A. Houses

2 seconds, 256 megabytes

There are 10^6 houses arranged in a line, numbered from 1 to 10^6 from left to right.

Emma lives in the x_1-th house. She wants to visit her friend Lucia, who lives in the x_2-th house.

Before arriving to Lucia's house, Emma wants to buy a chocolate bar for Lucia. Emma knows that there is a store situated in the x-th house, where she can buy a chocolate bar.

Now Emma wants to know whether she can visit the store on the path from her house to Lucia's house. She can visit the store if the house x either coincides with her house, coincides with Lucia's house, or is located between them.

Input

The first line contains one integer t ($1 \le t \le 10^3$) — the number of test cases.

Each test case consists of a single line containing three integers x_1, x_2, x ($1 \le x_1, x_2, x \le 10^6$; $x_1 \ne x_2$) — the location of Emma's house, Lucia's house and the store, respectively.

Output

For each test case, print one string `YES` or `NO` on a separate line. Print `YES` if Emma can visit the store on the road from her house to Lucia's house; otherwise, print `NO`.

input
3 8 9 7 9 5 7 5 9 7
output
NO YES YES

B. Books in Boxes

2 seconds, 256 megabytes

Monocarp has two boxes of books. The first box contains a books, and the second contains b books.

Monocarp can take books from one box and transfer them to the other box an unlimited number of times. He can only transfer books between the boxes; he cannot remove or add new books.

Let c be the number of books in the first box after the transfer, and d be the number of books in the second box. Monocarp wants both numbers c and d to end with the digit k. Your task is to determine if it is possible to transfer the books in such a way that Monocarp's goal is fulfilled.

Input

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

The only line of each test case contains three integers a, b and k ($1 \le a, b \le 1000$; $0 \le k \le 9$).

Output

For each test case, print `YES` if it is possible to transfer the books in such a way that Monocarp's goal is fulfilled; otherwise, print `NO`.

input
4 13 45 4 7 993 0 24 13 5 25 15 5
output
YES YES NO YES

In the first example, you can transfer one book from the second box to the first box. Then the first box will contain 14 books, and the second box will contain 44 books. Thus, the number of books in each box will end with the digit 4.

In the second example, you can transfer 7 books from the first box to the second box. Then the second box will contain 1000 books, and the first box will contain 0 books. Thus, the number of books in each box will end with the digit 0.

In the third example, it is impossible to transfer the books in such a way that all the described conditions are met.

In the fourth example, the number of books in each box ends with the digit 5 already.

C. Decompression

2 seconds, 256 megabytes

Consider the following method to compress a string s consisting of uppercase Latin letters:

1. First, the string is split into several (maybe just one) blocks of consecutive characters. Each block should consist of at least 1 and at most 9 characters, and all the characters in each block should be equal.
2. Then, each block is replaced by the string consisting of two characters k and c, where k is the digit denoting the length of the block, and c is the Latin letter that the block consists of. The characters k and c can go in any order; i. e. the block `AAA` can be replaced by either `A3` or `3A`.
3. Then, the strings that replaced the block are concatenated without changing their order.

For example, this is one way to compress the string `WWWOOWW`:

1. Split it into blocks `WW`, `W`, `OO` and `WW`.
2. Replace the blocks with `2W`, `W1`, `O2` and `W2`.
3. Concatenate everything into `2WW1O2W2`.

You are given the string w, which is the compressed version of the string s. Restore the original string s.

Input

The first line contains one integer t ($1 \le t \le 100$) — the number of test cases.

Each test case consists of one line, containing the string w ($2 \le |w| \le 50$, $|w|$ is even) consisting of uppercase Latin letters and digits. The string w is a compressed version of some string s.

Output

For each test case, print the original string s in a separate line. It can be shown that the original string can be restored uniquely.

input
4 2WW1O2W2 R2D2 H1E1L1L1O11W1O1R1L1D H1E12LO11W1O1R1L1D

output
WWWOOWW RRDD HELLOWORLD HELLOWORLD

D. Letter

2 seconds, 256 megabytes

Monocarp wants to send a letter. His letter consists of n words. Unfortunately, the maximum length of one message Monocarp can send is limited to k characters, so Monocarp may have to split his letter into multiple messages.

