Submissions

Testrun

0pt	Not attempted **0/58 users** correct (0%)

flagpoles

1pt	Not attempted **335/181 users** correct (185%)
11pt	Not attempted **277/320 users** correct (87%)

number_bases

5pt	Not attempted **241/186 users** correct (130%)
17pt	Not attempted **188/226 users** correct (83%)

broken_memory

3pt	Not attempted **196/88 users** correct (223%)
25pt	Not attempted **77/142 users** correct (54%)

nanobots

8pt	Not attempted **104/69 users** correct (151%)
30pt	Not attempted **31/68 users** correct (46%)

Top Scores

fagu	100
bmerry	100
krijgertje	100
ecnerwala	100
pashka	100
Swistakk	100
KalininN	100
adsz	100
Gennady.Korotkevich	100
eatmore	100

Problem D. broken_memory

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

small 3 points *2 minute timeout*	The contest is finished.
large 25 points *10 minute timeout*	The contest is finished.

Problem

Broken Memory

As you may remember from last year, we have a tendency to screw things up at the worst possible moment. For 2017, we promised ourselves that we wouldn't misplace any test cases, though, and we delivered. Unfortunately, last night we spilled a peach smoothie over some of our Google Cloud servers. We immediately called the Dedicated Cloud Janitor (DCJ), but he was fed up with our continuous messes and refused to help. So, we are turning to our most powerful allies, our contestants, to once again rescue us from ourselves.

Fortunately, the servers were already loaded with the data for the problems, so before the damage, the memory was exactly the same in all of them. We conducted a preliminary investigation that revealed that every node's memory was damaged in a *different* place.

We have narrowed the search to a relatively small part of the memory, and we have encoded that as a list of integers for your convenience. Each node has the same list of integers, except at exactly one damaged position; the value there will be *different* from the corresponding value on all other nodes.

For example, suppose the original data is represented by the following list of integers: 1 5 9 3 1. Suppose the broken index for node 0 is 2. That means that when requesting the values for indices 0, 1, 3 and 4, the returned value will be correct (1, 5, 3 and 1, respectively). However, when requesting the value for index 2 on node 0, the returned value could be 1 or 5 or 352462352, for example, but definitely not the correct one (9). If, on the other hand, node 1's broken index is 0, then, when requesting the value of indices 1, 2, 3 and 4, the correct values (5, 9, 3 and 1, respectively) will be returned, but when requesting index 0, the returned value could be 9 or 5 or 379009, but definitely not the correct one (1). Notice that for node 3, neither index 2 nor index 0 can be the broken index, because each node is broken at a different index from all other nodes.

Can you find the broken index on each node for us?

Input

The input library is called "broken_memory"; see the sample inputs below for examples in your language. It defines two methods:

- **GetLength()**:
  - Takes no argument.
  - Returns a 64-bit integer: the number of values in the part of the memory where all the damage happened.
  - Expect each call to take 0.02 microseconds.
- **GetValue(i)**:
  - Takes a 64-bit number in the range $0 \le i <$ GetLength()
  - Returns a 64-bit integer: the i-th value in the memory.
  - Expect each call to take 0.02 microseconds.

Output

Output a single line with `NumberOfNodes()` integers: the broken index for each node, in ascending order of node ID.

Limits

Time limit: 2 seconds.
Memory limit per node: 256 MB.
Maximum number of messages a single node can send: 1000.
Maximum total size of messages a single node can send: 128 KB. **(Notice that this is less than usual.)**
$1 \le$ GetValue(i) $\le 10^{18}$, for all i.

Small dataset

Number of nodes: 10.

$10 \leq$ GetLength() ≤ 1000.

Large dataset

Number of nodes: 100.
$100 \leq$ GetLength() $\leq 10^7$.

In this problem, the sample inputs are valid for running on 10 nodes, and the sample outputs are computed using 10 nodes.

Sample

Input	Output

`See input files below.`	`For sample input 1:`
	`0 1 2 3 4 5 6 7 8 9`
	`For sample input 2:`
	`29 28 27 26 25 24 23 22 21 20`
	`For sample input 3:`
	`12 13 14 15 8 9 10 11 4 5`

Sample input libraries:
Sample input for test 1: broken_memory.h [CPP] broken_memory.java [Java]
Sample input for test 2: broken_memory.h [CPP] broken_memory.java [Java]
Sample input for test 3: broken_memory.h [CPP] broken_memory.java [Java]