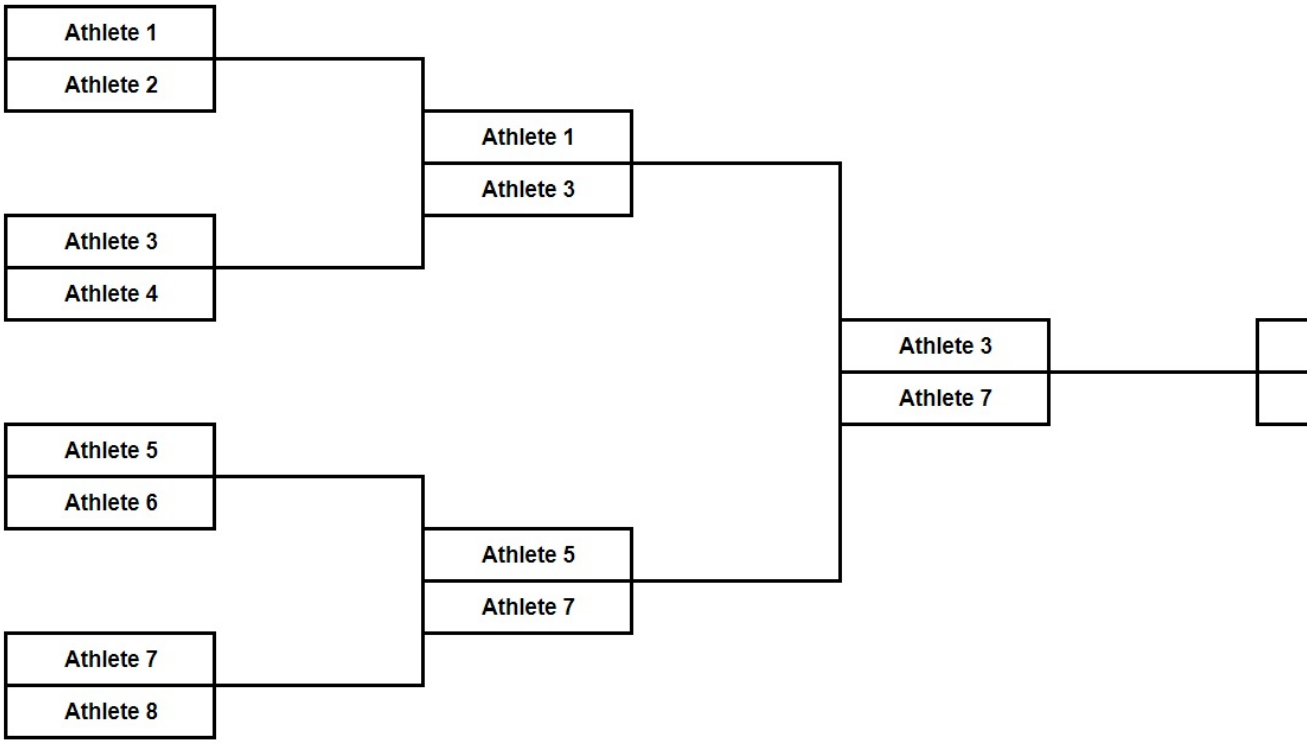
The pairs are formed as follows:

- in the first stage, athlete 1 competes against athlete 2; 3 competes against 4; 5 competes against 6, and so on;
- in the second stage, the winner of the match "1–2" competes against the winner of the match "3–4"; the winner of the match "5–6" competes against the winner of the match "7–8", and so on;
- the next stages are held according to the same rules.

When athletes x and y compete, the winner is decided as follows:

- if $x + y$ is odd, the athlete with the lower index wins (i. e. if $x < y$, then x wins, otherwise y wins);
- if $x + y$ is even, the athlete with the higher index wins.

The following picture describes the way the tournament with $n = 3$ goes.



Your task is the following one: given the integer n, determine the index of the athlete who wins the tournament.

Input

The first line contains one integer t ($1 \le t \le 30$) — the number of test cases.

Each test case consists of one line containing one integer n ($1 \le n \le 30$).

Output

For each test case, print one integer — the index of the winner of the tournament.

Example

input
2 3 1
output
7 1

Note

The case $n = 3$ is shown in the picture from the statement.

If $n = 1$, then there's only one match between athletes 1 and 2. Since $1 + 2 = 3$ is an odd number, the athlete with the lower index wins. So, the athlete 1 is the winner.

