A. Candies

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarpus has got n candies and m friends ($n \geq m$). He wants to make a New Year present with candies to each friend. Polycarpus is planning to present all candies and he wants to do this in the fairest (that is, most equal) manner. He wants to choose such a_i, where a_i is the number of candies in the i-th friend's present, that the maximum a_i differs from the least a_i as little as possible.

For example, if n is divisible by m, then he is going to present the same number of candies to all his friends, that is, the maximum a_i won't differ from the minimum one.

Input

The single line of the input contains a pair of space-separated positive integers n, m ($1 \leq n, m \leq 100; n \geq m$) — the number of candies and the number of Polycarpus's friends.

Output

Print the required sequence $a_1, a_2, \ldots, a_m$, where a_i is the number of candies in the i-th friend's present. All numbers a_i must be positive integers, total up to n, the maximum one should differ from the minimum one by the smallest possible value.

Examples

input
12 3
output
4 4 4

input
15 4
output
3 4 4 4

input
18 7
output
2 2 2 3 3 3 3

Note

Print a_i in any order, separate the numbers by spaces.

B. Optimizer

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

A process RAM is a sequence of bytes that are indexed from 1 to n. Polycarpus's program contains such instructions as "`memset`", that is, the operations of filling memory cells on a segment with some value. The details are: the code only contains m instructions that look like "`set13 a_i l_i`". Instruction i fills a continuous memory segment of length l_i, starting from cell number a_i, (that it cells with numbers a_i, $a_i + 1$, ..., $a_{i + l_i - 1}$) with values 13.

In Polycarpus's code, the optimizer's task is to remove the maximum number of instructions from his code in such a way that the remaining instructions set value 13 in all the memory bytes that got this value from the code before the optimization. Also, the value 13 should be set only in the memory bytes that got this value from the code before the optimization. Your task is to implement the optimizer for such program.

Input

The first line contains integers n and m ($1 \le n \le 2 \cdot 10^6$, $1 \le m \le 2 \cdot 10^5$) — the number of bytes (memory cells) and the number of instructions in Polycarpus's code. Then m lines follow, each line contains a pair of integers a_i, l_i ($1 \le a_i \le n$, $1 \le l_i \le n - a_i + 1$).

Output

Print in the first line the sought maximum number of instructions that can be removed from the code. In the second line print the numbers of the instructions. The instructions are numbered from 1 to m in the order they appeared in the input. If there are multiple solutions, print any of them.

Examples

input
10 4 3 3 3 1 4 1 9 2
output
2 2 3

input
1 1 1 1
output
0

C. White, Black and White Again

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarpus is sure that his life fits the description: "first there is a white stripe, then a black one, then a white one again". So, Polycarpus is sure that this rule is going to fulfill during the next n days. Polycarpus knows that he is in for w good events and b not-so-good events. At least one event is going to take place during each day. As each day is unequivocally characterizes as a part of a white or a black stripe, then each day is going to have events of the same type only (ether good or not-so-good).

What is the number of distinct ways this scenario can develop over the next n days if Polycarpus is in for a white stripe (a stripe that has good events only, the stripe's length is at least 1 day), the a black stripe (a stripe that has not-so-good events only, the stripe's length is at least 1 day) and a white stripe again (a stripe that has good events only, the stripe's length is at least 1 day). Each of n days will belong to one of the three stripes only.

Note that even the events of the same type are distinct from each other. Even if some events occur on the same day, they go in some order (there are no simultaneous events).

Write a code that prints the number of possible configurations to sort the events into days. See the samples for clarifications on which scenarios should be considered distinct. Print the answer modulo 1000000009 $(10^9 + 9)$.

Input

The single line of the input contains integers n, w and b ($3 \le n \le 4000$, $2 \le w \le 4000$, $1 \le b \le 4000$) — the number of days, the number of good events and the number of not-so-good events. It is guaranteed that $w + b \ge n$.

Output

Print the required number of ways modulo 1000000009 $(10^9 + 9)$.

Examples

input
3 2 1
output
2

input
4 2 2
output
4

input
3 2 2
output
4

Note

We'll represent the good events by numbers starting from 1 and the not-so-good events — by letters starting from 'a'. Vertical lines separate days.

In the first sample the possible ways are: "1|a|2" and "2|a|1". In the second sample the possible ways are: "1|a|b|2", "2|a|b|1", "1|b|a|2" and "2|b|a|1". In the third sample the possible ways are: "1|ab|2", "2|ab|1", "1|ba|2" and "2|ba|1".

D. Polygon

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarpus loves convex polygons, especially if all their angles are the same and all their sides are different. Draw for him any such polygon with the given number of vertexes.

Input

The input contains a single integer n ($3 \le n \le 100$) — the number of the polygon vertexes.

Output

Print n lines, containing the coordinates of the vertexes of the n-gon "x_i y_i" in the counter clockwise order. The coordinates of the vertexes shouldn't exceed 10^6 in their absolute value. The side lengths should fit within limits $[1, 1000]$ (not necessarily integer). Mutual comparing sides and angles of your polygon during the test will go with precision of 10^{-3}.

If there is no solution, print "No solution" (without the quotes).

Examples

input
8

output
1.000 0.000 7.000 0.000 9.000 2.000 9.000 3.000 5.000 7.000 3.000 7.000 0.000 4.000 0.000 1.000