Codeforces Round #542 [Alex Lopashev Thanks-Round] (Div. 1)

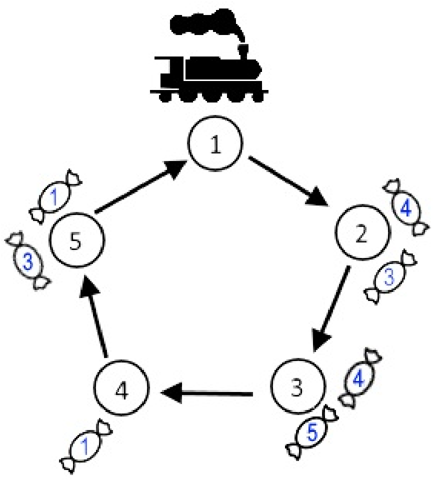
A1. Toy Train (Simplified)

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 $1 \leq i < n$ 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 ($2 \leq n \leq 100$; $1 \leq m \leq 200$) — 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.

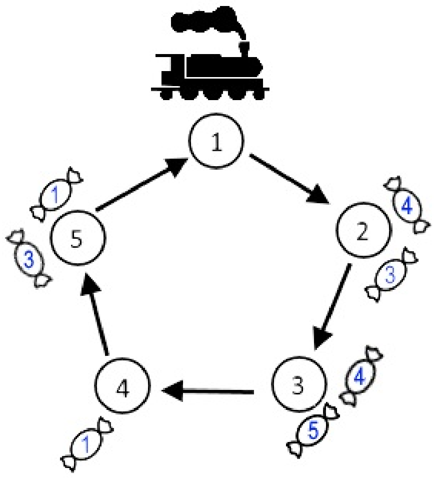
If the train were to start at station 2, however, it would need to move to station 1 before it could load the first candy, which would take one additional second. Thus, the answer in this scenario is $5 + 1 = 6$ seconds.

A2. 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.

B. 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.

C. Morse Code

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

In Morse code, an letter of English alphabet is represented as a string of some length from 1 to 4. Moreover, each Morse code representation of an English letter contains only dots and dashes. In this task, we will represent a dot with a "`0`" and a dash with a "`1`".

Because there are $2^1 + 2^2 + 2^3 + 2^4 = 30$ strings with length 1 to 4 containing only "`0`" and/or "`1`", not all of them correspond to one of the 26 English letters. In particular, each string of "`0`" and/or "`1`" of length **at most** 4 translates into a distinct English letter, except the following four strings that do not correspond to any English alphabet: "`0011`", "`0101`", "`1110`", and "`1111`".

You will work with a string S, which is initially empty. For m times, either a dot or a dash will be appended to S, one at a time. Your task is to find and report, after each of these modifications to string S, the number of non-empty sequences of English letters that are represented with some substring of S in Morse code.

Since the answers can be incredibly tremendous, print them modulo $10^9 + 7$.

Input

The first line contains an integer m ($1 \le m \le 3\,000$) — the number of modifications to S.

Each of the next m lines contains either a "`0`" (representing a dot) or a "`1`" (representing a dash), specifying which character should be appended to S.

Output

Print m lines, the i-th of which being the answer after the i-th modification to S.

Examples

input

```
3
1
1
1
```

output

```
1
3
7
```

input

```
5
1
0
1
0
1
```

output

```
1
4
10
22
43
```

input

```
9
1
1
0
0
0
1
1
0
1
```

output

```
1
3
10
24
51
109
213
421
833
```

Note

Let us consider the first sample after all characters have been appended to S, so S is "111".

As you can see, "1", "11", and "111" all correspond to some distinct English letter. In fact, they are translated into a 'T', an 'M', and an 'O', respectively. All non-empty sequences of English letters that are represented with some substring of S in Morse code, therefore, are as follows.

1. "T" (translates into "1")
2. "M" (translates into "11")
3. "O" (translates into "111")
4. "TT" (translates into "11")
5. "TM" (translates into "111")
6. "MT" (translates into "111")
7. "TTT" (translates into "111")

Although unnecessary for this task, a conversion table from English alphabets into Morse code can be found here.

D. Isolation

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

Find the number of ways to divide an array a of n integers into any number of disjoint non-empty segments so that, in each segment, there exist at most k distinct integers that appear exactly once.

