Problem A. Academic Distance

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Starting from the second semester, Mr. Eto will take classes at Kyoto University. Mr. Eto is not accustomed to the structure of the university because in the first semester he had only online lectures.

There are N classes today. The schedule contains the coordinates of N classrooms in the order in which they have to be visited. The coordinates of the i-th classroom are (x_i, y_i). Assuming Mr. Eto starts the day in the first classroom, and ends in the N-th classroom, calculate the total distance he has to travel.

In Kyoto University campus, the distance traveled from the coordinates (a, b) to the coordinates (c, d) is equal to $|a - c| + |b - d|$.

Input

The first line of the input contains one integer N ($1 \le N \le 100$), the number of classrooms in today's schedule. Then N lines follow, i-th of them containing integer coordinates x_i and y_i of the i-th classroom in the schedule ($-100 \le x_i, y_i \le 100$).

Output

Print one integer: the total distance traveled by Mr. Eto by the end of the day.

Examples

standard input	standard output
3 1 2 2 3 4 6	7
1 0 0	0
4 -2 3 1 4 5 2 4 -2	15

Problem B. Bunch of Paper

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

There are N sheets of paper, enumerated by sequential integers from 1 to N. Each sheet has K integers written on it, so i-th sheet contains the integers $v_{i,1}, v_{i,2}, \ldots, v_{i,K}$.

Then we choose one integer from each sheet and create the sequence a_i, where i-th integer is chosen from i-th sheet of paper. There are K^N ways to make such a sequence. How many of them are non-decreasing? A sequence is non-decreasing if $a_i \le a_{i+1}$ for all $1 \le i \le N - 1$.

The answer may be too large, so print it modulo $10^9 + 7$.

Input

The first line of the input contains two integers N and K ($1 \le N \le 100$, $1 \le K \le 10^4$). The i-th of the following N lines contains K integers $v_{i,1}, v_{i,2}, \ldots, v_{i,K}$ ($1 \le v_{i,1} < v_{i,2} < \ldots < v_{i,K} \le 10^9$).

Output

Print the number of non-decreasing sequences, modulo $10^9 + 7$.

Examples

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

Problem C. Character Grid

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

This is an output-only problem.

Your task is to build a square grid with side length $N \geq 13$ and fill it with lowercase English letters such that the following property is held.

Let us denote the character at i-th row and j-th column as $c_{i,j}$.

Consider $N^2 \cdot (N-1)/2$ strings of the form $A_{i,j,p} = c_{i,j}c_{i,j+1}\ldots c_{i,j+p}$ for every $1 \leq i \leq N$ and every $1 \leq j, p \leq N-1$ such that $j+p \leq N$.

Consider also $N^2 \cdot (N-1)/2$ strings of the form $B_{i,j,p} = c_{i,j}c_{i+1,j}\ldots c_{i+p,j}$ for every $1 \leq j \leq N$ and every $1 \leq i, p \leq N-1$ such that $i+p \leq N$.

All those $N^2 \cdot (N-1)$ strings have to be pairwise distinct.

Input

There is no input.

Output

Print the answer in the following format: first print the N ($13 \leq N \leq 100$). Then print the square grid as N lines; i-th line shall contain one string of N characters, representing the i-th row of the grid.

If there are several correct solutions, any one of them will be accepted.

Example

standard input	standard output
	4 petr ozav odsk camp

Note

For the answer from the sample, the grid property is held, but the grid size is too small to be accepted as a solution.

Problem D. Distribute the Bars

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

There are N metal bars. The weight of i-th metal bar is $2i - 1$.

Your task is to distribute those metal bars into two or more groups such that the sum of weights of metal bars in each group will be the same, or determine that it is impossible. Note that each metal bar must go to exactly one group, and that it is not allowed to cut the metal bars.

Input

The input contains one integer N ($2 \le N \le 10^5$).

Output

If there is no way to distribute the metal bars into two or more equally weighted groups, print one line containing the integer -1.

Otherwise, on the first line, print the number of groups G ($2 \le G \le N$). Then print G lines, one for each group. The i-th of these lines must start with the the integer K_i, the number of metal bars in the i-th group. Then print K_i integers: the weights of the metal bars in the group. Each metal bar must be assigned to exactly one group, and the sums of weights of the metal bars in all groups must be the same.

