Data Transfer

Araz has a binary sequence of N bits that wants to send to Baba. As the data might be corrupted during the data transfer, he is going to attach a sequence of K more bits to the source data so that Baba can recover the original N bits in the case of data corruption. We know that at most one bit out of the total $N + K$ bits might be corrupted during the data transfer.

Your task is to help Araz and Baba in sending and receiving the data while minimizing K.

Implementation details

You should implement two different procedures:

```
int[] get_attachment(int[] source)
```

- This procedure plays for Araz.
- *source*: an integer array of length N, demonstrating the binary sequence provided to Araz (all integers are either 0 or 1).
- It should return an array of 0/1 integers, containing the K bit attachment that is appended to the source sequence. This array must not be longer than $2N$.

```
int[] retrieve(int[] data)
```

- This procedure plays for Baba.
- *data*: an integer array of length $N + K$, demonstrating the (possibly corrupted) sequence of data received by Baba (all integers are either 0 or 1).
- It must return an integer array of length N, containing the original sequence of N bits based on the data received by Baba.

Consider a function `manipulate` which receives a binary sequence and returns a similar sequence that at most one of its bits is toggled. Your implementation of `get_attachment` and `retrieve` must be such that for every binary sequence `source` (with length N) and every function `manipulate`, we should have `retrieve(manipulate(data)) = source`, where `data` is the concatenation of `source` and `get_attachment(original)`. Otherwise, your implementation is wrong.

There are T scenarios in a test case. For each scenario, the grader first calls the

procedure `get_attachment` with a source sequence. Next, it may toggle one of the bits in the source or attachment sequence. The result is then passed to the procedure `retrieve`. Note that in the judging system, these procedures are called in separate programs. In the first program, procedure `get_attachment` is called once for each scenario. Invocations of procedure `retrieve` are made in the second program. The behavior of your implementation for each scenario must be independent of the order of the scenarios, as scenarios might not have the same order in the two programs.

Subtasks and scoring

There are two subtasks:

1. (60 points) $N = 63$, $1 \leq T \leq 20\,000$
2. (40 points) $N = 255$, $1 \leq T \leq 200\,000$

Your score in each subtask will be 0 if the retrieved sequence was incorrect for any of the scenarios in any of the test cases. Otherwise, let Q be the maximum size of attachment (K) among all scenarios in all test cases of the subtask. Your score will then be computed as below.

Subtask 1: condition	score		Subtask 2: condition	score
$126 < Q$	0		$510 < Q$	0
$64 < Q \leq 126$	3		$256 < Q \leq 510$	2
$40 < Q \leq 64$	9		$140 < Q \leq 256$	6
$20 < Q \leq 40$	18		$35 < Q \leq 140$	12
$16 < Q \leq 20$	30		$32 < Q \leq 35$	20
$12 < Q \leq 16$	36		$16 < Q \leq 32$	24
$7 < Q \leq 12$	48		$9 < Q \leq 16$	32
$Q \leq 7$	60		$Q \leq 9$	40

Examples

Suppose that Araz attaches the source with the constant sequence $[0, 1, 0]$, and Baba just ignores the attachment and retrieves the first N bits of data as the source sequence. Clearly, this is just an example, not a correct strategy.

The grader makes the following procedure call:

```
get_attachment([0,1,1,0,...,0,0,1])
```

In this example, $source$ is $[0, 1, 1, 0, ..., 0, 0, 1]$ and the procedure returns $[0, 1, 0]$. So, the whole data to be sent is $[0, 1, 1, 0, ..., 0, 0, 1, 0, 1, 0]$.

Now, assume that the last bit of data is corrupted during the data transfer. Hence, the grader makes the following procedure call:

```
retrieve([0,1,1,0,...,0,0,1,0,1,1])
```

The procedure returns $[0, 1, 1, 0, ..., 0, 0, 1]$ which happens to be the correct answer.

Sample grader

The sample grader reads the input in the following format:

- line 1: T
- line $2 + i$ (for $0 \leq i \leq T - 1$): c $source$
  - c: The (0-based) index of the corrupted bit in the data ($-1 \leq c \leq N + K - 1$). If $c = -1$, then none of the bits will be corrupted.
  - $source$: a binary (`0`/`1`) string of length N representing the source sequence.

The sample grader writes the output in the following format:

- line $1 + i$ (for $0 \leq i \leq T - 1$): the verdict of your solution for scenario i. It also prints K when the source sequence is retrieved correctly.