Since the answer can be large, find it modulo $998\,244\,353$.

Input

The first line contains two space-separated integers n and k ($1 \leq k \leq n \leq 10^5$) — the number of elements in the array a and the restriction from the statement.

The following line contains n space-separated integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le n)$ — elements of the array a.

Output

The first and only line contains the number of ways to divide an array a modulo $998\,244\,353$.

Examples

input

```
3 1
1 1 2
```

output

```
3
```

input

```
5 2
1 1 2 1 3
```

output

```
14
```

input

```
5 5
1 2 3 4 5
```

output

```
16
```

Note

In the first sample, the three possible divisions are as follows.

- $[[1], [1], [2]]$
- $[[1, 1], [2]]$
- $[[1, 1, 2]]$

Division $[[1], [1, 2]]$ is not possible because two distinct integers appear exactly once in the second segment $[1, 2]$.

E. Legendary Tree

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

This is an interactive problem.

A legendary tree rests deep in the forest. Legend has it that individuals who realize this tree would eternally become a Legendary Grandmaster.

To help you determine the tree, Mikaela the Goddess has revealed to you that the tree contains n vertices, enumerated from 1 through n. She also allows you to ask her some questions as follows. For each question, you should tell Mikaela some two **disjoint** non-empty sets of vertices S and T, along with any vertex v that you like. Then, Mikaela will count and give you the number of pairs of vertices (s, t) where $s \in S$ and $t \in T$ such that the simple path from s to t contains v.

Mikaela the Goddess is busy and will be available to answer at most $11\,111$ questions.

This is your only chance. Your task is to determine the tree and report its edges.

Input

The first line contains an integer n $(2 \le n \le 500)$ — the number of vertices in the tree.

Output

When program has realized the tree and is ready to report the edges, print "ANSWER" in a separate line. Make sure that all letters are capitalized.

Then, print $n - 1$ lines, each containing two space-separated integers, denoting the vertices that are the endpoints of a certain edge. Each edge should be reported exactly once. Your program should then immediately terminate.

Interaction

For each question that you wish to ask, interact as follows.

1. First, print the size of S in its own line. In the following line, print $|S|$ space-separated distinct integers, denoting the vertices in S.
2. Similarly, print the size of T in its own line. In the following line, print $|T|$ space-separated distinct integers, denoting the vertices in T.

3. Then, in the final line, print v — the vertex that you choose for this question.
4. Read Mikaela's answer from input.

Be reminded that S and T must be disjoint and non-empty.

After printing a query do not forget to output end of line and flush the output. Otherwise you will get `Idleness limit exceeded`. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;
- `stdout.flush()` in Python;
- see documentation for other languages.

If your program asks too many questions, asks an invalid question or does not correctly follow the interaction guideline above, it may receive an arbitrary verdict. Otherwise, your program will receive the `Wrong Answer` verdict if it reports an incorrect tree.

Note that the tree is fixed beforehand and does not depend on your queries.

Hacks

Hacks should be formatted as follows.

The first line should contain a single integer n $(2 \le n \le 500)$ — the number of vertices in the tree.

The following $n-1$ lines should each contain two space-separated integers u and v, denoting the existence of an undirected edge (u, v) $(1 \le u, v \le n)$.

Example

input

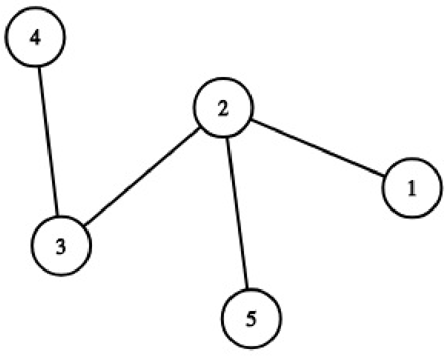
```
5
5
```

output

```
3
1 2 3
2
4 5
2
ANSWER
1 2
2 3
3 4
2 5
```

Note

In the sample, the tree is as follows.



$n = 5$ is given to the program. The program then asks Mikaela a question where $S = \{1, 2, 3\}$, $T = \{4, 5\}$, and $v = 2$, to which she replies with 5 (the pairs (s, t) are $(1, 4)$, $(1, 5)$, $(2, 4)$, $(2, 5)$, and $(3, 5)$).