Mechanical Doll

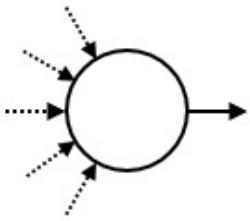
A mechanical doll is a doll which automatically repeats a specific sequence of motions. In Japan, many mechanical dolls have been created since ancient times.

The motions of a mechanical doll are controlled by a **circuit** that consists of **devices**. The devices are connected with tubes. Each device has some (possibly zero) **entrances**, and one or two **exits**. Each device can have arbitrarily many entrances. Each tube connects an exit of a device to an entrance of the same or another device. Exactly one tube is connected to each entrance, and exactly one tube is connected to each exit.

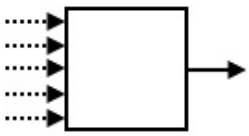
To describe how the doll makes motions, consider a **ball** that is placed on one of the devices. The ball travels through the circuit. At each step of the travel, the ball leaves the device using one of its exits, travels along the tube connected to the exit and enters the device at the other end of the tube.

There are three types of devices: **origin**, **trigger**, and **switch**. There are exactly one origin, M triggers, and S switches (S can be zero). You must decide the value of S. Each device has a unique serial number.

The origin is the device where the ball is initially placed. It has one exit. Its serial number is 0.

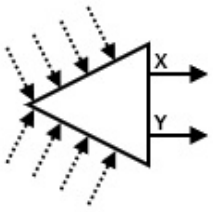


A trigger causes the doll to make one type of motion whenever the ball enters it. Every trigger has one exit. The serial numbers of the triggers are from 1 through M.



Each switch has two exits, which are called 'X' and 'Y'. The **state** of a switch is either 'X' or 'Y'. After the ball enters a switch, it leaves the switch using the exit given by the current state of the switch. After that, the switch changes its state to the opposite one.

Initially, the state of every switch is 'X'. The serial numbers of the switches are from -1 through $-S$.



You are given the number of triggers M. You are also given a sequence A of length N, each of whose element is a serial number of a trigger. Each trigger might appear some (possibly zero) times in the sequence A. Your task is to create a circuit that satisfies the following conditions:

- The ball returns to the origin after some steps.
- When the ball first returns to the origin, the state of every switch is 'X'.
- The ball first returns to the origin after entering triggers exactly N times. The serial numbers of the triggers, in the order that they are entered, are $A_0, A_1, \ldots, A_{N-1}$.
- Let P be the total number of state changes of all switches caused by the ball before the ball first returns to the origin. The value of P doesn't exceed $20\,000\,000$.

At the same time, you don't want to use too many switches.

Implementation details

You should implement the following procedure.

```
create_circuit(int M, int[] A)
```

- `M`: the number of triggers.
- `A`: an array of length N, giving the serial numbers of the triggers the ball needs to enter, in the order they are to be entered.
- This procedure is called exactly once.
- Note that the value of N is the length of the array `A`, and can be obtained as indicated in the implementation notice.

Your program should call the following procedure to answer.

```
answer(int[] C, int[] X, int[] Y)
```

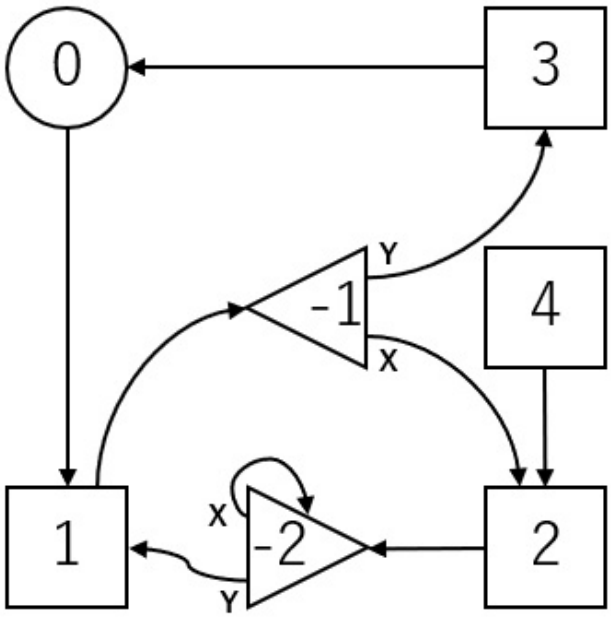
- `C`: an array of length $M + 1$. The exit of the device i ($0 \leq i \leq M$) is connected to the device `C[i]`.
- `X`, `Y`: arrays of the same length. The length S of these arrays is the number of the

switches. For the switch $-j$ ($1 \leq j \leq S$), its exit 'X' is connected to the device `X[j - 1]` and its exit 'Y' is connected to the device `Y[j - 1]`.
- Every element of `C`, `X`, and `Y` must be an integer between $-S$ and M, inclusive.
- S must be at most $400\,000$.
- This procedure must be called exactly once.
- The circuit represented by `C`, `X`, and `Y` must satisfy the conditions in the problem statement.