B. Prove Him Wrong

time limit per test: 2 seconds
memory limit per test: 256 megabytes

input: standard input
output: standard output

Recently, your friend discovered one special operation on an integer array a:

1. Choose two indices i and j ($i \neq j$);
2. Set $a_i = a_j = |a_i - a_j|$.

After playing with this operation for a while, he came to the next conclusion:

- For every array a of n integers, where $1 \leq a_i \leq 10^9$, you can find a pair of indices (i, j) such that the total sum of a will **decrease** after performing the operation.

This statement sounds fishy to you, so you want to find a counterexample for a given integer n. Can you find such counterexample and prove him wrong?

In other words, find an array a consisting of n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^9$) such that for all pairs of indices (i, j) performing the operation won't decrease the total sum (it will increase or not change the sum).

Input
The first line contains a single integer t ($1 \leq t \leq 100$) — the number of test cases. Then t test cases follow.

The first and only line of each test case contains a single integer n ($2 \leq n \leq 1000$) — the length of array a.

Output
For each test case, if there is no counterexample array a of size n, print `NO`.

Otherwise, print YES followed by the array a itself ($1 \leq a_i \leq 10^9$). If there are multiple counterexamples, print any.

Example

input
3 2 512 3

output
YES 1 337 NO YES 31 4 159

Note
In the first test case, the only possible pairs of indices are $(1, 2)$ and $(2, 1)$.

If you perform the operation on indices $(1, 2)$ (or $(2, 1)$), you'll get $a_1 = a_2 = |1 - 337| = 336$, or array $[336, 336]$. In both cases, the total sum increases, so this array a is a counterexample.

C. Fault-tolerant Network

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

There is a classroom with two rows of computers. There are n computers in each row and each computer has its own grade. Computers in the first row has grades $a_1, a_2, \ldots, a_n$ and in the second row — $b_1, b_2, \ldots, b_n$.

Initially, all pairs of **neighboring** computers in each row are connected by wire (pairs $(i, i + 1)$ for all $1 \leq i < n$), so two rows form two independent computer networks.

Your task is to combine them in one common network by connecting one or more pairs of computers from **different** rows. Connecting the i-th computer from the first row and the j-th computer from the second row costs $|a_i - b_j|$.

You can connect one computer to several other computers, but you need to provide at least a basic fault tolerance: you need to connect computers in such a way that the network stays connected, despite one of its computer failing. In other words, if one computer is broken (no matter which one), the network won't split in two or more parts.

That is the minimum total cost to make a fault-tolerant network?

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

The first line of each test case contains the single integer n ($3 \leq n \leq 2 \cdot 10^5$) — the number of computers in each row.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^9$) — the grades of computers in the first row.

The third line contains n integers $b_1, b_2, \ldots, b_n$ ($1 \leq b_i \leq 10^9$) — the grades of computers in the second row.

It's guaranteed that the total sum of n doesn't exceed $2 \cdot 10^5$.

Output
For each test case, print a single integer — the minimum total cost to make a fault-tolerant network.

Example

input
2 3 1 10 1 20 4 25 4 1 1 1 1 1000000000 1000000000 1000000000 1000000000

output
31 1999999998

Note
In the first test case, it's optimal to connect four pairs of computers:

1. computer 1 from the first row with computer 2 from the second row: cost $|1-4|=3$;
2. computer 3 from the first row with computer 2 from the second row: cost $|1-4|=3$;
3. computer 2 from the first row with computer 1 from the second row: cost $|10-20|=10$;
4. computer 2 from the first row with computer 3 from the second row: cost $|10-25|=15$;

In total, $3+3+10+15=31$.
In the second test case, it's optimal to connect 1 from the first row with 1 from the second row, and 4 from the first row with 4 from the second row.

D. Nearest Excluded Points

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

You are given n distinct points on a plane. The coordinates of the i-th point are (x_i, y_i).

For each point i, find the nearest (in terms of Manhattan distance) point with **integer coordinates** that is not among the given n points. If there are multiple such points — you can choose any of them.

The Manhattan distance between two points (x_1, y_1) and (x_2, y_2) is $|x_1 - x_2| + |y_1 - y_2|$.

Input
The first line of the input contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of points in the set.

The next n lines describe points. The i-th of them contains two integers x_i and y_i ($1 \le x_i, y_i \le 2 \cdot 10^5$) — coordinates of the i-th point.