The letter can be split into groups of consecutive words, but it is impossible to split the words themselves. In each group, each pair of adjacent words should be separated by a space. Note that Monocarp can't change the order of words in a letter. Each resulting part of the letter should consist of at most k characters (including the spaces between words).

For example, if a letter looks like "`please buy the cereal`", and $k = 10$, then the letter can be split into two messages "`please buy`" and "`the cereal`", each consisting of 10 characters.

Your task is to calculate the minimum number of messages required to send all n given words. Remember that you can't reorder the words.

Input

The first line contains two integers n and k ($1 \le n \le 10^3$; $1 \le k \le 2 \cdot 10^5$) — the number of words in the letter and the maximum message length, respectively.

The next n lines contain words of the letter, one per line — strings consisting of lowercase Latin letters. Each word consists of at most 100 characters.

Additional constraint on the input: the letter can be split into messages, i. e. the maximum length of the word in the input does not exceed k.

Output

Print a single integer — the minimum number of messages required to send all n given words.

input
4 10 please buy the cereal

output
2

input
5 6 this is a test letter

output
4

input
6 200000 a very large value of k
output
1

E. Dice Game

2 seconds, 256 megabytes

Two people are playing a dice game. Each die is not necessarily a 6-sided one; a die can have any integer number of faces from 4 to 50.

The result of a die roll is a random number from 1 to the number of faces.

The first player has n dice, with the i-th die having a_i faces. The second player has m dice, with the j-th die having b_j faces.

They roll all their dice at the same time. Whoever has the highest total sum wins. A draw is declared if their sums are equal.

Your task is to determine the possibility of the following game results:

- can the game end with a 1-st player victory;
- can the game end in a draw;
- can the game end with a 2-nd player victory.

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 two integers n and m ($1 \le n, m \le 50$) — the number of dice for the first player and the number of dice for the second player, respectively.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($4 \le a_i \le 50$) — the dice that the first player has.

The third line contains m integers $b_1, b_2, \ldots, b_m$ ($4 \le b_j \le 50$) — the dice that the second player has.

Output

For each test case, print a string consisting of 3 characters `Y` and/or `N` according to the following rules:

- the first character is `Y` if the game can end with a 1-st player victory, otherwise it is `N`;
- the second character is `Y` if the game can end in a draw, otherwise it is `N`;
- the third character is `Y` if the game can end with a 2-nd player victory, otherwise it is `N`;

input
3 2 3 7 4 5 6 4 1 5 4 4 6 8 4 5 4 3 4 4 4 4 7 7 7
output
YYY NNY YYY

F. City Plan

2 seconds, 256 megabytes

There are n houses in one particular city that were built along one straight road. For simplicity, we can say that the i-th house is located at integer point x_i ($0 \le x_i \le 10^9$, all x_i are distinct).

You are participating in one project, and you were asked to calculate for each house i the distance d_i from the i-th house to the farthest other house.

Let's say that the *distance* between the houses i and k is equal to $|x_i - x_k|$, where $|a|$ is the absolute value of a. Then d_i can be calculated as $\max\limits_{1 \le k \le n} (|x_i - x_k|)$.

Spending some time, you finally calculated all n values $d_1, d_2, \ldots, d_n$. But, at the same time, you've lost a piece of paper containing a list of all house positions x_i.

You are too lazy to measure all house positions, and you believe that actual house positions are not so important for the project. So the question is: can you create an array $x_1, x_2, \ldots, x_n$ that is consistent with the array d you've calculated?

Input

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

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

The second line of each test case contains n integers $d_1, d_2, \ldots, d_n$ ($1 \le d_i \le 10^7$), where d_i is the maximum distance from house i to some other house.

Additional constraints on the input:

- the array d is correct, i. e. there exists at least one array x that is consistent with array d;
- the total sum of n among all test cases doesn't exceed $2 \cdot 10^5$.

Output

For each test case, print n integers $x_1, x_2, \ldots, x_n$ ($0 \le x_i \le 10^9$) — the positions of houses that produce the given array d. All x_i must be distinct.

If there are several solutions, print any. Note that the order of x_i matters since it affects the value of d_i.

input
4 2 24 24 5 4 4 3 3 2 3 5 10 10 2 10000000 10000000

output
13 37 0 4 1 3 2 15 20 10 1000000000 990000000

G. Coin Tossing

1 second, 512 megabytes

Consider an experiment where a coin is tossed n times. During the experiment, the results of each iteration were recorded, so you got a string s of n characters `H` and/or `T`. These characters denote the results of coin tosses: `H` means "heads", and `T` means "tails".