If there is more than one solution, print any one of them.

Examples

standard input	standard output
4	2 2 1 7 2 3 5
2	-1
3	-1

Problem E. Efficient Partitioning

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Let a *partition* of $[0, N)$ be an integer sequence $S = (s_0, \ldots, s_r)$ that satisfies the following three conditions:

- $s_0 = 0$,
- $s_r = N$,
- $s_i < s_{i+1}$ $(0 \le i < r)$.

That is, for each i, $[s_i, s_i + 1)$ represents a continuous interval, and $[0, N)$ is the union of these r intervals.

Given are three sequences of length N consisting of integers between -10^9 and 10^9: A $(a_0, \ldots, a_{N-1})$, B $(b_0, \ldots, b_{N-1})$, C $(c_0, \ldots, c_{N-1})$.

Let the score of partition S be $f(S)$ defined as follows:

$$f(S) = \min_{0 \le i < r} \left\{ b_{s_i} + c_{s_{i+1}-1} + \sum_{s_i \le j < s_{i+1}} a_j \right\}$$

Find the maximum value of f over all possible partitions S.

Input

The first line of input contains one integer N $(1 \le N \le 2 \cdot 10^5)$. The second line contains n integers: i-th of them denotes a_i. The third and fourth lines describe the sequences b and c in the same format $(-10^9 \le a_i, b_i, c_i \le 10^9)$.

Output

Print one integer: the maximum score over all possible partitions.

Examples

standard input	standard output
2 1 -1 -1 4 1 -2	1
1 1000000000 1000000000 1000000000	3000000000

Problem F. Find the MST for Grid

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Consider a grid graph: the vertices are lined up into a grid of H rows by W columns. Let us denote the vertex in the i-th row and j-th column as (i, j).

To define the weights of the graph edges, we will consider four non-decreasing sequences, A, B, C, and D, consisting of $H - 1$, W, H, and $W - 1$ positive integers, respectively:

- there is a bidirectional edge connecting vertices (i, j) and $(i + 1, j)$ of weight $A_i + B_j$ for all i and j such that $1 \le i \le H - 1$ and $1 \le j \le W$;
- there is a bidirectional edge connecting vertices (i, j) and $(i, j + 1)$ of weight $C_i + D_j$ for all i and j such that $1 \le i \le H$ and $1 \le j \le W - 1$;
- the graph contains no other edges.

Find the total weight of the edges in the minimal spanning tree of this graph.

Input

The first line of input contains two positive integers H and W ($2 \le H, W \le 10^5$).

The second line contains $H - 1$ integers A_i: the elements of the sequence A.

The third line contains W integers B_i: the elements of the sequence B.

The fourth line contains H integers C_i: the elements of the sequence C.

The fifth line contains $W - 1$ integers D_i: the elements of the sequence D.

It is guaranteed that $A_{i-1} \le A_i$, $B_{i-1} \le B_i$, $C_{i-1} \le C_i$, and $D_{i-1} \le D_i$ for $i > 1$, and additionally, $1 \le A_i, B_i, C_i, D_i \le 10^6$.

Output

Print the total weight of the edges in the minimal spanning tree of the given graph. Note that the answer may not fit into a 32-bit integer.

Examples

standard input	standard output
2 3 1 1 3 6 1 4 1 2	17
4 3 1 13 15 3 6 11 3 6 6 11 9 17	173

Problem G. Generate the Sequences

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Consider S, the sequence of integer sequences. Initially, $S_0 = (1)$. After that, we construct $S_1, S_2, \ldots, S_n$ as follows.

Let $|S_i|$ be the length of the sequence S_i, and $s_{i,j}$ be the j-th element of S_i. Then S_{i+1} will have length $|S_i| + 1$ and can be obtained from $|S_i|$ using one of the following two operations:

- Write 1 or the given integer m as the element with number $|S_i| + 1$ of the new sequence.
- Select an index j ($1 \le j < |S_i|$), choose integer x such that $s_{i,j} < x < s_{i,j+1}$ or $s_{i,j} > x > s_{i,j+1}$, and place it between $s_{i,j}$ and $s_{i,j+1}$, shifting the right part's indices by 1.

Given n and m, find the number of different ordered sets of sequences $S_1 \ldots S_n$. Two sets are considered different if, at least for one i from 1 to n, the sequences S_i in those sets differ. As the answer may be too large, print it modulo 998 244 353.

