Codeforces Round #557 (Div. 1) [based on Forethought Future Cup - Final Round]

A. Hide and Seek

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

Alice and Bob are playing a game on a line with n cells. There are n cells labeled from 1 through n. For each i from 1 to $n-1$, cells i and $i+1$ are adjacent.

Alice initially has a token on some cell on the line, and Bob tries to guess where it is.

Bob guesses a sequence of line cell numbers $x_1, x_2, \ldots, x_k$ in order. In the i-th question, Bob asks Alice if her token is currently on cell x_i. That is, Alice can answer either "YES" or "NO" to each Bob's question.

At most one time in this process, before or after answering a question, Alice is allowed to move her token from her current cell to some **adjacent** cell. Alice acted in such a way that she was able to answer "NO" to **all** of Bob's questions.

Note that Alice can even move her token before answering the first question or after answering the last question. Alice can also choose to not move at all.

You are given n and Bob's questions $x_1, \ldots, x_k$. You would like to count the number of scenarios that let Alice answer "NO" to all of Bob's questions.

Let (a, b) denote a scenario where Alice starts at cell a and ends at cell b. Two scenarios (a_i, b_i) and (a_j, b_j) are different if $a_i \neq a_j$ or $b_i \neq b_j$.

Input

The first line contains two integers n and k ($1 \leq n, k \leq 10^5$) — the number of cells and the number of questions Bob asked.

The second line contains k integers $x_1, x_2, \ldots, x_k$ ($1 \leq x_i \leq n$) — Bob's questions.

Output

Print a single integer, the number of scenarios that let Alice answer "NO" to all of Bob's questions.

Examples

input
5 3 5 1 4
output
9

input
4 8 1 2 3 4 4 3 2 1
output
0

input
100000 1 42
output
299997

Note

The notation (i, j) denotes a scenario where Alice starts at cell i and ends at cell j.

In the first example, the valid scenarios are $(1, 2), (2, 1), (2, 2), (2, 3), (3, 2), (3, 3), (3, 4), (4, 3), (4, 5)$. For example, $(3, 4)$ is valid since Alice can start at cell 3, stay there for the first three questions, then move to cell 4 after the last question.

$(4, 5)$ is valid since Alice can start at cell 4, stay there for the first question, the move to cell 5 for the next two questions. Note that $(4, 5)$ is only counted once, even though there are different questions that Alice can choose to do the move, but remember, we only count each pair of starting and ending positions once.

In the second example, Alice has no valid scenarios.

In the last example, all (i, j) where $|i - j| \leq 1$ except for $(42, 42)$ are valid scenarios.

B. Chladni Figure

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

Inaka has a disc, the circumference of which is n units. The circumference is equally divided by n points numbered clockwise from 1 to n, such that points i and $i + 1$ ($1 \leq i < n$) are adjacent, and so are points n and 1.

There are m straight segments on the disc, the endpoints of which are all among the aforementioned n points.

Inaka wants to know if her image is *rotationally symmetrical*, i.e. if there is an integer k ($1 \leq k < n$), such that if all segments are rotated clockwise around the center of the circle by k units, the new image will be the same as the original one.

Input

The first line contains two space-separated integers n and m ($2 \leq n \leq 100\,000$, $1 \leq m \leq 200\,000$) — the number of points and the number of segments, respectively.

The i-th of the following m lines contains two space-separated integers a_i and b_i ($1 \leq a_i, b_i \leq n$, $a_i \neq b_i$) that describe a segment connecting points a_i and b_i.

It is guaranteed that no segments coincide.

Output

Output one line — "Yes" if the image is rotationally symmetrical, and "No" otherwise (both excluding quotation marks).

You can output each letter in any case (upper or lower).

Examples

input
12 6 1 3 3 7 5 7 7 11 9 11 11 3
output
Yes

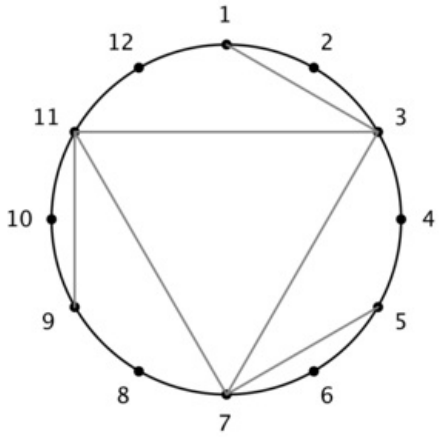
input
9 6 4 5 5 6 7 8 8 9 1 2 2 3
output
Yes

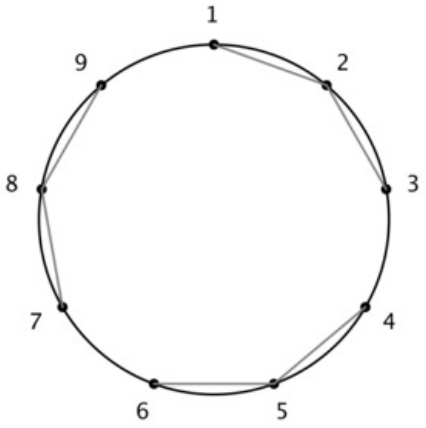
input
10 3 1 2 3 2 7 2
output
No

input
10 2 1 6 2 7
output
Yes

Note

The first two examples are illustrated below. Both images become the same as their respective original ones after a clockwise rotation of 120 degrees around the center.