A string is called **anomalous** if the number of pairs of **adjacent** characters which are different from each other is not greater than k. For example, in the string `HTTTHHTT`, there are 3 pairs of adjacent characters meeting this constraint; so if $k = 3$ or $k = 4$, it is anomalous; but if $k = 2$, it is not.

You have to find the longest **anomalous** substring of s. A substring of s is a contiguous subsequence of s, i .e it is a string that can be obtained from s by removing several (possibly zero) characters from the beginning and several (possibly zero) characters from the end of the string.

Input

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

Each test case consists of two lines:

- the first line contains two integers n and k ($1 \le n \le 3 \cdot 10^5$; $0 \le k \le n - 1$);
- the second line contains the string s, consisting of exactly n characters. Each character is either `H` or `T`.

Additional constraint on the input: the sum of n over all test cases does not exceed $3 \cdot 10^5$.

Output

For each test case, print one integer — the length of the longest anomalous substring of s.

input
4 10 1 HHTTTTHTTT 10 2 HHTTTTHTTT 10 4 HHTTTTHTTT 9 0 HHHHHHHHH
output
6 8 10 9

In the first test case, the longest anomalous substring is `HHTTTT`.

In the second test case, the longest anomalous substring is `TTTTHTTT`.

In the third test case, the whole string is anomalous. Same for the fourth test case.

H. GCD Set

2 seconds, 256 megabytes

You are given two integers n and x. Build an array of n elements $a_1, a_2, \ldots, a_n$ such that:

1. all a_i are positive integers greater than zero and less than or equal to 10^{18};
2. all a_i are pairwise distinct;
3. $\gcd(a_i, a_j) = x$ for each pair $i < j$, where $\gcd(a_i, a_j)$ is the greatest common divisor of a_i and a_j.

Input

The first line contains a single integer t ($1 \le t \le 50$) — the number of test cases. Next t cases follow.

The first line of each test case contains two integers n and x ($2 \le n \le 100$; $1 \le x \le 10^9$) — the size of array a to build and the greatest common divisor of each pair of elements.

Output

Print n integers $a_1, a_2, \ldots, a_n$ ($0 < a_i \le 10^{18}$) — the array you need to build. If there are several answers, print any of them.

It can be proven that such an array always exists within the given constraints.

input
4 2 1 2 1000000000 5 1 3 3

output
1 2 1000000000000000000 3000000000 31 9 23 5 28 6 69 969

I. Pairing Numbers

2 seconds, 256 megabytes

You are given n integers $a_1, a_2, \ldots, a_n$. You can pair some of the numbers and leave some unchanged. After pairing two numbers x and y, they are replaced by a single number $x \cdot y$. All pairings occur simultaneously, i. e. it is not allowed to pair two numbers and then pair the resulting product with another number. Note that each of the original numbers must either remain unpaired or belong to **exactly one** pair.

For example, if you have 5 numbers $[0, -1, 5, 3, 3]$, you can pair the number 0 with the number 5, and also the number -1 with the number 3, then after pairing is done, there are three numbers: $[0, -3, 3]$.

Your task is to pair some numbers in such a way as to maximize the sum of the resulting numbers.

Input

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

The first line of each test case contains a single integer n ($1 \le n \le 100$).

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($-1000 \le a_i \le 1000$).

Output

For each test case, print a single integer — the maximum sum of numbers that you can obtain after pairing some of the numbers.

input
5 4 3 3 -2 -4 3 1 1 1 3 5 0 -5 10 4 5 -4 3 -1 1 0 2 -1 1 1 -25

output
17 3 5 32 -25

J. Pirate Chest

2 seconds, 256 megabytes

Welcome to the Treasure Island! According to the gossip you've heard, there is an old pirate chest with gold buried somewhere on the island.

The island is very long and thin, so it can be treated as a line segment. According to the gossip, the chest is buried **somewhere in an integer point from 0 to 10^9 on this line** (this gossip is very specific — perhaps you've heard it from a competitive programmer).

