Problem A. Survival Route

Input file:	stdin
Output file:	stdout
Time limit:	1 second
Memory limit:	256 megabytes

The surface of recently discovered planet i1c5l is a perfectly shaped sphere with radius R. Earth's aerospace forces sent a research module Waldemar-63 that was supposed to land on the planet's surface at point A. However, due to miscalculation, Waldemar-63 got to point B instead. Fortunately, Waldemar series are equipped with a train of wheels that will help in getting to the initial destination point. Nevertheless, power capacity is quite limited so it is important to get to point A by minimal distance run.

A major complication is that some point O on the surface of the planet radiates waves of unknown origin. The waves spread inside the radius r (the distance is measured on the surface). As the previous 62 missions show, any device that gets into the radiation area instantly breaks down. Thus, Waldemar-63's route should avoid the radiation area. Find the minimal distance that the device has to undertake from B to A.

Input

The first line contains integers X_A, Y_A ($-90 \le X_A \le 90$, $180 \le Y_A \le 180$) — coordinates (latitude and longitude in degrees) of point A. Second line — coordinates of point B. Third — three integers: coordinates of point O and the radiation radius r. In the fourth line — positive integer not exceeding 1000 — planet's radius R. It is guaranteed that points A and B are beyond the area of radiation.

Output

Output the distance determined. The answer is considered correct if the absolute or relative error is less than 10^{-4}.

Examples

stdin	stdout
55 49 55 129 90 0 1 2	1.510687

Problem B. Dispersed parentheses

Input file:	stdin
Output file:	stdout
Time limit:	1 second
Memory limit:	256 megabytes

The sequence of calculations in arithmetic expressions is usually set by a certain arrangement of parentheses. For example, $(3 \cdot (2 + 1)) \cdot (4 - 5)$. After deleting all the elements from the expression except parentheses remaining symbols form *a parentheses sequence* (())(). Let's assume that adding character «0» does not corrupt the sequence. Let's call such sequence a *disperse parentheses sequence*. Also this can be defined as follows:

- An empty line is a disperse parentheses sequence.
- If S and T — disperse parentheses sequences, then lines $0S, S0, (S)$ and ST are also disperse parentheses sequences.

The depth of disperse parentheses sequence is the maximum difference between the number of opening and closing parentheses in the sequence prefix. (The prefix of line S is the line, which can be obtained from S by deleting symbols from the tail of the line. For example, the prefixes of line «$ABCAB$» are lines «», «A», «AB», «ABC», «$ABCA$» and «$ABCAB$»). Thus, the depth of the sequence «(0)(0())0» equals two (prefix «(0)(0(» contains three openinig and one closing parentheses).

Calculate the number of possible disperse parentheses sequences n symbols long, that have a depth k.

Input

Single line contains space-separated integers n and k ($1 \le n \le 300$, $0 \le k \le n$).

Output

Output the number of possible disperse parentheses sequences n symbols long, that have a depth k modulo $(10^9 + 9)$.

Examples

stdin	stdout
3 0	1
3 1	3
3 2	0

Problem C. Chocolate triangles

Input file:	stdin
Output file:	stdout
Time limit:	4 seconds
Memory limit:	256 megabytes

"Tortoff" Inc. makes your favorite chocolate products. For your convenience we produce chocolate exclusively in the shape of convex polygon. You can break it into pieces along any non-intersecting diagonals. According to our expert surveys the tastiest pieces of chocolate are triangular. Any customer can help us improve the quality of our products. It is important for us to know the number of ways our exclusive chocolate can be broken into exactly k triangular parts.

Input

Single line contains integers n and k ($3 \le n \le 300$, $0 \le k \le n - 2$).

Output

Output the number of ways the n-polygon can be cut into exactly k triangular parts along non-intersecting diagonals inside the polygon. Output answer modulo $(10^9 + 9)$.