If some of the above conditions are not satisfied, your program is judged as **`Wrong Answer`**. Otherwise, your program is judged as **`Accepted`** and your score is calculated by S (see Subtasks).

Example

Let $M = 4$, $N = 4$, and $A = [1, 2, 1, 3]$. The grader calls `create_circuit(4, [1, 2, 1, 3])`.



The above figure shows a circuit, which is described by a call `answer([1, -1, -2, 0, 2], [2, -2], [3, 1])`. The numbers in the figure are the serial numbers of the devices.

Two switches are used. Thus $S = 2$.

Initially, the states of the switches -1 and -2 are both 'X'.

The ball travels as follows:

$$0 \longrightarrow 1 \longrightarrow -1 \xrightarrow{\text{X}} 2 \longrightarrow -2 \xrightarrow{\text{X}} -2 \xrightarrow{\text{Y}} 1 \longrightarrow -1 \xrightarrow{\text{Y}} 3 \longrightarrow 0$$

- When the ball first enters the switch -1, its state is 'X'. Hence, the ball travels to the trigger 2. Then the state of the switch -1 is changed to 'Y'.

- When the ball enters the switch -1 for the second time, its state is 'Y'. Hence, the ball travels to the trigger 3. Then the state of the switch -1 is changed to 'X'.

The ball first returns to the origin, having entered the triggers $1, 2, 1, 3$. The states of the switches -1 and -2 are both 'X'. The value of P is 4. Therefore, this circuit satisfies the conditions.

The file `sample-01-in.txt` in the zipped attachment package corresponds to this example. Other sample inputs are also available in the package.

Constraints

- $1 \leq M \leq 100\,000$
- $1 \leq N \leq 200\,000$
- $1 \leq A_k \leq M$ $(0 \leq k \leq N-1)$

Subtasks

The score and the constraints for each test case are as follows:

1. (2 points) For each i $(1 \leq i \leq M)$, the integer i appears at most once in the sequence $A_0, A_1, \ldots, A_{N-1}$.
2. (4 points) For each i $(1 \leq i \leq M)$, the integer i appears at most twice in the sequence $A_0, A_1, \ldots, A_{N-1}$.
3. (10 points) For each i $(1 \leq i \leq M)$, the integer i appears at most 4 times in the sequence $A_0, A_1, \ldots, A_{N-1}$.
4. (10 points) $N = 16$
5. (18 points) $M = 1$
6. (56 points) No additional constraints

For each test case, if your program is judged as **`Accepted`**, your score is calculated according to the value of S:

- If $S \leq N + \log_2 N$, you gain the full score for the test case.
- For each test case in Subtasks 5 and 6, if $N + \log_2 N < S \leq 2N$, you gain a partial score. The score for the test case is $0.5 + 0.4 \times \left(\dfrac{2N - S}{N - \log_2 N} \right)^2$, multiplied by the score assigned to the subtask.
- Otherwise, the score is 0.

Note that your score for each subtask is the minimum of the scores for the test cases in the subtask.

Sample grader

The sample grader reads the input from the standard input in the following format.

- line 1: M N
- line 2: A_0 A_1 $\ldots$ A_{N-1}

The sample grader produces three outputs.

First, the sample grader outputs your answer to a file named `out.txt` in the following format.

- line 1: S
- line $2 + i$ ($0 \leq i \leq M$): `C[i]`
- line $2 + M + j$ ($1 \leq j \leq S$): `X[j - 1] Y[j - 1]`

Second, the sample grader simulates the moves of the ball. It outputs the serial numbers of the devices the ball entered in order to a file named `log.txt`.

Third, the sample grader prints the evaluation of your answer to the standard output.

- If your program is judged as **`Accepted`**, the sample grader prints S and P in the following format `Accepted: S P`.
- If your program is judged as **`Wrong Answer`**, it prints `Wrong Answer: MSG`. The meaning of `MSG` is as follows:
  - `answered not exactly once`: The procedure `answer` is called not exactly once.
  - `wrong array length`: The length of `C` is not $M + 1$, or the lengths of `X` and `Y` are different.
  - `over 400000 switches`: S is larger than $400\,000$.
  - `wrong serial number`: There is an element of `C`, `X`, or `Y` which is smaller than $-S$ or larger than M.
  - `over 20000000 inversions`: The ball doesn't return to the origin within $20\,000\,000$ state changes of the switches.
  - `state 'Y'`: There is a switch whose state is 'Y' when the ball first returns to the origin.
  - `wrong motion`: The triggers which cause motions are different from the sequence A.

Note that the sample grader might not create `out.txt` and/or `log.txt` when your program is judged as `Wrong Answer`.