Input

The input consists of one line containing two integers n and m ($1 \le n \le 3000$, $2 \le m \le 10^8$).

Output

Print the number of different sequences S modulo 998 244 353.

Examples

standard input	standard output
2 3	5
1024 52689658	654836147

Note

Here are the possible sequences in the first example:

- $S_1 = (1, 3)$ (first operation), then $S_2 = (1, 2, 3)$ (second operation);
- $S_1 = (1, 1)$ (first operation), then $S_2 = (1, 1, 3)$ (first operation);
- $S_1 = (1, 1)$ (first operation), then $S_2 = (1, 1, 1)$ (first operation);
- $S_1 = (1, 3)$ (first operation), then $S_2 = (1, 3, 3)$ (first operation);
- $S_1 = (1, 3)$ (first operation), then $S_2 = (1, 3, 1)$ (first operation).

Therefore, the answer is 5.

Problem H. How to Move the Beans

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

There is a grid consisting of H rows and W columns. The grid is cylindrical: it has left and right sides glued together, so columns 1 and W are neighbors.

Some of the grid squares contain dishes, and beans are placed on some of those dishes such that, initially, each dish contains no more than one bean. Later in the game, a dish is allowed to contain any number of beans.

Alice and Bob are playing a game on this grid, moving in turns. Alice moves first. On each turn, the player can pick any bean, denote its current row and column by (r, c), and move it according to the following rules:

- A bean can be moved only to a square where a dish is placed.
- A bean can not be moved to a square where this particular bean was before (all the beans are distinguishable).
- From (r, c), a bean can be moved either one square down (to $(r + 1, c)$, possible only when $r < H$), one square to the right (to $(r, c + 1)$ if $c < W$ or to $(r, 1)$ if $c = W$), or one square to the left (to $(r, c - 1)$ if $c > 1$ or to (r, W) if $c = 1$).

The player who can not move any bean on their turn loses the game.

Determine who will win if both players are playing optimally.

Input

The first line of the input contains two integers H and W ($1 \le H, W \le 1000$).

Then the initial grid description follows, consisting of H lines, each containing a string of length W. The j-th character of i-th of these lines is either '`#`' when there is no dish at square (i, j), '`.`' when there is an empty dish in that square, or '`B`' when there is a dish with exactly one bean in that square. It is **not guaranteed** that the grid contains characters of all three types (for example, a grid without beans is valid).

Output

If Alice wins the game when both players are playing optimally, print "`Alice`". Otherwise, print "`Bob`".

Examples

standard input	standard output
2 3 B.# #..	Alice
1 1 B	Bob
1 3 B#.	Alice

Note

In the first example, the only bean is initially at $(1, 1)$. Alice moves it to $(1, 2)$. Bob's only one move is to $(2, 2)$, then Alice moves the bean to $(2, 3)$ and Bob has no moves left, so Alice is the winner.

Problem I. Interesting Coloring

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Given is an undirected simple connected graph, consisting of N vertices and M edges.

The vertices of this graph are enumerated by sequential integers from 1 to N, and the edges are enumerated by sequential integers from 1 to M, respectively. Edge i connects vertex u_i and vertex v_i.

The following special property holds for this graph: for every edge i ($1 \le i \le M$), there exists a path connecting u_i and v_i that does not contain this edge. We will call such path a *bypass path* of edge i.

There may be more than one bypass path for the same edge.

We will color the edges by colors enumerated by sequential integers from 1 to M, assigning exactly one color to every edge. Some colors may be left unused, others may be used more than once.

The coloring of the edges is called *interesting* if the following properties hold:

- If two edges have a common vertex, their colors are different.
- For every edge, there exists a *special* bypass path: a bypass path containing the edges colored with no more than 8 different colors.

Your task is to find an interesting coloring and, for each of the M edges, print any set of colors that can be used to build a special bypass path for that edge.

It can be shown that, under the constraints above, there exists at least one interesting coloring.

Input

The first line of input contains two integers N and M ($3 \le N \le 5555$, $3 \le M \le \min(N(N-1)/2, 9999)$).

The i-th of the M following lines describes the i-th edge and contains two integers u_i and v_i ($1 \le u_i < v_i \le N$).