It is guaranteed that all points in the input are distinct.

Output
Print n lines. In the i-th line, print the point with **integer coordinates** that is not among the given n points and is the nearest (in terms of Manhattan distance) to the i-th point from the input.

Output coordinates should be in range $[-10^6; 10^6]$. It can be shown that any optimal answer meets these constraints.

If there are several answers, you can print any of them.

Examples

input
6 2 2 1 2 2 1 3 2 2 3 5 5
output
1 1 1 1 2 0 3 1 2 4 5 4

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

E. Sum of Matchings

time limit per test: 4 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Let's denote the size of the maximum matching in a graph G as $MM(G)$.

You are given a bipartite graph. The vertices of the first part are numbered from 1 to n, the vertices of the second part are numbered from $n+1$ to $2n$. **Each vertex's degree is** 2.

For a tuple of four integers (l, r, L, R), where $1 \le l \le r \le n$ and $n+1 \le L \le R \le 2n$, let's define $G'(l, r, L, R)$ as the graph which consists of all vertices of the given graph that are included in the segment $[l, r]$ or in the segment $[L, R]$, and all edges of the given graph such that each of their endpoints belongs to one of these segments. In other words, to obtain $G'(l, r, L, R)$ from the original graph, you have to remove all vertices i such that $i \notin [l, r]$ and $i \notin [L, R]$, and all edges incident to these vertices.

Calculate the sum of $MM(G(l, r, L, R))$ over all tuples of integers (l, r, L, R) having $1 \le l \le r \le n$ and $n+1 \le L \le R \le 2n$.

Input
The first line contains one integer n ($2 \le n \le 1500$) — the number of vertices in each part.

Then $2n$ lines follow, each denoting an edge of the graph. The i-th line contains two integers x_i and y_i ($1 \le x_i \le n$; $n+1 \le y_i \le 2n$) — the endpoints of the i-th edge.

There are no multiple edges in the given graph, and each vertex has **exactly** two incident edges.

Output
Print one integer — the sum of $MM(G(l, r, L, R))$ over all tuples of integers (l, r, L, R) having $1 \le l \le r \le n$ and $n + 1 \le L \le R \le 2n$.

Example

input
5 4 6 4 9 2 6 3 9 1 8 5 10 2 7 3 7 1 10 5 8
output
314

F. Tower Defense

time limit per test: 4 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

Monocarp is playing a tower defense game. A level in the game can be represented as an OX axis, where each lattice point from 1 to n contains a tower in it.

The tower in the i-th point has c_i mana capacity and r_i mana regeneration rate. In the beginning, before the 0-th second, each tower has full mana. If, at the end of some second, the i-th tower has x mana, then it becomes $min(x + r_i, c_i)$ mana for the next second.

There are q monsters spawning on a level. The j-th monster spawns at point 1 at the beginning of t_j-th second, and it has h_j health. Every monster is moving 1 point per second in the direction of increasing coordinate.

When a monster passes the tower, the tower deals $min(H, M)$ damage to it, where H is the current health of the monster and M is the current mana amount of the tower. This amount gets subtracted from both monster's health and tower's mana.

Unfortunately, sometimes some monsters can pass all n towers and remain alive. Monocarp wants to know what will be the total health of the monsters after they pass all towers.

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

The i-th of the next n lines contains two integers c_i and r_i ($1 \le r_i \le c_i \le 10^9$) — the mana capacity and the mana regeneration rate of the i-th tower.

The next line contains a single integer q ($1 \le q \le 2 \cdot 10^5$) — the number of monsters.

The j-th of the next q lines contains two integers t_j and h_j ($0 \le t_j \le 2 \cdot 10^5$; $1 \le h_j \le 10^{12}$) — the time the j-th monster spawns and its health.

The monsters are listed in the increasing order of their spawn time, so $t_j < t_{j+1}$ for all $1 \le j \le q - 1$.

Output
Print a single integer — the total health of all monsters after they pass all towers.

Examples

input
3 5 1 7 4 4 2 4 0 14 1 10 3 16 10 16
output
4

input
5 2 1 4 1 5 4 7 5 8 3 9 1 21 2 18 3 14 4 24 5 8 6 25 7 19 8 24 9 24
output
40

Codeforces (c) Copyright 2010-2022 Mike Mirzayanov
The only programming contests Web 2.0 platform