C. Thanos Nim

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

Alice and Bob are playing a game with n piles of stones. It is guaranteed that n is an even number. The i-th pile has a_i stones.

Alice and Bob will play a game alternating turns with Alice going first.

On a player's turn, they must choose **exactly** $\frac{n}{2}$ nonempty piles and independently remove a positive number of stones from each of the chosen piles. They **can** remove a **different** number of stones from the piles in a single turn. The first player unable to make a move loses (when there are less than $\frac{n}{2}$ nonempty piles).

Given the starting configuration, determine who will win the game.

Input

The first line contains one integer n ($2 \le n \le 50$) — the number of piles. It is guaranteed that n is an even number.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 50$) — the number of stones in the piles.

Output

Print a single string "Alice" if Alice wins; otherwise, print "Bob" (without double quotes).

Examples

input
2 8 8
output
Bob

input
4 3 1 4 1
output
Alice

Note

In the first example, each player can only remove stones from one pile ($\frac{2}{2} = 1$). Alice loses, since Bob can copy whatever Alice does on the other pile, so Alice will run out of moves first.

In the second example, Alice can remove 2 stones from the first pile and 3 stones from the third pile on her first move to guarantee a win.

D. Palindrome XOR

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

You are given a string s consisting of characters "1", "0", and "?". The first character of s is guaranteed to be "1". Let m be the number of characters in s.

Count the number of ways we can choose a pair of integers a, b that satisfies the following:

- $1 \leq a < b < 2^m$
- When written without leading zeros, the base-2 representations of a and b are both palindromes.
- The base-2 representation of `bitwise XOR` of a and b matches the pattern s. We say that t matches s if the lengths of t and s are the same and for every i, the i-th character of t is equal to the i-th character of s, or the i-th character of s is "?".

Compute this count modulo 998244353.

Input
The first line contains a single string s ($1 \leq |s| \leq 1\,000$). s consists only of characters "1", "0" and "?". It is guaranteed that the first character of s is a "1".

Output
Print a single integer, the count of pairs that satisfy the conditions modulo 998244353.

Examples

input
10110
output
3

input
1?0???10
output
44

input
1??????????????????????????????????????
output
519569202

input
1
output
0

Note
For the first example, the pairs in base-2 are $(111, 10001), (11, 10101), (1001, 11111)$.

E. Rainbow Coins

time limit per test: 6 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Carl has n coins of various colors, and he would like to sort them into piles. The coins are labeled $1, 2, \ldots, n$, and each coin is exactly one of red, green, or blue. He would like to sort the coins into three different piles so one pile contains all red coins, one pile contains all green coins, and one pile contains all blue coins.

Unfortunately, Carl is colorblind, so this task is impossible for him. Luckily, he has a friend who can take a pair of coins and tell Carl if they are the same color or not. Using his friend, Carl believes he can now sort the coins. The order of the piles doesn't matter, as long as all same colored coins are in the one pile, and no two different colored coins are in the same pile.

His friend will answer questions about multiple pairs of coins in batches, and will answer about all of those pairs in parallel. Each coin should be in at most one pair in each batch. The same coin can appear in different batches.

Carl can use only 7 batches. Help him find the piles of coins after sorting.

Interaction
You will be given multiple test cases. The first line contains an integer t ($1 \leq t \leq 5$) — the number of test cases.

The first line of each test case contains one integer n ($1 \leq n \leq 10^5$) — the number of coins. If you read in a value of -1 here, that means that you printed an invalid answer in the previous case and exit immediately to avoid getting other verdicts.

To ask a question, print "Q k x_1 y_1 $\ldots$ x_k y_k" ($1 \leq k \leq n/2, 1 \leq x_i, y_i \leq n, x_i \neq y_i$). k denotes the number of pairs in the batch, and x_i, y_i denote the i-th pair of coins in the batch. A coin can only appear at most once in a batch. All x_i and y_i should be distinct.

The judge will respond with a bitstring of length k, where the i-th character is "1" if x_i and y_i are the same color, and "0" otherwise. The judge will respond with -1 if you ever ask an invalid query. Make sure to exit immediately to avoid getting other verdicts.

When you are ready to answer, print four lines.

The first line contains "A k_1 k_2 k_3" ($0 \le k_1, k_2, k_3$ and $k_1 + k_2 + k_3 = n$). These denote the sizes of the three piles.

The next line has k_1 integers, the labels of the coins in the first pile.

The next line has k_2 integers, the labels of the coins in the second pile.

The next line has k_3 integers, the labels coins in the third pile.

Each coin must appear in exactly one pile.