You may assume that each pair (u, v) appears in the list at most once, that the given graph is connected and that, after removal of any edge (u, v), there still exists a bypass path connecting u and v.

Output

Print any interesting coloring in the following format.

On the first line, print M integers. The i-th of these integers, C_i, must be the color of the i-th edge ($1 \le C_i \le M$).

Then print M lines. The i-th of these lines describes the color set of the special bypass path for edge i. This line must start with the integer k_i ($1 \le k_i \le 8$): the number of the colors in the list. It must be followed by k_i pairwise distinct integers between 1 and M: the list of colors. The colors can be printed in any order. There must exist a special bypass path between u_i and v_i which does not use any colors except the colors in the list. Note that this means the list of colors does not have to be the minimal possible, and there can even be a path that uses only a part of the list: the checking program only makes sure that the listed colors are sufficient.

Example

standard input	standard output
10 11 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9 10 1 10 1 4	1 2 3 4 5 6 7 8 9 10 5 3 2 3 5 3 1 3 5 3 1 2 5 6 5 6 7 8 9 10 7 4 5 6 7 8 9 10 6 4 5 7 8 9 10 6 4 5 6 8 9 10 6 4 5 6 7 9 10 6 4 5 6 7 8 10 8 4 5 6 7 8 9 1 2 3 1 2 3

Note

In the example, there are two bypass paths for the first edge.

The longer one contains 9 colors (from 2 to 10), so it is not special.

The shorter one consists of the edges 2, 3, and 11 (colors 2, 3, and 5), so it is special.

Problem J. Joy with Permutations

Input file:	*standard input*
Output file:	*standard output*
Time limit:	5 seconds
Memory limit:	512 mebibytes

This is an interactive problem.

Alice secretly invents a permutation of first N integers $a_1, a_2, \ldots a_N$ and tells N to Bob.

Bob asks questions to identify this permutation.

He may ask questions of two types:

- Type 1, formatted as "`? 1` i j k": Bob chooses three different integers i, j, k ($1 \le i, j, k \le N$), Alice looks at the three integers a_i, a_j, and a_k, and tells Bob the **value** of their median (the one that is neither minimum nor maximum).
- Type 2, formatted as "`? 2` i j": Bob chooses two different integers i, j ($1 \le i, j \le N$), and Alice answers i if $a_i < a_j$, or j otherwise.

The game seems to be too easy for Bob, so Alice invented new rules. First, Bob may ask only $2N$ questions of type 1 and only 2 questions of type 2. Second, Alice may change the permutation freely as long as it is consistent with all answers that were given before.

Help Bob to win and write the program that identifies the permutation.

Interaction Protocol

First, the jury program tells you one integer N on a separate line ($4 \le N \le 60\,000$).

Then you may ask questions.

A question of type 1 is a line formatted as "`? 1` i j k" ($1 \le i, j, k \le N$; i, j, and k are pairwise distinct). The jury program then prints a line with a single integer: the median of the values a_i, a_j, and a_k. You may ask a question of this type no more than $2N$ times.

A question of type 2 is a line formatted as "`? 2` i j" ($1 \le i, j \le N$; $i \ne j$). The jury program then prints a line with a single integer: i if $a_i < a_j$, or j if $a_i > a_j$. You may ask a question of this type no more than twice.

When the permutation is uniquely determined, print the answer on a line formatted as "`!` a_1 a_2 $\ldots$ a_N".

Note that interaction is **adaptive**: the jury program may change the permutation anytime as long as it is consistent with past answers. In particular, if you are trying to guess the answer when it is not yet uniquely determined, the jury program may immediately pick another one and say you were wrong.

Don't forget to print the newline character and flush the output buffer after printing a question or an answer, otherwise, you may get the "`Idleness Limit Exceeded`" error.

Example

standard input	standard output
5 4 4 2	? 1 1 2 3 ? 2 2 4 ? 1 1 5 4 ! 3 5 4 1 2

Problem K. Kingdoms and Quarantine

Input file:	*standard input*
Output file:	*standard output*
Time limit:	8 seconds
Memory limit:	512 mebibytes

There are two kingdoms A (with N_1 cities) and B (with N_2 cities), and M bidirectional roads, each connecting a city from A and a city from B, such that there is no more than one road connecting any pair of cities.