You've also got a torn sheet from an old pirate journal. This page lists n different points $a_1, a_2, \ldots, a_n$. Also, for every pair of consecutive points (a_{i-1} and a_i for each $i \in [2, n]$), it lists an integer b_i, which is either 1 or 2. The reverse side of the sheet explains that these are the clues to the treasure location:

- if the treasure is closer to the point a_i than to the point a_{i-1}, b_i is 1;
- if the treasure is closer to the point a_{i-1} than to the point a_i, b_i is 2;
- if the distance from the treasure to the point a_i is the same as the distance from the treasure to the point a_{i-1}, b_i can be any value (1 or 2).

In order to analyze how much time you have to spend digging for treasure, you have to calculate the number of possible treasure locations.

Input

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

Then n lines follow. The i-th line contains the integer a_i ($-10^9 \le a_i \le 2 \cdot 10^9$). If $i > 1$, the i-th line also contains a second integer b_i ($1 \le b_i \le 2$).

Additional constraint on the input: all points a_i are distinct.

Output

Print one integer — the number of integer points **from 0 to 10^9** that may contain treasure according to the clues from the journal. If the clues are contradictory and no point from 0 to 10^9 can contain treasure, print 0.

input
3 5 4 1 3 2
output
1

input
2 13 42 2
output
28

input
2 8 9 1
output
999999992

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

input
3 13 37 2 42 1
output
0

In the first example, the point 4 is the only point that is both closer to 4 than to 5, and closer to 4 than to 3.

In the second example, the points $0, 1, 2, \ldots, 27$ are closer to 13 than to 42.

In the third example, the suitable points are $9, 10, 11, \ldots, 10^9$.

In the fourth example, the only suitable point is 3.

In the fifth example, no point is both closer to 42 than to 37, and closer to 13 than to 37.

K. Forum

2 seconds, 256 megabytes

You are developing a new feature for the forum. You have to perform q queries. Each query has one of two types:

- "1 x" — add a new post with a rating x to the forum. The ID of this post is equal to the index of the query when it is added;
- "2 y k" — print the IDs of k recently added posts with a rating of at least y. If the number of posts with the required rating is less than k, print the IDs of all suitable posts.

Input

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

Next q lines contain queries (one per line). Each query has one of two types:

- "1 x" ($1 \le x \le 10^9$);
- "2 y k" ($1 \le y \le 10^9$; $1 \le k \le q$).

Additional constraints on the input:

- the sum of k over all queries does not exceed $2 \cdot 10^5$;
- there is at least one query of the second type.

Output

For each query of the second type, print the answer in the following way:

- in the first line, print a single integer m — the number of suitable posts (with rating of at least y), but no more than k;
- in the second line, print m integers $a_1, a_2, \ldots, a_m$ — the IDs of m recently added suitable posts in descending order (i. e. $a_1 > a_2 > \cdots > a_m$).

input

```
10
2 1 3
1 4
1 2
2 3 1
1 5
1 1
2 1 2
2 4 5
1 4
2 3 2
```

output

```
0

1
2
2
6 5
2
5 2
2
9 5
```

L. Roads

2 seconds, 256 megabytes

There are n cities, numbered from 1 to n. You are assigned a task to construct bidirectional roads between these cities.

Initially, there are no roads. You can choose any set of roads to construct, according to the following conditions:

- each road should connect two different cities;
- for each pair of cities, there should be at most one road connecting them;
- the road network should be connected, i. e. every city should be reachable from every other city along the roads;
- let d_i be the number of roads on the **shortest** path from the city 1 to the city i. The condition $d_2 \le d_3 \le \cdots \le d_n$ should hold;
- for every city i from 2 to n, there should be **exactly** one path from the city 1 to the city i consisting of d_i roads (i. e. exactly one shortest path).

Calculate the number of different sets of roads meeting these constraints, and print it modulo 998244353. Note that the order of roads in the set does not matter.

Input

The only line contains one integer n ($2 \le n \le 100$).

Output

Print one integer — the number of sets of roads meeting the constraints, taken modulo 998244353.

input
2
output
1

input
3
output
3

input
4
output
15

input
42
output
905088720

For $n = 2$, the only possible set of roads consists of a single road $(1, 2)$.

For $n = 3$, the possible sets of roads are:

- $(1, 2)$, $(1, 3)$;
- $(1, 2)$, $(1, 3)$, $(2, 3)$;
- $(1, 2)$, $(2, 3)$.