Codeforces Round #542 [Alex Lopashev Thanks-Round] (Div. 2)

A. Be Positive

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

You are given an array of n integers: $a_1, a_2, \ldots, a_n$. Your task is to find some **non-zero integer** d ($-10^3 \leq d \leq 10^3$) such that, after each number in the array is divided by d, the number of positive numbers that are presented in the array is greater than or equal to half of the array size (i.e., at least $\lceil \frac{n}{2} \rceil$). Note that those positive numbers do not need to be an integer (e.g., a 2.5 counts as a positive number). If there are multiple values of d that satisfy the condition, you may print any of them. In case that there is no such d, print a single integer 0.

Recall that $\lceil x \rceil$ represents the smallest integer that is not less than x and that zero (0) is neither positive nor negative.

Input

The first line contains one integer n ($1 \leq n \leq 100$) — the number of elements in the array.

The second line contains n space-separated integers $a_1, a_2, \ldots, a_n$ ($-10^3 \leq a_i \leq 10^3$).

Output

Print one integer d ($-10^3 \leq d \leq 10^3$ and $d \neq 0$) that satisfies the given condition. If there are multiple values of d that satisfy the condition, you may print any of them. In case that there is no such d, print a single integer 0.

Examples

input
5 10 0 -7 2 6
output
4

input
7 0 0 1 -1 0 0 2
output
0

Note

In the first sample, $n = 5$, so we need at least $\lceil \frac{5}{2} \rceil = 3$ positive numbers after division. If $d = 4$, the array after division is $[2.5, 0, -1.75, 0.5, 1.5]$, in which there are 3 positive numbers (namely: 2.5, 0.5, and 1.5).

In the second sample, there is no valid d, so 0 should be printed.

B. Two Cakes

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

Sasha and Dima want to buy two n-tier cakes. Each cake should consist of n different tiers: from the size of 1 to the size of n. Tiers should go in order from the smallest to the biggest (from top to bottom).

They live on the same street, there are $2 \cdot n$ houses in a row from left to right. Each house has a pastry shop where you can buy a cake tier. Unfortunately, in each pastry shop you can buy only one tier of only one specific size: in the i-th house you can buy a tier of the size a_i ($1 \leq a_i \leq n$).

Since the guys carry already purchased tiers, and it is impossible to insert a new tier in the middle of the cake, they agreed to buy tiers from the smallest to the biggest. That is, each of them buys tiers in order: 1, then 2, then 3 and so on up to n.

Initially, Sasha and Dima are located near the first (leftmost) house. Output the minimum distance that they will have to walk in total to buy both cakes. The distance between any two neighboring houses is exactly 1.

Input

The first line of the input contains an integer number n — the number of tiers in each cake ($1 \leq n \leq 10^5$).

The second line contains $2 \cdot n$ integers $a_1, a_2, \ldots, a_{2n}$ ($1 \le a_i \le n$), where a_i is equal to the size of the tier, which can be bought in the i-th house. Remember that in each house you can buy only one tier. It is guaranteed that every number from 1 to n occurs in a exactly two times.

Output

Print one number — the minimum distance that the guys have to walk in total to buy both cakes. Guys can be near same house at the same time. They begin near the first (leftmost) house. Each of the guys should buy n tiers in ascending order of their sizes.

Examples

input
3 1 1 2 2 3 3
output
9

input
2 2 1 1 2
output
5

input
4 4 1 3 2 2 3 1 4
output
17

Note

In the first example, the possible optimal sequence of actions is:

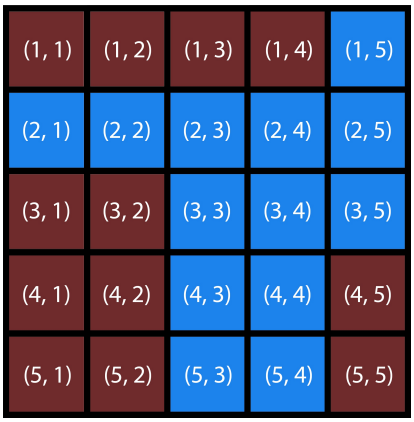
- Sasha buys a tier of size 1 near the 1-st house ($a_1 = 1$);
- Dima goes to the house 2;
- Dima buys a tier of size 1 near the 2-nd house ($a_2 = 1$);
- Sasha goes to the house 4;
- Sasha buys a tier of size 2 near the 4-th house ($a_4 = 2$);
- Sasha goes to the house 5;
- Sasha buys a tier of size 3 near the 5-th house ($a_5 = 3$);
- Dima goes to the house 3;
- Dima buys a tier of size 2 near the 3-rd house ($a_3 = 2$);
- Dima goes to the house 6;
- Dima buys a tier of size 3 near the 6-th house ($a_6 = 3$).