The cities in the kingdom A are enumerated from 1 to N_1, and the cities in the kingdom B are enumerated from $N_1 + 1$ to $N_1 + N_2$. The roads are enumerated from 1 to M; the road i connects two cities a_i and b_i, where a_i and b_i satisfy $1 \le a_i \le N_1$ and $N_1 + 1 \le b_i \le N_1 + N_2$.

Once upon a time, a dangerous virus appeared in one kingdom, so the Kings decided to close some roads.

Let D_j be the initial number of roads connecting the city j with other cities, and d_j be the number of currently active (not closed) roads connecting the city j with other cities.

The road x can be closed if and only if following conditions are met **before** closing the road:

- It was not closed before.
- The numbers d_{a_x} and D_{b_x} must have the same parity (both even or both odd).
- The numbers d_{b_x} and D_{a_x} must have the same parity (both even or both odd).

Find the maximum number of roads that can be closed, and then find a sequence of road closing operations such that this maximum is achieved.

Input

The first line of input contains three integers, N_1, N_2, and M: the number of cities in kingdom A, the number of cities in kingdom B, and the number of roads, respectively ($1 \le N_1, N_2, M \le 3000$, $1 \le M \le N_1 \cdot N_2$).

The i-th of the following M lines describes the road i and contains two integers a_i and b_i ($1 \le a_i \le N_1$, $N_1 + 1 \le b_i \le N_1 + N_2$): the numbers of cities connected by that road. You may assume that, for different i and j, $a_i \ne a_j$ or $b_i \ne b_j$.

Output

On the first line, print the integer K: the maximum number of roads that can be closed. On the second line, print K integers r_i ($1 \le r_i \le M$): the numbers of roads to be closed, in the order of closing them.

If there are several optimal answers, print any one of them.

Examples

standard input	standard output
2 3 5 1 3 1 4 1 5 2 4 2 5	3 1 4 2
1 2 2 1 2 1 3	0
4 3 7 1 5 2 5 2 6 2 7 3 6 4 5 4 7	5 1 7 6 2 4

Note

In the first example, $D_1 = 3$, $D_2 = 2$, $D_3 = 1$, $D_4 = 2$, $D_5 = 2$.

Initially, $d_1 = 3$, $d_2 = 2$, $d_3 = 1$, $d_4 = 2$, $d_5 = 2$, so we can close the following roads:

- Road 1 connecting city 1 and city 3.
- Road 4 connecting city 2 and city 4.
- Road 5 connecting city 2 and city 5.

Let us close road 1, then

$d_1 = 2$, $d_2 = 2$, $d_3 = 0$, $d_4 = 2$, $d_5 = 2$.

After that, the roads that can be closed are the following:

- Road 4 connecting city 2 and city 4.
- Road 5 connecting city 2 and city 5.

Let us close road 4, then

$d_1 = 2$, $d_2 = 1$, $d_3 = 0$, $d_4 = 1$, $d_5 = 2$.

Now, we can close only road 2, connecting city 1 and city 4.

After that, $d_1 = 1$, $d_2 = 1$, $d_3 = 0$, $d_4 = 0$, $d_5 = 2$.

It can be shown that it is impossible to close more than three roads, so the answer is 3.

Problem L. Lazy Judge

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	512 mebibytes

This is an interactive problem.

The judges are working on the strategy for the jury program for the modified version of the problem **J** from the current contest.

In that problem, Alice secretly invents a permutation of first N integers $a_1, a_2, \ldots a_N$ and tells N to Bob.

Bob asks some questions to identify this permutation. Alice may change the permutation in the process as long as it is consistent with her previous answers.

The judges are planning to create an AliceBot that will do the following.

There are two phases: the question phase and the answer phase.

In the question phase, the judge tells to AliceBot an integer N. Then AliceBot has to answer some questions the judge asks about the permutation.

In the subsequent answer phase, AliceBot must compose two different permutations $a_1, \ldots, a_N$ and $b_1, \ldots, b_N$ that are consistent with the answers from the previous phase.

The judge who asks questions has an initial patience $P = 2N$. Each time the judge asks a question, the judge's patience decreases.

