Educational Codeforces Round 88 (Rated for Div. 2)

A. Berland Poker

2 seconds, 256 megabytes

The game of Berland poker is played with a deck of n cards, m of which are jokers. k players play this game (n is divisible by k).

At the beginning of the game, each player takes $\frac{n}{k}$ cards from the deck (so each card is taken by exactly one player). The player who has the maximum number of jokers is the winner, and he gets the number of points equal to $x - y$, where x is the number of jokers in the winner's hand, and y is the maximum number of jokers among all other players. If there are two or more players with maximum number of jokers, all of them are winners and they get 0 points.

Here are some examples:

- $n = 8, m = 3, k = 2$. If one player gets 3 jokers and 1 plain card, and another player gets 0 jokers and 4 plain cards, then the first player is the winner and gets $3 - 0 = 3$ points;
- $n = 4, m = 2, k = 4$. Two players get plain cards, and the other two players get jokers, so both of them are winners and get 0 points;
- $n = 9, m = 6, k = 3$. If the first player gets 3 jokers, the second player gets 1 joker and 2 plain cards, and the third player gets 2 jokers and 1 plain card, then the first player is the winner, and he gets $3 - 2 = 1$ point;
- $n = 42, m = 0, k = 7$. Since there are no jokers, everyone gets 0 jokers, everyone is a winner, and everyone gets 0 points.

Given n, m and k, calculate the maximum number of points a player can get for winning the game.

Input

The first line of the input contains one integer t ($1 \le t \le 500$) — the number of test cases.

Then the test cases follow. Each test case contains three integers n, m and k ($2 \le n \le 50$, $0 \le m \le n$, $2 \le k \le n$, k is a divisors of n).

Output

For each test case, print one integer — the maximum number of points a player can get for winning the game.

input
4 8 3 2 4 2 4 9 6 3 42 0 7

output
3 0 1 0

Test cases of the example are described in the statement.

B. New Theatre Square

2 seconds, 256 megabytes

You might have remembered Theatre square from the problem 1A. Now it's finally getting repaved.

The square still has a rectangular shape of $n \times m$ meters. However, the picture is about to get more complicated now. Let $a_{i,j}$ be the j-th square in the i-th row of the pavement.

You are given the picture of the squares:

- if $a_{i,j} =$ "*", then the j-th square in the i-th row should be **black**;
- if $a_{i,j} =$ ".", then the j-th square in the i-th row should be **white**.

The black squares are paved already. You have to pave the white squares. There are two options for pavement tiles:

- 1×1 tiles — each tile costs x burles and covers exactly 1 square;
- 1×2 tiles — each tile costs y burles and covers exactly 2 adjacent squares of the **same row**. **Note that you are not allowed to rotate these tiles or cut them into 1×1 tiles.**

You should cover all the white squares, no two tiles should overlap and no black squares should be covered by tiles.

What is the smallest total price of the tiles needed to cover all the white squares?

Input

The first line contains a single integer t ($1 \le t \le 500$) — the number of testcases. Then the description of t testcases follow.

The first line of each testcase contains four integers n, m, x and y ($1 \le n \le 100$; $1 \le m \le 1000$; $1 \le x, y \le 1000$) — the size of the Theatre square, the price of the 1×1 tile and the price of the 1×2 tile.

Each of the next n lines contains m characters. The j-th character in the i-th line is $a_{i,j}$. If $a_{i,j} =$ "*", then the j-th square in the i-th row should be black, and if $a_{i,j} =$ ".", then the j-th square in the i-th row should be white.

It's guaranteed that the sum of $n \times m$ over all testcases doesn't exceed 10^5.

Output

For each testcase print a single integer — the smallest total price of the tiles needed to cover all the white squares in burles.

input
4 1 1 10 1 . 1 2 10 1 .. 2 1 10 1 . . 3 3 3 7 ..* *.. .*.

output
10 1 20 18

In the first testcase you are required to use a single 1×1 tile, even though 1×2 tile is cheaper. So the total price is 10 burles.

In the second testcase you can either use two 1×1 tiles and spend 20 burles or use a single 1×2 tile and spend 1 burle. The second option is cheaper, thus the answer is 1.

The third testcase shows that you can't rotate 1×2 tiles. You still have to use two 1×1 tiles for the total price of 20.

In the fourth testcase the cheapest way is to use 1×1 tiles everywhere. The total cost is $6 \cdot 3 = 18$.

C. Mixing Water

2 seconds, 256 megabytes

There are two infinite sources of water:

- hot water of temperature h;
- cold water of temperature c ($c < h$).

You perform the following procedure of alternating moves:

1. take **one** cup of the **hot** water and pour it into an infinitely deep barrel;
2. take **one** cup of the **cold** water and pour it into an infinitely deep barrel;
3. take **one** cup of the **hot** water . . .
4. and so on . . .

Note that you always start with the cup of hot water.