So, Sasha goes the distance $3 + 1 = 4$, and Dima goes the distance $1 + 1 + 3 = 5$. In total, they cover a distance of $4 + 5 = 9$. You can make sure that with any other sequence of actions they will walk no less distance.

C. Connect

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

Alice lives on a flat planet that can be modeled as a square grid of size $n \times n$, with rows and columns enumerated from 1 to n. We represent the cell at the intersection of row r and column c with ordered pair (r, c). Each cell in the grid is either *land* or *water*.



An example planet with $n = 5$. It also appears in the first sample test.

Alice resides in *land* cell (r_1, c_1). She wishes to travel to *land* cell (r_2, c_2). At any moment, she may move to one of the cells adjacent to where she is—in one of the four directions (i.e., up, down, left, or right).

Unfortunately, Alice cannot swim, and there is no viable transportation means other than by foot (i.e., she can walk only on *land*). As a result, Alice's trip may be impossible.

To help Alice, you plan to create **at most one** tunnel between some two *land* cells. The tunnel will allow Alice to freely travel between the two endpoints. Indeed, creating a tunnel is a lot of effort: the cost of creating a tunnel between cells (r_s, c_s) and (r_t, c_t) is $(r_s - r_t)^2 + (c_s - c_t)^2$.

For now, your task is to find the minimum possible cost of creating at most one tunnel so that Alice could travel from (r_1, c_1) to (r_2, c_2). If no tunnel needs to be created, the cost is 0.

Input
The first line contains one integer n ($1 \le n \le 50$) — the width of the square grid.

The second line contains two space-separated integers r_1 and c_1 ($1 \le r_1, c_1 \le n$) — denoting the cell where Alice resides.

The third line contains two space-separated integers r_2 and c_2 ($1 \le r_2, c_2 \le n$) — denoting the cell to which Alice wishes to travel.

Each of the following n lines contains a string of n characters. The j-th character of the i-th such line ($1 \le i, j \le n$) is `0` if (i, j) is *land* or `1` if (i, j) is *water*.

It is guaranteed that (r_1, c_1) and (r_2, c_2) are *land*.

Output
Print an integer that is the minimum possible cost of creating at most one tunnel so that Alice could travel from (r_1, c_1) to (r_2, c_2).

Examples

input
5 1 1 5 5 00001 11111 00111 00110 00110
output
10

input
3 1 3 3 1 010 101 010
output
8

Note
In the first sample, a tunnel between cells $(1, 4)$ and $(4, 5)$ should be created. The cost of doing so is $(1 - 4)^2 + (4 - 5)^2 = 10$, which is optimal. This way, Alice could walk from $(1, 1)$ to $(1, 4)$, use the tunnel from $(1, 4)$ to $(4, 5)$, and lastly walk from $(4, 5)$ to $(5, 5)$.

In the second sample, clearly a tunnel between cells $(1, 3)$ and $(3, 1)$ needs to be created. The cost of doing so is $(1 - 3)^2 + (3 - 1)^2 = 8$.

D1. Toy Train (Simplified)

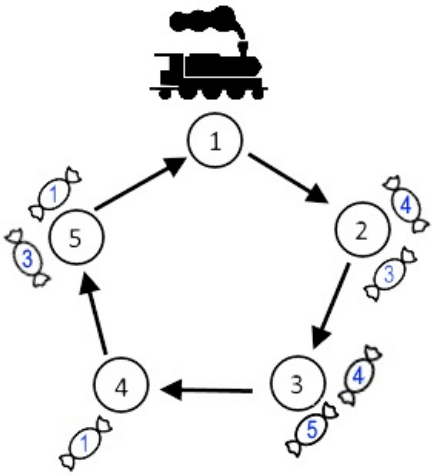
time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This is a simplified version of the task Toy Train. These two versions differ only in the constraints. Hacks for this version are disabled.

