Submissions

Testrun

0pt	Not attempted **0/327 users** correct (0%)

pancakes

2pt	Not attempted **984/406 users** correct (242%)
11pt	Not attempted **920/975 users** correct (94%)

weird_editor

3pt	Not attempted **859/434 users** correct (198%)
20pt	Not attempted **505/807 users** correct (63%)

todd_and_steven

1pt	Not attempted **718/365 users** correct (197%)
30pt	Not attempted **230/437 users** correct (53%)

query_of_death

4pt	Not attempted **483/262 users** correct (184%)
29pt	Not attempted **230/377 users** correct (61%)

Top Scores

mk.al13n	100
semiexp.	100
qwerty787788	100
EgorKulikov	100
ikatanic	100
ecnerwala	100
Golovanov399	100
fagu	100
eatmore	100
Errichto.rekt	100

Problem E. query_of_death

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 4 points *2 minute timeout*	The contest is finished.
large 29 points *10 minute timeout*	The contest is finished.

Problem

Query of Death

We planned a nice simple warm-up DCJ problem for you this year: find the sum of many values. You can call a `GetLength()` function to get the number of values and a `GetValue(i)` function to get the i-th of those values; to make it even easier, each of those values is either 0 or 1. Simple, right? Unfortunately, we have been having a technical difficulty, and now the contest is starting and it is too late to fix it.

The issue is that there is exactly one value of i — we are not sure what that value is, but we will call it i_{qod} — that is a "query of death" (a term occasionally used at Google for a query with severe adverse effects) that causes the following malfunction. The first time that GetValue(i_{qod}) is called on a node, the function will return the correct i_{qod}-th value. However, this will cause the GetValue function to "break" on that node. After that, *every* future call to GetValue(i) on that node will return 0 or 1 purely at (pseudo)random, independently of the value of i or of any previous calls. Other nodes are not affected when a node breaks in this way, but the malfunction can still happen in the future: any other node on which you call GetValue(i_{qod}) will also break.

The i_{qod} value that causes the breakage is the same for every node within a test case; it may vary across test cases, though. Nodes do not remain broken across different test cases.

As an example, suppose that we have two unbroken nodes A and B, and two values i_{ok} and i_{qod}. Then the following sequence of calls would produce the following results:

1. GetValue(i_{ok}) on node A: the correct value is returned.
2. GetValue(i_{qod}) on node A: the correct value is returned, but node A breaks.
3. GetValue(i_{ok}) on node B: the correct value is returned.
4. GetValue(i_{ok}) on node A: a random value is returned.
5. GetValue(i_{qod}) on node A: a random value is returned.
6. GetValue(i_{qod}) on node B: the correct value is returned, but node B breaks.
7. GetValue(i_{qod}) on node B: a random value is returned.
8. GetValue(i_{ok}) on node B: a random value is returned.
9. GetValue(i_{qod}) on node A: a random value is returned.
10. GetValue(i_{ok}) on node A: a random value is returned.

We apologize for the inconvenience, but can you find the sum anyway?

Input

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

- **GetLength()**:
  - Takes no argument.
  - Returns a 64-bit integer: the total number of values to be summed up. (This function still works correctly even on a broken node.)
  - Expect each call to take 0.2 microseconds.
- **GetValue(i)**:
  - Takes a 64-bit number in the range $0 \le$ `i` $<$ `GetLength()`.
  - Returns a 32-bit number (which is always either 0 or 1): the `i`-th value if the node is not broken, or 0 or 1 at (pseudo)random if the node is broken.
  - Expect each call to take 0.2 microseconds.

Output

Output a single line with one integer: the sum of all of the values.

Limits

Time limit: 2 seconds.
Memory limit per node: 128 MB.
Maximum number of messages a single node can send: 1000.
Maximum total size of messages a single node can send: 8 MB.
There is exactly one i_{qod} value, which is the same for each node, and it is within the allowed range for GetLength().
0 ≤ GetValue(i) ≤ 1, for all i.

Small dataset

Number of nodes: 10.
1 ≤ GetLength() ≤ 10^4.

Large dataset

Number of nodes: 100.
1 ≤ GetLength() ≤ 10^8.

Sample

Input	Output

`See input files below.`	`For sample input 1:`
	`2`
	`For sample input 2:`
	`3`
	`For sample input 3:`
	`3`

The code for the samples simulates the node-breaking behavior described in the statement; the actual test cases have the specified behavior, but the implementation (e.g., of randomness on a broken node) is not necessarily the same.

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