There are three types of questions the judge can ask:

- Type 1, formatted as "`? 1` i j k": the judge chooses three different integers i, j, k ($1 \le i, j, k \le N$), AliceBot looks at the three integers a_i, a_j, and a_k, and tells Bob the **value** of their median (the one that is neither minimum nor maximum). Each such question decreases the judge's patience by 2.
- Type 2, formatted as "`? 2` i j": the judge chooses two different integers i, j ($1 \le i, j \le N$), and AliceBot answers i if $a_i < a_j$, or j otherwise. Each such question decreases the judge's patience by 2.
- Type 3, formatted as "`? 3` i j": the judge chooses two different integers i, j ($1 \le i, j \le N$), and AliceBot tells the minimum **value** among a_i and a_j. Each such question decreases the judge's patience by 1.

You may assume that judge's patience will be **strictly greater** than 2 after asking a question. When the judge decides that they asked enough questions, the command "`!`" is sent to the AliceBot, switching it to the answer phase.

In the answer phase, AliceBot tells the judge two permutations $a_1, \ldots, a_N$ and $b_1, \ldots, b_N$. These two permutations must be consistent with all the answers given in the question phase, and the number of positions i such that $a_i \ne b_i$ has to be at least $\lceil p/2 \rceil$, where p is the judge's patience at the end of the question phase.

Because the judge is too lazy, you are asked to implement the AliceBot. It can be shown that the problem is solvable for every possible N from the constraints.

Interaction Protocol

First, the jury program tells you one integer N on a separate line ($4 \le N \le 50\,000$).

Then the jury asks questions.

A question of type 1 is a line formatted as "`? 1` i j k" ($1 \le i, j, k \le N$; i, j, and k are pairwise distinct). You then print a line with a single integer: the median of the values a_i, a_j, and a_k.

A question of type 2 is a line formatted as "`? 2 i j`" ($1 \le i, j \le N$; $i \ne j$). You then print a line with a single integer: i if $a_i < a_j$, or j if $a_i > a_j$.

A question of type 3 is a line formatted as "`? 3 i j`" ($1 \le i, j \le N$; $i \ne j$). You then print a line with a single integer: the minimum value among a_i and a_j.

Let there will be a total of q_1 questions of type 1, q_2 questions of type 2, and q_3 questions of type 3. You may assume that the value $p = 2N - 2q_1 - 2q_2 - q_3$ is **strictly greater** than 2.

To switch to the answering phase, the jury program issues a line consisting of the '`!`' sign. After that, you must print two lines, the first line containing N space-separated integers $a_1, \ldots, a_N$, and the second line containing N space-separated integers $b_2, \ldots, b_N$. Each of these two sequences must be a permutation of $1, \ldots, N$, and they must differ in at least $\lceil p/2 \rceil$ positions.

Don't forget to print the newline character and flush the output buffer after printing the answers and after printing each permutation, otherwise, you may get the "`Idleness Limit Exceeded`" error.

Example

standard input	standard output
5 ? 1 1 2 3 ? 2 2 4 ? 3 4 5 !	 4 4 1 3 5 4 1 2 5 4 3 2 1

Problem M. Multiple Parentheses

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Consider strings consisting of the brackets '(' and ')'.

The regular bracket sequences are the strings which can be obtained by the following rules:

- Empty string is a regular bracket sequence.
- If A is a regular bracket sequence, then (A) is a regular bracket sequence.
- If A and B are regular bracket sequences, then the concatenation of A and B is a regular bracket sequence.

You are given N boxes numbered $1, 2, \ldots, N$, and also two integers, M and K. Your task is to put exactly one regular bracket sequence in each of N boxes such that the following conditions are met:

- The total number of '(' brackets in all N boxes is equal to M.
- The regular bracket sequences of length $2 \cdot K$ cannot be put into the boxes.

Count the number of different ways to do that. Two distributions are considered different if there exists a number i such that box i contains different regular bracket sequences in those distributions.

Because the answer may be very large, print the answer modulo 998 244 353.

Input

The input contains one line with three integers N, M, and K in it ($1 \le M, N \le 10^6$, $1 \le K \le M$).

Output

Print the answer modulo 998 244 353.

Examples

standard input	standard output
2 2 1	4
1 1 1	0
24 120 30	379268651

Note

For the first example, the following distributions meet the conditions:

- (()), empty;
- ()(), empty;
- empty, (());
- empty, ()().

So, the answer is 4.