Alice received a set of Toy Train™ from Bob. It consists of one train and a connected railway network of n stations, enumerated from 1 through n. The train occupies one station at a time and travels around the network of stations in a circular manner. More precisely, the immediate station that the train will visit after station i is station $i + 1$ if $1 \le i < n$ or station 1 if $i = n$. It takes the train 1 second to travel to its next station as described.

Bob gave Alice a fun task before he left: to deliver m candies that are initially at some stations to their independent destinations using the train. The candies are enumerated from 1 through m. Candy i ($1 \le i \le m$), now at station a_i, should be delivered to

station b_i ($a_i \neq b_i$).



The blue numbers on the candies correspond to b_i values. The image corresponds to the 1-st example.

The train has infinite capacity, and it is possible to load off any number of candies at a station. However, only **at most one** candy can be loaded from a station onto the train before it leaves the station. You can choose any candy at this station. The time it takes to move the candies is negligible.

Now, Alice wonders how much time is needed for the train to deliver all candies. Your task is to find, for each station, the minimum time the train would need to deliver all the candies were it to start from there.

Input

The first line contains two space-separated integers n and m ($2 \leq n \leq 100$; $1 \leq m \leq 200$) — the number of stations and the number of candies, respectively.

The i-th of the following m lines contains two space-separated integers a_i and b_i ($1 \leq a_i, b_i \leq n$; $a_i \neq b_i$) — the station that initially contains candy i and the destination station of the candy, respectively.

Output

In the first and only line, print n space-separated integers, the i-th of which is the minimum time, in seconds, the train would need to deliver all the candies were it to start from station i.

Examples

input
5 7 2 4 5 1 2 3 3 4 4 1 5 3 3 5
output
10 9 10 10 9

input
2 3 1 2 1 2 1 2
output
5 6

Note

Consider the second sample.

If the train started at station 1, the optimal strategy is as follows.

1. Load the first candy onto the train.
2. Proceed to station 2. This step takes 1 second.
3. Deliver the first candy.
4. Proceed to station 1. This step takes 1 second.
5. Load the second candy onto the train.
6. Proceed to station 2. This step takes 1 second.
7. Deliver the second candy.
8. Proceed to station 1. This step takes 1 second.
9. Load the third candy onto the train.
10. Proceed to station 2. This step takes 1 second.
11. Deliver the third candy.

Hence, the train needs 5 seconds to complete the tasks.

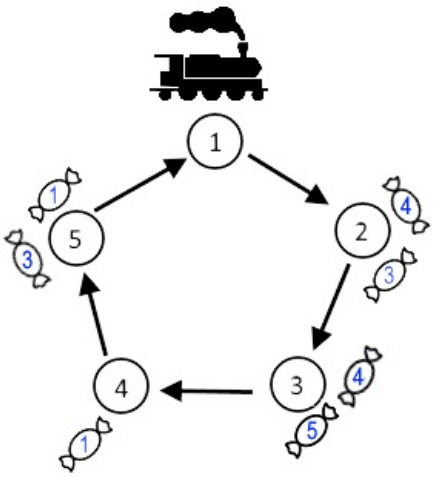
If the train were to start at station 2, however, it would need to move to station 1 before it could load the first candy, which would take one additional second. Thus, the answer in this scenario is $5 + 1 = 6$ seconds.

D2. Toy Train

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Alice received a set of Toy Train™ from Bob. It consists of one train and a connected railway network of n stations, enumerated from 1 through n. The train occupies one station at a time and travels around the network of stations in a circular manner. More precisely, the immediate station that the train will visit after station i is station $i + 1$ if $1 \leq i < n$ or station 1 if $i = n$. It takes the train 1 second to travel to its next station as described.

Bob gave Alice a fun task before he left: to deliver m candies that are initially at some stations to their independent destinations using the train. The candies are enumerated from 1 through m. Candy i ($1 \leq i \leq m$), now at station a_i, should be delivered to station b_i ($a_i \neq b_i$).



The blue numbers on the candies correspond to b_i values. The image corresponds to the 1-st example.