The barrel is initially empty. You have to pour **at least one cup** into the barrel. The water temperature in the barrel is an average of the temperatures of the poured cups.

You want to achieve a temperature as close as possible to t. So if the temperature in the barrel is t_b, then the **absolute difference** of t_b and t ($|t_b - t|$) should be as small as possible.

How many cups should you pour into the barrel, so that the temperature in it is as close as possible to t? If there are multiple answers with the minimum absolute difference, then print the smallest of them.

Input

The first line contains a single integer T ($1 \le T \le 3 \cdot 10^4$) — the number of testcases.

Each of the next T lines contains three integers h, c and t ($1 \le c < h \le 10^6$; $c \le t \le h$) — the temperature of the hot water, the temperature of the cold water and the desired temperature in the barrel.

Output

For each testcase print a single positive integer — the minimum number of cups required to be poured into the barrel to achieve the closest temperature to t.

input
3 30 10 20 41 15 30 18 13 18
output
2 7 1

In the first testcase the temperature after 2 poured cups: 1 hot and 1 cold is exactly 20. So that is the closest we can achieve.

In the second testcase the temperature after 7 poured cups: 4 hot and 3 cold is about 29.857. Pouring more water won't get us closer to t than that.

In the third testcase the temperature after 1 poured cup: 1 hot is 18. That's exactly equal to t.

D. Yet Another Yet Another Task

1.5 seconds, 512 megabytes

Alice and Bob are playing yet another card game. This time the rules are the following. There are n cards lying in a row in front of them. The i-th card has value a_i.

First, Alice chooses a non-empty consecutive segment of cards $[l; r]$ ($l \le r$). After that Bob removes a single card j from that segment ($l \le j \le r$). The score of the game is the total value of the remaining cards on the segment ($a_l + a_{l+1} + \dots + a_{j-1} + a_{j+1} + \dots + a_{r-1} + a_r$). In particular, if Alice chooses a segment with just one element, then the score after Bob removes the only card is 0.

Alice wants to make the score as big as possible. Bob takes such a card that the score is as small as possible.

What segment should Alice choose so that the score is maximum possible? Output the maximum score.

Input

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

The second line contains n integers $a_1, a_2, \dots, a_n$ ($-30 \le a_i \le 30$) — the values on the cards.

Output

Print a single integer — the final score of the game.

input
5 5 -2 10 -1 4
output
6

input
8 5 2 5 3 -30 -30 6 9
output
10

input
3 -10 6 -15
output
0

In the first example Alice chooses a segment $[1; 5]$ — the entire row of cards. Bob removes card 3 with the value 10 from the segment. Thus, the final score is $5 + (-2) + (-1) + 4 = 6$.

In the second example Alice chooses a segment $[1; 4]$, so that Bob removes either card 1 or 3 with the value 5, making the answer $5 + 2 + 3 = 10$.

In the third example Alice can choose any of the segments of length 1: $[1; 1]$, $[2; 2]$ or $[3; 3]$. Bob removes the only card, so the score is 0. If Alice chooses some other segment then the answer will be less than 0.

E. Modular Stability

2 seconds, 512 megabytes

We define $x \bmod y$ as the remainder of division of x by y ($\%$ operator in C++ or Java, `mod` operator in Pascal).

Let's call an array of positive integers $[a_1, a_2, \ldots, a_k]$ *stable* if for every permutation p of integers from 1 to k, and for every non-negative integer x, the following condition is met:

$$(((x \bmod a_1) \bmod a_2) \ldots \bmod a_{k-1}) \bmod a_k = (((x \bmod a_{p_1}) \bmod a_{p_2}) \ldots \bmod a_{p_{k-1}}) \bmod a_{p_k}$$

That is, for each non-negative integer x, the value of $(((x \bmod a_1) \bmod a_2) \ldots \bmod a_{k-1}) \bmod a_k$ does not change if we reorder the elements of the array a.

For two given integers n and k, calculate the number of *stable* arrays $[a_1, a_2, \ldots, a_k]$ such that $1 \le a_1 < a_2 < \cdots < a_k \le n$.

Input

The only line contains two integers n and k ($1 \le n, k \le 5 \cdot 10^5$).

Output

Print one integer — the number of *stable* arrays $[a_1, a_2, \ldots, a_k]$ such that $1 \le a_1 < a_2 < \cdots < a_k \le n$. Since the answer may be large, print it modulo 998244353.

input
7 3
output
16

input
3 7
output
0

input
1337 42
output
95147305

input
1 1
output
1

input
500000 1
output
500000

F. RC Kaboom Show

6 seconds, 256 megabytes

You know, it's hard to conduct a show with lots of participants and spectators at the same place nowadays. Still, you are not giving up on your dream to make a car crash showcase! You decided to replace the real cars with remote controlled ones, call the event "Remote Control Kaboom Show" and stream everything online.