Examples

stdin	stdout
4 0	1
4 1	0
4 2	2

Problem D. LWDB

Input file:	stdin
Output file:	stdout
Time limit:	9 seconds
Memory limit:	256 megabytes

The Large Wood Database is created to securely store and paint any existing tree. Update for LWDB provides new functionality, so it is time to think over the graph theory. A weighed tree is stored in the LWDB. In the query language for LWDB Management System (LWDB MS) two types of queries are available:

1. «1 v d c» — paint all tree-vertices at the distance not exceeding d from the vertice v in color c. Initial color for any vertices is 0.
2. «2 v» — return the color of the vertice v.

It is required to prototype LWDB MS and respond to all user's queries.

Input

The first line contains an integer N ($1 \le N \le 10^5$) — the number of tree vertices. The following N-1 lines contain the description of branches, three numbers in each line a_i, b_i, w_i ($1 \le a_i, b_i \le N$, $a_i \ne b_i$, $1 \le w_i \le 10^4$), where i-th branch with weight w_i connects vertices a_i and b_i. The next line contains integer Q ($1 \le Q \le 10^5$) — number of queries. In each of Q following lines there are two types of queries:

1. Numbers 1, v, d, c ($1 \le v \le N$, $0 \le d \le 10^9$, $0 \le c \le 10^9$).
2. Numbers 2, v ($1 \le v \le N$).

Input numbers are integers.

Output

For each second type query output the color of requested vertice in a separate line.

Examples

stdin	stdout
5 1 2 30 1 3 50 3 4 70 3 5 60 8 1 3 72 6 2 5 1 4 60 5 2 3 2 2 1 2 144 7 2 4 2 5	6 6 0 5 7

Problem E. Pea-City

Input file:	stdin
Output file:	stdout
Time limit:	2 seconds
Memory limit:	256 megabytes

The Pea King was a wise ruler and a good strategist. He spent all his time dealing with the state affairs. As the kingdom grew, population increased. The new houses popped up everywhere like mushrooms after the rain. The kings of neighboring states were becoming envious and started to plan a war. In order to protect the capital from the invaders, the Pea King decided to build a new city wall around the capital borders that would lock all the buildings on the city outskirts inside of it as well.

For the aesthetic effect, namely to make the map on the wall in king's parlor look pretty, it was decided that the capital city should have rectangular borders. Keeping this in mind the total land area should remain as minimal as possible and all the N houses registered should stay inside the borders of the rectangular city.

Input

The first line contains single integer N — number of houses in the city ($3 \le N \le 80\,000$). The following N lines contain pairs of integers X, Y — Cartesian coordinates of the houses ($-25\,000 \le X, Y \le 25\,000$). It's guaranteed that there are at least three houses not on one line.

Output

Output the coordinates of the rectangular corner points in the counterclockwise order, each pair of coordinates on a separate line. The answer is considered correct if the absolute or relative error for rectangular area does not exceed 10^{-5} and all houses are strictly inside the city or at the distance not exceeding 10^{-5} from its borders.

Examples

stdin	stdout
3 0 0 2 2 2 0	0.000000 2.000000 0.000000 0.000000 2.000000 0.000000 2.000000 2.000000

Problem F. Beautiful sums

Input file:	stdin
Output file:	stdout
Time limit:	1 second
Memory limit:	256 megabytes

Beautiful sums are the sums of several consequent positive integers. For example, the sums $7 + 8$ and $4 + 5 + 6$ are beautiful, and the sum $3 + 5 + 7$ is not beautiful even though the value in all cases equals 15. (The sum of single summand 15 also considered beautiful.)

Given this, *the beauty index* of integer is the number of its representations as a beautiful sum. For example, the beauty index of number 15 equals 4 as 15 is represented by a beautiful sum in four ways: $15 = 7 + 8 = 4 + 5 + 6 = 1 + 2 + 3 + 4 + 5$.

One number is more beautiful than another if its beauty index is higher. If numbers have equal beauty indexes the smaller one is considered more beautiful. For example, 15 is the smallest integer having beauty index 4.

You have to find the smallest integer for given beauty index n.

Input

Single line contains an integer n ($1 \le n \le 10^5$).

Output

Output the smallest integer for given beauty index n modulo $(10^9 + 9)$.

Examples

stdin	stdout
3	9
4	15

Problem G. Nano alarm-clocks

Input file:	stdin
Output file:	stdout
Time limit:	1 second
Memory limit:	256 megabytes

An old watchmaker has n stopped nano alarm-clocks numbered with integers from 1 to n. Nano alarm-clocks count time in hours, and in one hour there are million minutes, each minute lasting a million seconds. In order to repair them all the watchmaker should synchronize the time on all nano alarm-clocks. In order to do this he moves clock hands a certain time forward (may be zero time). Let's name this time shift a transfer time.

Your task is to calculate the minimal total transfer time required for all nano alarm-clocks to show the same time.

Input

The first line contains a single integer n — the number of nano alarm-clocks ($2 \le n \le 10^5$). In each i-th of the next n lines the time h, m, s, shown on the i-th clock. Integers h, m and s show the number of hours, minutes and seconds respectively. ($0 \le h < 12$, $0 \le m < 10^6$, $0 \le s < 10^6$).

Output

Output three integers separated with spaces h, m and s — total minimal transfer time, where h, m and s — number of hours, minutes and seconds respectively ($0 \le m < 10^6$, $0 \le s < 10^6$).

Examples

stdin	stdout
2 10 0 0 3 0 0	5 0 0
3 11 999999 999999 0 0 0 11 999999 999999	0 0 2

Problem H. Lunch

Input file:	stdin
Output file:	stdout
Time limit:	1 second
Memory limit:	256 megabytes

The swamp looks like a narrow lane with length n covered by floating leaves sized 1, numbered from 1 to n with a fly sitting on the top of each. A little toad is sitting on one of the leaves instead of a fly. Its name is Kvait and it is about to have lunch. It can jump to the bordering leaf or jump it over to the next one in any direction. When landing it eats a fly. Kvait is already quite a big toad and the leaves are unstable so when it jumps away the leaf starts sinking.

In order to have lunch Kvait needs to eat all of the flies. It starts his journey from the leaf with number s and has to finish on the leaf with number f. Yet jumping to the bordering leaf takes more Kvait's energy than skipping a leaf over. It is necessary to plan the toad's movements to get lunch with minimal energy spent.

Input

Single line contains three integers n, s, f ($2 \le n \le 10\,000$, $1 \le s, f \le n$) — the number of leaves, number of a starting leaf and the number of the finish leaf respectively.

Output

Output the minimal number of jumps to the bordering leaves required for the toad to have lunch. If there is no way to eat up, output a single number -1.

Examples

stdin	stdout
4 1 2	1

Problem I. Accounting Numeral System

Input file:	stdin
Output file:	stdout
Time limit:	1 second
Memory limit:	256 megabytes

The latest Accounting Numeral System is the top accounting system in the whole world. Its creator, Dr. Ceizenpok, is the best expert of the respective authority. Any positive integer n in this system based m is represented as a sum of m parts:

$$n = C_{x_m}^m + C_{x_{m-1}}^{m-1} + C_{x_{m-2}}^{m-2} + \ldots + C_{x_1}^1,$$

while $x_1, x_2, \ldots, x_m$ — are such integers that $0 \le x_1 < x_2 < \ldots < x_m$. Numbers $C_k^m = \frac{k!}{m!\,(k-m)!}$ our experts call accounting indexes. Each number n in this system is recorded as $n = \overline{(x_m)\ldots(x_2)(x_1)}$, and it is considered that $0! = 1$ and $C_k^m = 0$, if $m > k$. For example, number 9 in the accounting system based 3 is recorded as $(\mathbf{4})(\mathbf{3})(\mathbf{2})$, because $9 = C_{\mathbf{4}}^3 + C_{\mathbf{3}}^2 + C_{\mathbf{2}}^1$, and number 1 in this system based 2 looks like:$(\mathbf{2})(\mathbf{0})$, because $1 = C_{\mathbf{2}}^2 + C_{\mathbf{0}}^1$.

You have to find a representation of an integer n in the accounting numeral system based m.

Input

Single line contains two integers n and m ($1 \le n \le 10^{16}$, $2 \le m \le 1\,000$).

Output

Single line should contain a sequence of m space-separated integers $x_m, \ldots, x_2, x_1$, that form a number designation n in the accounting numeral system. Number x_m is the leftmost digit in the number designation n, and x_1 — its rightmost one.

Examples

stdin	stdout
9 3	4 3 2
5 2	3 2

Problem J. Ceizenpok's formula

Input file:	stdin
Output file:	stdout
Time limit:	1 second
Memory limit:	256 megabytes

Dr. Ceizenpok from planet i1c5l became famous across the whole Universe thanks to his recent discovery — the Ceizenpok's formula. This formula has only three arguments: n, k and m, and its value is a number of k-combinations of a set of n modulo m.

While the whole Universe is trying to guess what the formula is useful for, we need to automate its calculation.

Input

Single line contains three integers n, k, m, separated with spaces ($1 \le n \le 10^{18}$, $0 \le k \le n$, $2 \le m \le 1\,000\,000$).

Output

Write the formula value for given arguments n, k, m.

Examples

stdin	stdout
2 1 3	2
4 2 5	1

Problem K. Dividing an orange

Input file:	stdin
Output file:	stdout
Time limit:	1 second
Memory limit:	256 megabytes

On a far-away planet i1c5l a community of n people harvested k oranges. Now they have to divide the harvest among themselves.

This community is ruled by a democratic principle based on the rank hierarchy. Thus, the harvest is divided the following way: each person gets a rank from 1 to n. Then, a person ranked 1 announce his decision: who and how many oranges get. Afterwards all n people vote «for» or «against». If at least half of the people vote «for», then the decision is accepted. Otherwise the person ranked 1 is ostracized from the community and it is turn to person ranked 2 to announce a decision, and the procedure is repeated.

Each person wish to get the best for himself trying to get as many oranges as possible. Between the cases with equal amount of oranges earned he will prefer the one with less people in community left. If a person is ostracized from the community it is considered that he got a negative amount of oranges. If several optimal solutions exist a person can choose any. Each person knows that other people also try to find the optimal solution being guided by the same principles.

However, one of the community members has m wildcards that can give him predefined ranks. The task is to find out the minimal and the maximum number of oranges that can be obtained for each wildcard rank.

Input

The first line contains integers n, k and m ($1 \le n, k \le 10^9$, $1 \le m \le 10^5$) — the amount of people, oranges and wildcards respectively.

The second line contains m integers a_1, a_2, ..., a_m ($1 \le ai \le n$), where a_i — the rank given by i-th wildcard.

Output

For each of a_i write a minimal and a maximum amount of oranges on a separate line, which will be obtained by the wildcard owner or «-1 -1» (without quote), if he or she will be ostracized.

Examples

stdin	stdout
2 10 2 1 2	10 10 0 0
3 1 2 1 3	0 0 1 1

Note

In the first example the person with the first rank takes all the oranges and votes «for».

In the second example the person ranked 1 has to give the orange to the person of third rank. If the first one takes one orange then the rest of the members will vote «against». If the first gives the orange to the second then the second will also be «against» as he knows that by ostracizing the first they will also get an orange but there will be less people left in the community. Because the third one will also be «against» this is not an option for the first one.

Problem L. The Pool for Lucky Ones

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 megabytes

A new swimming pool has been built in Kazan for the forthcoming Water Sports World Championship. The pool has N lanes. Some of the lanes are already occupied by swimmers. Tatar scientists have divided the lanes into the lucky and unlucky ones. The unlucky lanes are those with the maximum amount of swimmers. That is, there is no other lane where there would be more swimmers than on unlucky one. The unlucky lanes make swimmers unhappy. The rest of the lanes are considered to be lucky. The lucky lanes make people happy. The scientists took a decision to make more people happy. In order to do this they had an agreement with the pool manager saying they can move a single person from any lane to the one neighboring if it was necessary. The swimmer from the first lane can only be moved to the second lane, and the swimmer from the last lane — to the one before last.

Input

The first line contains an integer N — the amount of lanes in the pool ($3 \le N \le 10^5$). The second line contains N integers p_i separated with spaces, describing distribution of swimmers between the lanes where p_i is the amount of swimmers on i-th lane ($0 \le p_i \le 10^5$).

Output

Output a single number — minimal possible number of unhappy swimmers.

Examples

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