The train has infinite capacity, and it is possible to load off any number of candies at a station. However, only **at most one** candy can be loaded from a station onto the train before it leaves the station. You can choose any candy at this station. The time it takes to move the candies is negligible.

Now, Alice wonders how much time is needed for the train to deliver all candies. Your task is to find, for each station, the minimum time the train would need to deliver all the candies were it to start from there.

Input

The first line contains two space-separated integers n and m ($2 \leq n \leq 5\,000$; $1 \leq m \leq 20\,000$) — the number of stations and the number of candies, respectively.

The i-th of the following m lines contains two space-separated integers a_i and b_i ($1 \leq a_i, b_i \leq n$; $a_i \neq b_i$) — the station that initially contains candy i and the destination station of the candy, respectively.

Output

In the first and only line, print n space-separated integers, the i-th of which is the minimum time, in seconds, the train would need to deliver all the candies were it to start from station i.

Examples

input
5 7 2 4 5 1 2 3 3 4 4 1 5 3 3 5
output
10 9 10 10 9

input
2 3 1 2 1 2 1 2
output

```
5 6
```

Note

Consider the second sample.

If the train started at station 1, the optimal strategy is as follows.

1. Load the first candy onto the train.
2. Proceed to station 2. This step takes 1 second.
3. Deliver the first candy.
4. Proceed to station 1. This step takes 1 second.
5. Load the second candy onto the train.
6. Proceed to station 2. This step takes 1 second.
7. Deliver the second candy.
8. Proceed to station 1. This step takes 1 second.
9. Load the third candy onto the train.
10. Proceed to station 2. This step takes 1 second.
11. Deliver the third candy.

Hence, the train needs 5 seconds to complete the tasks.

If the train were to start at station 2, however, it would need to move to station 1 before it could load the first candy, which would take one additional second. Thus, the answer in this scenario is $5 + 1 = 6$ seconds.

E. Wrong Answer

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

Consider the following problem: given an array a containing n integers (indexed from 0 to $n-1$), find $\max\limits_{0 \le l \le r \le n-1} \sum\limits_{l \le i \le r} (r-l+1) \cdot a_i$. In this problem, $1 \le n \le 2\,000$ and $|a_i| \le 10^6$.

In an attempt to solve the problem described, Alice quickly came up with a blazing-fast greedy algorithm and coded it. Her implementation in pseudocode is as follows:

```
function find_answer(n, a)
  # Assumes n is an integer between 1 and 2000, inclusive
  # Assumes a is a list containing n integers: a[0], a[1], ..., a[n-1]
  res = 0
  cur = 0
  k = -1
  for i = 0 to i = n-1
    cur = cur + a[i]
    if cur < 0
      cur = 0
      k = i
    res = max(res, (i-k)*cur)
  return res
```

Also, as you can see, Alice's idea is not entirely correct. For example, suppose $n = 4$ and $a = [6, -8, 7, -42]$. Then, `find_answer(n, a)` would return 7, but the correct answer is $3 \cdot (6 - 8 + 7) = 15$.

You told Alice that her solution is incorrect, but she did not believe what you said.

Given an integer k, you are to find any sequence a of n integers such that the correct answer and the answer produced by Alice's algorithm differ by exactly k. Note that although the choice of n and the content of the sequence is yours, you must still follow the constraints earlier given: that $1 \le n \le 2\,000$ and that the absolute value of each element does not exceed 10^6. If there is no such sequence, determine so.

Input

The first and only line contains one integer k ($1 \le k \le 10^9$).

Output

If there is no sought sequence, print "`-1`".

Otherwise, in the first line, print one integer n ($1 \le n \le 2\,000$), denoting the number of elements in the sequence.

Then, in the second line, print n space-separated integers: $a_0, a_1, \ldots, a_{n-1}$ ($|a_i| \le 10^6$).

Examples

input
8
output
4 6 -8 7 -42

input
612
output
7 30 -12 -99 123 -2 245 -300

Note

The first sample corresponds to the example given in the problem statement.

In the second sample, one answer is $n = 7$ with $a = [30, -12, -99, 123, -2, 245, -300]$, in which case `find_answer(n, a)` returns 1098, while the correct answer is 1710.