For the preparation you arranged an arena — an infinite 2D-field. You also bought n remote controlled cars and set them up on the arena. Unfortunately, the cars you bought can only go forward without turning left, right or around. So you additionally put the cars in the direction you want them to go.

To be formal, for each car i ($1 \le i \le n$) you chose its initial position (x_i, y_i) and a direction vector (dx_i, dy_i). Moreover, each car has a constant speed s_i units per second. So after car i is launched, it stars moving from (x_i, y_i) in the direction (dx_i, dy_i) with constant speed s_i.

The goal of the show is to create a car collision as fast as possible! You noted that launching every car at the beginning of the show often fails to produce any collisions at all. Thus, you plan to launch the i-th car at some moment t_i. **You haven't chosen t_i, that's yet to be decided.** Note that it's not necessary for t_i to be integer and t_i is allowed to be equal to t_j for any i, j.

The show starts at time 0. The show ends when two cars i and j ($i \ne j$) collide (i. e. come to the same coordinate at the same time). The duration of the show is the time between the start and the end.

What's the fastest crash you can arrange by choosing all t_i? If it's possible to arrange a crash then print the shortest possible duration of the show. Otherwise, report that it's impossible.

Input

The first line contains a single integer n ($1 \le n \le 25000$) — the number of cars.

Each of the next n lines contains five integers $x_i, y_i, dx_i, dy_i, s_i$ ($-10^3 \le x_i, y_i \le 10^3$; $1 \le |dx_i| \le 10^3$; $1 \le |dy_i| \le 10^3$; $1 \le s_i \le 10^3$) — the initial position of the i-th car, its direction vector and its speed, respectively.

It's guaranteed that all cars start at distinct positions (i. e. $(x_i, y_i) \ne (x_j, y_j)$ for $i \ne j$).

Output

Print the shortest possible duration of the show if it's possible to arrange a crash by choosing all t_i. Otherwise, print "`No show :(`".

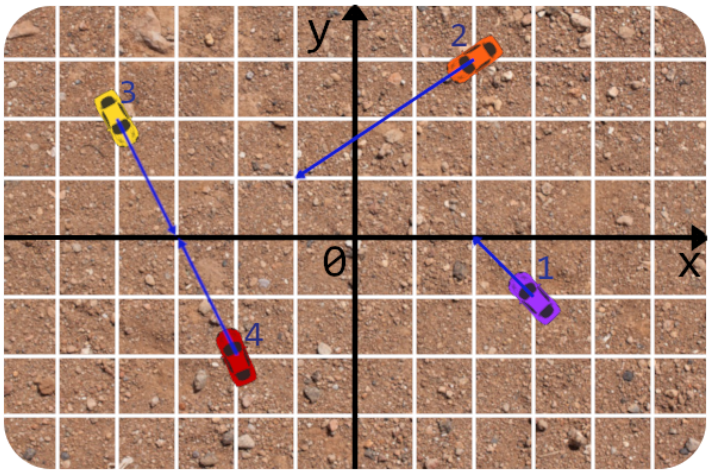
Your answer is considered correct if its absolute or relative error does not exceed 10^{-6}.

Formally, let your answer be a, and the jury's answer be b. Your answer is accepted if and only if $\frac{|a-b|}{\max(1, |b|)} \le 10^{-6}$.

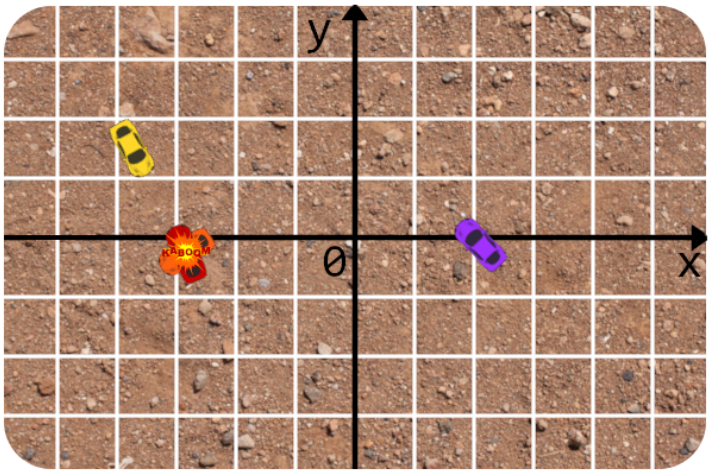
input
4 3 -1 -1 1 2 2 3 -3 -2 10 -4 2 1 -2 1 -2 -2 -1 2 4
output
0.585902082262898

input
2 -1 1 -1 1 200 1 1 1 5 200
output
No show :(

Here is the picture for the first example:



The fastest cars to crash are cars 2 and 4. Let's launch car 2 at 0, car 4 at about 0.096762 and cars 1 and 3 at arbitrary time. That way cars 2 and 4 will crash into each other at about 0.585902. So here's what it looks like at the moment of the collision:



Here's the picture for the second example: