BUET Inter University Programming Contest 2023

https://toph.co/c/buet-inter-university-2023

Schedule

The contest will run for **5h0m0s**.

Authors

The authors of this contest are Ahmed.032170, Anachor, Arghya, Iftekhar_Hakim, longlongint, LUMBERJACK_MAN, N3710rD, neo11235, Sk_Sabit, tbs_sajal15, and upobir.

Rules

This contest is formatted as per the official rules of ICPC Regional Programming Contests.

You can use Bash 5.0, Brainf*ck, C# Mono 6.0, C++11 GCC 7.4, C++14 GCC 8.3, C++17 GCC 9.2, C++20 GCC 12.1, C11 GCC 12.1, C11 GCC 9.2, Common Lisp SBCL 2.0, Erlang 22.3, Free Pascal 3.0, Go 1.18, Grep 3.7, Haskell 8.6, Java 1.8, Kotlin 1.1, Lua 5.4, Node.js 10.16, Perl 5.30, PHP 7.2, PyPy 7.1 (2.7), PyPy 7.1 (3.6), Python 2.7, Python 3.11, Python 3.7, Ruby 2.7, Ruby 3.2, Rust 1.57, Swift 5.3, and Whitespace in this contest.

Be fair, be honest. Plagiarism will result in disqualification. Judges' decisions will be final.

Notes

There are 12 challenges in this contest.

Please make sure this booklet contains all of the pages.

If you find any discrepencies between the printed copy and the problem statements in Toph Arena, please rely on the later.

A. Satisfaction

You have vacation for the next N days. Each day you will either create a problem or you will hold a contest with **exactly** one problem. Formally speaking, your action for the next N days can be described by a string of length N containing characters 'P' and 'C'. If the i^{th} character is 'P' then you will be creating a problem on i^{th} day, if i^{th} character is 'C' then you will hold a contest on that day with a problem that is created before i^{th} day and **unused** till then. Each problem has an interesting factor and each contest has an importance factor. If you put a problem with interesting factor x to a contest of importance factor y, then you get $x \times y$ satisfaction.

You need to put problems to contests in such a way that the sum of your satisfaction is **maximized**. Also the interesting factor of the problems are **non increasing** as your are not getting any sharper day by day. Input will be generated in a way that, for each contest you will have at least one problem available and as you love contests, you will not miss the chance to be the problem-setter in any contest.

Input

First line contains number of test cases $T (1 \leq T \leq 10000)$.

Each testcase starts with an integer $N (1 \leq N \leq 10^6)$.

The next line contains a string $A (A[1] \ldots A[N])$ containing N characters, each character is either 'P' or 'C'.

The next line contains N space separated integers $V[1], V[2], \ldots, V[N]$, if $A[i] =$ 'P' then $V[i]$ denotes interesting factor of the problem that you create on i^{th} day, else $V[i]$ denotes importance factor of the contest on i^{th} day.

For all $1 \leq i \leq N, 1 \leq V[i] \leq 10^6$

If $i < j$ and $A[i] = A[j] =$ 'P' then $V[i] \geq V[j]$

It's guaranteed that the sum of N over all test cases does not exceed 10^6.

Output

For each case, output a single integer, which denotes the maximum satisfaction you can get if you optimally place the problems in the contests.

Samples

Input	Output
2 5 PPCPC 9 8 9 6 6 5 PPCPC 9 8 1 7 9	129 89

For the first testcase,
Answer is $9 * 9 + 6 * 8 = 129$. On the 1^{st} contest (day 3) you will put the problem created on day 1 and on the 2^{nd} contest (day 5) you will assign the problem created on day 2.

For the second testcase,
Answer is $8 * 1 + 9 * 9 = 89$. On the 1^{st} contest (day 3) you will put the problem created on day 2 and on the 2^{nd} contest (day 5) you will assign the problem created on day 1.