You may only ask at most 7 batches per test case.

After printing a query and the answer 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.

Hack Format

To hack, use the following format. Note that you can only hack with one test case.

The first line should contain a single integer t ($t = 1$).

The second line should contain a single string s consisting of only the characters "R", "G", "B" ($1 \le |s| \le 10^5$). The i-th character in this string represents the color of the i-th coin.

Example

input
3 3 1 1 1 3 1 0 0 6 000 010 100 001 000

output
Q 1 1 2 Q 1 2 3 Q 1 1 3 A 3 0 0 1 2 3   Q 1 1 3 Q 1 2 3 Q 1 1 2 A 2 0 1 1 3 2 Q 3 1 2 3 4 5 6 Q 3 1 3 2 5 4 6 Q 3 1 4 2 6 3 5 Q 3 1 5 2 4 3 6 Q 3 1 6 2 3 4 5 A 2 2 2 1 4 2 5 3 6

Note

In the example, there are three test cases.

In the first test case, there are three coins. We ask about the pairs $(1, 2)$, $(2, 3)$, and $(1, 3)$ in different batches and get that they are all the same color. Thus, we know all the coins are the same color, so we can put them all in one pile. Note that some piles can be empty and those are denoted with an empty line.

In the second test case, there are three coins again. This time, we only get that $(1, 3)$ are the same and $(1, 2)$ and $(2, 3)$ are different. So, one possible scenario is that coins 1 and 3 are red and coin 2 is green.

In the last case, there are 6 coins. The case shows how to ask and receive answers for multiple pairs in one batch.

F. Zigzag Game

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a complete bipartite graph with $2n$ nodes, with n nodes on each side of the bipartition. Nodes 1 through n are on one side of the bipartition, and nodes $n+1$ to $2n$ are on the other side. You are also given an $n \times n$ matrix a describing the edge weights. a_{ij} denotes the weight of the edge between nodes i and $j+n$. Each edge has a distinct weight.

Alice and Bob are playing a game on this graph. First Alice chooses to play as either "increasing" or "decreasing" for herself, and Bob gets the other choice. Then she places a token on any node of the graph. Bob then moves the token along any edge incident to that node. They now take turns playing the following game, with Alice going first.

The current player must move the token from the current vertex to some adjacent unvisited vertex. Let w be the last weight of the last edge that was traversed. The edge that is traversed must be strictly greater than w if the player is playing as "increasing", otherwise, it must be strictly less. The first player unable to make a move loses.

You are given n and the edge weights of the graph. You can choose to play as either Alice or Bob, and you will play against the judge. You must win all the games for your answer to be judged correct.

Interaction

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 50$). Description of the test cases follows.

Each test starts with an integer n ($1 \le n \le 50$) — the number of nodes on each side of the bipartition.

The next n lines contain n integers a_{ij} ($1 \le a_{ij} \le n^2$). a_{ij} denotes the weight of the edge between node i and node $j+n$. All a_{ij} will be distinct.

You first print a string "A" or "B", denoting which player you want to play ("A" for Alice and "B" for Bob).

If playing as Alice, first print either "I" or "D" (denoting whether you choose "increasing" or "decreasing"). Then, print the node that you wish to start at.

If playing as Bob, read in a character "I" or "D" (denoting whether the judge chose "increasing" or "decreasing"), then the node that the judge chooses to start at.

To make a move, print the index of the node that you wish to go to. To read what move the judge made, read an integer from standard in.

If the judge responds with -1, then that means the judge has determined it has no legal moves (or it may have just given up) and that you won the case. Stop processing this case immediately and start processing the next case.

If the judge responds with -2, that means that the judge has determined that you made an invalid move, so you should exit immediately to avoid getting other verdicts.

If you are unable to make a move or give up, print -1 and then exit immediately. This will give you a `Wrong answer` verdict.

After printing a move 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.

Hack Format

To hack, use the following format. Note that you can only hack with one test case.

The first line should contain a single integer t ($t = 1$).

The second line should contain an integer n ($1 \le n \le 50$).

The next n lines should contain n integers each, the a_{ij} ($1 \le a_{ij} \le n^2$). a_{ij} denotes the weight of the edge between node i and node $j+n$. All a_{ij} must be distinct.

The judge will play a uniformly random legal move with this case versus the hacked program. If it has no legal moves left, it will give up and declare the player as the winner.

Example

input
2 3 3 1 9 2 5 7

```
6 4 8
6
-1
1
1
I 1
-1
```

output

```
A
D 3
2
B
2
```

Note

The first example has two test cases. In the first test case, the graph looks like the following.

In the sample output, the player decides to play as Alice and chooses "decreasing" and starting at node 3. The judge responds by moving to node 6. After, the player moves to node 2. At this point, the judge has no more moves (since the weight must "increase"), so it gives up and prints -1.

In the next case, we have two nodes connected by an edge of weight 1. The player decides to play as Bob. No matter what the judge chooses, the player can move the token to the other node and the judge has no moves so will lose.