B. Peculiar Partitioning II

You are given an array a of length n. Each element of the array is a non-negative integer less than 2^k. You have to partition it into two non-empty sub-sequences X and Y such that each element belongs to exactly one of X or Y. The score of such a partitioning is given by $(x_1 \wedge x_2 \cdots \wedge x_{|X|}) \vee (y_1 \wedge y_2 \cdots \wedge y_{|Y|})$. Here, $\wedge$ denotes the bitwise AND operation and $\vee$ denotes the bitwise OR operation.

For each x between 0 and $2^k - 1$, find the number of ways to partition the array such that the score is x. Since this number can be large, you only need to find it modulo 998244353.

Two ways of partitioning are considered different if and only if there exists a pair of indices i, j such that they are in the same part in one way, but in different parts in the other.

Input

The first line contains two space separated integers, n and k.

The second line contains n space-separated integers $a_1, a_2, \cdots, a_n$.

Constraints

- $2 \leq n \leq 3 \times 10^5$
- $1 \leq k \leq 18$
- $0 \leq a_i < 2^k$

Output

Output 2^kspace separated integers $f_0, f_1, \cdots, f_{2^k-1}$. where, f_x is the number of ways, modulo 998244353, to partition such that the score is x.

Samples

Input	Output
4 2 1 1 2 3	0 4 0 3

Input	Output
5 3 5 1 4 2 3	4 5 2 1 2 1 0 0

Input	Output
2 1 0 1	0 1

C. The Very Last Exam

You have spent the last twenty years studying competitive programming. Now only one exam stands between you and becoming the very best. You open the exam paper and see a single problem.

You are given n bags, each containing some distinct stones of varying weights. From each bag, you can pick **at most** one stone. You are also given a target fraction P/Q. You have to pick stones so that their average weight is exactly P/Q. How many ways can this be done so? Output the count of such ways.

Note that two ways of picking stones are different if there is some bag from which you pick a stone in one way but not in the other or if from some bag you pick different stone (different stones can have same weight).

Input

First line of input will have a integer T denoting the number of independent testcases.

Each testcase will start with a line containing one integer, n denoting number of bags.

Next n lines will contain the description of bags.

Each line will contain several space separated integers. They will start with an integer k denoting the number of stones in the bag. Next k integers will denote the weights of the stones in the bag. Final line of the testcase will contain two integers P and Q denoting the target fraction P/Q.

Constraints

- $1 \leq T \leq 10$
- $1 \leq n \leq 10^5$
- $1 \leq k \leq 99$
- $1 \leq$ weight of each stone $\leq 10^9$
- $1 \leq P, Q \leq 10^{18}$
- Product of $(k+1)$ over all bags in a testcase will be at most 10^{12}

Output

For each testcase output the count of ways in a single line.

Samples

Input	Output
2 2 2 1 1 2 1 1 1 1 3 2 2 3 3 1 4 2 3 3 2 1 5 2	8 5

In the first testcase, among the nine possible combinations of picking stones, one picks none of the stones and other eight picks stones in such a way that the average is exactly $1 = \frac{1}{1}$.

D. Festive Shuffling

You are given a **binary** string S and k distinct indices to perform an operation.

The operation is like below:

You can shuffle the given indices any number of times (possibly zero).

You have to find out the **maximum** length of a **palindromic** subsequence of string S after shuffling.

Example of an operation: Say S = "001110", Let the indices be $[1, 3, 5]$. Then after the operation the string content can be, 100110.

Input

The first line of input contains a string S and an integer k

The next line will contain k space separated integers $a_1, a_2, \ldots, a_k$ denoting the indices that you can shuffle, all of them are guaranteed to be distinct.

Constraints

$1 \leq |S| \leq 500$

$0 \leq k \leq |S|$

$1 \leq a_i \leq |S|$

Output

Output one line containing the output, the maximum length of any palindromic subsequence after the operation.

Samples

Input	Output
10010 2 1 2	5

E. Ankara Messi

It is Argentina vs Croatia in the football world cup semifinal of 2022 in a parallel universe.

The football field can be considered as an infinite 2- dimensional plane. Lionel Messi of Argentina has the ball. He is at coordinate $(x, 0)$. Joško Gvardiol, a defender of Croatia is at $(x - 1, 0)$. At any moment, Gvardiol runs directly towards Messi at speed 1 unit per second. Messi knows the movement strategy of Gvardiol. Messi himself has a maximum speed of v unit per second.

Gvardiol will *tackle* Messi if the distance between them becomes **strictly** less than 1 unit. Gvardiol has a choice to *foul* Messi at a moment when Messi moves from a position which is more than 1 unit away to a position which is exactly 1 unit away from Gvardiol. A *dribble* is a path on the field which Messi can take to move to coordinate $(0, 0)$ without getting *tackled* or *fouled*.

Messi does not know if Gvardiol intends to *foul* or not. So to be safe, he plans his *dribble* assuming that Gvardiol will *foul* him given the chance.

A *dribble* X is called *boring* if Messi can make a *dribble* shorter than X if he knows beforehand that Gvardiol won't *foul*.

Messi wants to know the length of the shortest *dribble* which is **not** *boring*. It can be proved that such a *dribble* exists under the given constraints.

Input

First line contains an integer T $(1 \leq T \leq 5 \times 10^4)$, the number of test cases.
Each of the next T lines contains two space separated floating point numbers x $(2 \leq x \leq 100)$ and v $(1 < v \leq 100)$. Both numbers have 5 digits after decimal.

Output

Print the answer for each case in a separate line. Your answer is considered correct if its absolute or relative error does not exceed 10^{-6}. Formally, let your answer be A, and the jury's answer be B. Your answer is accepted if and only if $\frac{|A-B|}{\max(1,B)} \leq 10^{-6}$.

Samples

Input	Output
2 2.50000 2.60000 10.00000 100.00000	4.374718737669 10.637610774649

F. Chocolate

- *BUET CSE FEST'23 has come to an end!!!! Chimatu Can't stop crying as he already started to miss the festive vibe so much. To make him happy again you need to buy some chocolates for him. But yeah you are a well known miserable person, and so that you want to spend as less money as you can.*

Now comes the real problem. You need to buy n chocolates.

You can buy 2 types of chocolates. Each type has a **infinite** amount of chocolates.

The **first** type chocolate has an initial price p, where the **second** type chocolate has an initial price q.

Each time one buys a chocolate of the first type, the price of this type gets decreased by x. Similarly, each time a second type chocolate is bought, the price of this type gets decreased by y.

Say $p = 9$ and $x = 3$ for the first type of chocolate. Currently the price is 9. After buying one chocolate the price decreases by 3 and now the new price is 6. After buying another chocolate the price will again decrease by 3 and the new price will be 3.

You have to find the **minimum cost** to buy n chocolates and make Chimatu happy!!!

[Note: It is guaranteed that the price of any chocolate will never be negative]

Input

The first line of the input contains an integer t denoting the number of test cases.

The next line of each test case contains 5 space separated integers.

- n- The amount of chocolates to be bought.
- p- The initial price of first type chocolate
- q- The initial price of second type chocolate
- x- The amount of price reduces after buying each chocolate of first type
- y- The amount of price reduces after buying each chocolate of second type

Constraints

$1 \leq t \leq 10^5$

$1 \leq n \leq 10^6$

$1 \leq x \leq p \leq 10^7$

$1 \leq y \leq q \leq 10^7$

Output

The output should contain t lines. ith line of output should contain the **minimum cost** for ith test case.

Samples

Input	Output
1 5 100 50 5 3	220

G. LR

This task is simple. You will be given 2 positive integers L and R.

Find the number of positive even integers x such that $L \leq x \leq R$.

Input

The first line of input contains an integer T denoting the number of test cases.

The only line of each test case contains two integers L and R.

Constraints

$1 \leq T \leq 100$

$1 \leq L \leq R \leq 1000$

Output

Output one integer, the answer to the problem.

Samples

Input	Output
1 1 2	1

H. Better Together!

There are N cities and initially 0 roads between them. You are the Civil Engineer (but a programmer from heart!) who is in charge of constructing roads between these cities.

The cities are numbered from 0 to $N-1$. There are M types of road constructions which can be performed among these cities. For each $k = 1, 2, 3, \ldots, M$, the k-th kind of road construction is to choose an integer x such that $0 \leq x < N$ and construct a new two-way road connecting city x and city $(x + A_k) \bmod N$. Performing the k-th kind of road construction once costs C_k taka.

You can do these M kinds of road constructions any number of times (possibly zero) in any order. For example, if three kinds of road constructions are available, you can choose to perform the first type of road construction twice, the second zero times, and the third once.

The people of these cities want to travel from one city to another city. You have to determine whether it is possible to make all the cities connected together. If possible, then determine the **minimum** total cost. Here, "connected" means, there exists at least one path from each city to every other city.

Constraints:

- $2 \leq N \leq 10^9$
- $1 \leq M \leq 10^5$
- $1 \leq A_k \leq N - 1$
- $1 \leq C_k \leq 10^9$
- All values in input are integers.

Input

Input is given from Standard Input in the following format:

$N\ M$
$A_1\ C_1$
$A_2\ C_2$
$\vdots$
$A_M\ C_M$

Output

If it is possible to make all the cities connected, print the **minimum** total cost needed to do so.
If it is impossible to make all the cities connected, print -1.

Samples

Input	Output
4 2 2 3 3 5	11
If we first do the road construction of the first kind to connect cities 0 and 2, then do it again to connect cities 1 and 3, and finally the road construction of the second kind to connect cities 1 and 0, all the cities will be connected together. The total cost incurred here is $3 + 3 + 5 = 11$ taka, which is the minimum possible.	

Input	Output
6 1 3 4	-1
There is no way to make all the cities connected, so we should print -1.	

I. Reassemble

There are N posts in a line numbered from 1 to N from left to right. Each post i has a distance d_i which describes how far it is from the base of operation. The more left you go, the closer you get to the base. More formally, $1 \leq d_1 < d_2 < d_3 < \ldots < d_N$.

Each post has a different number of soldiers. Post i has a troop of i soldiers. Now it is time for the drill and you need to shuffle the positions of the troops. You want the troop i to go the position P_i for all $1 \leq i \leq N$.

To keep the discipline, you will give the following commands one at a time. Choose two distinct posts, i and j where $(1 \leq i, j \leq N)$, and swap the troops. All soldiers currently stationed at post i will go to post j and all soldiers currently stationed at post j will go to post i. When moving from post i to j, one soldier needs to walk $|d_i - d_j|$ unit of distance.

You can give **as many** commands as you want until each troop is at its desired position but you want to do it such that the total distance crossed by all soldiers is as **minimum** as possible. You also want to **minimize** the total number of commands **after** minimizing total distance crossed.

Input

The first line of the input will contain an integer, T $(1 \leq T \leq 100)$ which denotes the number of test cases.

Each of the test cases starts with one integer N $(1 \leq N \leq 10^6)$ describing the number of posts.

The next line will contain N space separated integers $d_1,\ d_2,\ d_3, \ldots,\ d_N$ where d_i denotes the distance of post i from base. $1 \leq d_i \leq 10^7,\ d_i < d_{i+1}$ for all $1 \leq i < N$.

The next line will contain N more space separated integers $P_1,\ P_2,\ P_3, \ldots,\ P_N$. Troop currently stationed at post i should be stationed at post P_i after all the commands are given. $1 \leq P_i \leq N$ and all P_i are distinct, so P is a **permutation** of length N.

It is guaranteed that the sum of N over all test cases does not exceed 10^6.

Output

For each test case output two integers C and Q in the first line, where C is the minimum distance crossed by all soldiers and Q is the minimum number of commands needed to do that. Then output Q more lines, each of them will contain two integers i, j $(i \neq j,\ 1 \leq i, j \leq N)$ which denotes a command. **NOTE: Troop currently stationed at post i must be stationed at post P_i after all Q commands are applied in chronological order. First you need to minimize C, then minimize Q. If there are multiple sets of commands of size Q so that the total distance crossed remains C, print any of them.**

Samples

Input	Output
2 3 1 10 20 2 3 1 4 1 2 3 4 3 4 1 2	86 2 3 2 2 1 20 2 3 1 4 2

J. Eulers Peculiar Dream

After discovering the famous totient function $\phi(x)$ Euler got so fond of it that he applied it to every number he could find on his table. He played all day with this newly discovered function. Then he went to sleep. He was dreaming about the function too.

In the dream, he found a very big number x. Since he was playing with totient function he immediately applied totient function to x and got a new number $\phi(x)$. But the number was still very large so he again applied totient to it and got another number $\phi(\phi(x))$ which was still way too big. As a result, Euler got frustrated. He wondered how many times he had to apply totient function before reaching 1. Euler solved it in the dream. Can you do it too?

Formally, let $y_0 = x$

and $y_n = \phi(y_{n-1})$ where n is an integer and $n \geq 1$

Find **smallest** n such that $y_n = 1$

Note: Euler totient function of a positive integer n is defined as the number of positive integers i less than or equal to n such that $gcd(n, i) = 1$

Input

First line of input contains an integer $k (1 \leq k \leq 10^5)$

Next line contains k prime numbers $p_1, p_2, \ldots p_k$

Next line contains k integers $e_1, e_2, \ldots e_k$

For each $1 \leq i \leq k$, $2 \leq p_i \leq 10^6$

It is guaranteed that each p_i is **pairwise distinct**.

For each $1 \leq i \leq k$, $1 \leq e_i \leq 10^6$

Then the input number is calculated as $x = \prod_{i=1}^{k} p_i^{e_i}$

Output

Print the **smallest** positive n such that $y_n = 1$

Since n can be very large, output it modulo 998244353

Samples

Input	Output
2 3 2 1 1	2

Input	Output
1 11 1	4

K. Cold Rainy Night in Stoke

Night is ending. You have just got N points in 2D-coordinate plane in your dream. Wait! Things were going well with the points. Suddenly, a circle appeared, which is centered at coordinate $(0, 0)$ and has a radius of R. Your mind is telling you to create a **convex** polygon, whose vertices are **subset** of those N points. This convex polygon must enclose the circle completely. That is, no point of the circle (including the points on its boundary) is **strictly** outside the polygon. The polygon may touch the circle.

This is not enough. You have to find the polygon with **smallest** area. Output the value of the smallest area$\times 2$. If it is impossible to construct such a polygon, output should be -1.

Let's just solve this problem before the night ends.

Input

First line of the input will have an integer T $(1 \leq T \leq 100)$, denoting the number of testcases. Each testcase is independent.

First line of each testcase will have an integer N $(1 \leq N \leq 500)$, denoting the number of points you have got.

Line i of the next N lines will have two integers x_i and y_i $(-10^4 \leq x_i, y_i \leq 10^4)$ denoting the abscissa and ordinate i-th point respectively, for $i = 1, 2, \ldots, N$.

Last line of each testcase will have an integer R $(1 \leq R \leq 10^4)$, denoting the radius of the circle.

Sum of N over all testcases will not exceed 500.

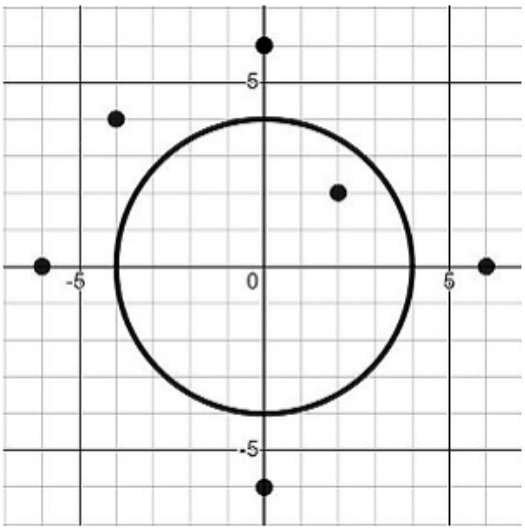
Output

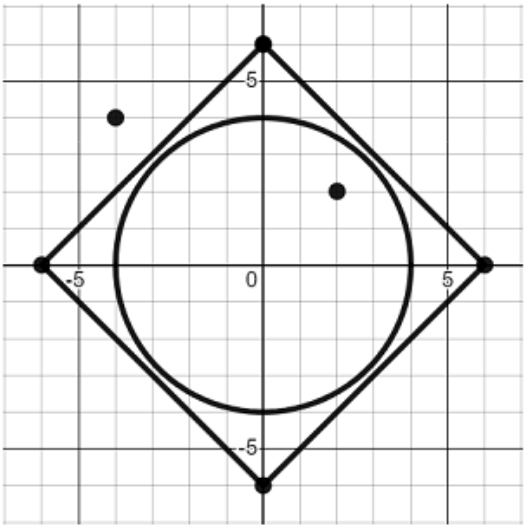
Print a separate line for each testcase. Line i of the output should have the output for i-th testcase. Output of the i-th testcase is the value of $2 \times A$, where A is the area of the smallest possible convex polygon. If there is no possible convex polygon, output is -1.

Samples

Input	Output
2 6 -6 0 -4 4 0 6 0 -6 2 2 6 0 4 1 5 5 1	144 -1

Following figures show the input of 1st testcase and the convex polygon with smallest area. Here the area is 72.





L. Gardening

You are given a tree with N vertices. Here, a tree is an undirected graph in which any two vertices are connected by ***exactly*** *one simple* path, or equivalently a connected **acyclic** undirected graph. A tree with N vertices, contains exactly $N-1$ undirected edges.

You have to process Q queries. In each of the query, you will be given 3 positive integers R, U and V. You have to perform the following operations on your tree —

- Add an edge between U and V.
- Remove an edge such that the resulting graph is still a **tree**. You can also remove the newly added edge.
- **Minimize** the sum of distance of every vertex from the vertex R. Formally , let d_i denote the shortest distance from vertex R to vertex i. You have to minimize $\sum_{1 \leq i \leq n} d_i$
- Print the **Minimized Sum**

Note that, Each query is **independent** which means operations in one query does not affect the other queries.

Input

The first line of the input will contain an integer, $N\ (2 \leq N \leq 3 \times 10^5)$ which denotes the number of vertices of the tree.

Each of the next $N-1$ lines will contain 2 space-separated integers u_i and $v_i (1 \leq u_i, v_i \leq N)$, indicating an edge between vertices u_i and v_i. It is guaranteed that the given edges form a tree.

The next line will contain an integer $Q\ (1 \leq Q \leq 3 \times 10^5)$, the number of queries.

Finally, each of the next Q line will contain 3 integers R_i,U_i and $V_i (1 \leq R_i, U_i, V_i \leq N)$ — representing each query.

Output

Print the answer for each query in separate lines.

Samples

Input	Output
5 2 1 3 1 4 3 5 1 4 4 2 1 2 1 2 4 2 1 4 4 1	9 8 9 6

Input	Output
6 1 2 2 3 3 4 4 6 1 5 2 1 5 6 1 5 4	9 9

Input	Output
10 2 1 3 2 4 3 5 3 6 4 7 2 8 4 9 8 10 7 2 1 10 4 9 1 9	27 23