Problem D. median

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 10 points *2 minute timeout*	The contest is finished.
large 19 points *10 minute timeout*	The contest is finished.

Problem

Median

In this year's online rounds, we ran into trouble with queries of death and broken memory, so to make it up to you, we decided to put a very easy median-finding problem in the Finals. We were determined to avoid any more issues with peach smoothie spills, so we asked our DCJ (Data Curating Janitor) to set up two special arrays of 100 nodes for this problem, one for each of the Small and Large datasets.

Our DCJ prepared each array according to our instruction:

1. Choose N pieces of data to store. Since the problem is about finding a median, each of the N pieces of data is a non-negative integer. (The same number might appear more than once in the data.) N is also guaranteed to be odd, because who likes finding the median of an even number of integers?
2. A median-finding problem would be far too easy if the data were given in order, so, choose one of the N! permutations of the data uniformly at (pseudo-)random.
3. Write these N pieces of data, in the order of the chosen permutation, clockwise around the edge of a circular disk, starting from a *fixed point* on the very bottom.
4. Make one exact copy of that disk for each node to use.
5. A contestant can call GetData(i) on a node. Each disk has a data-reading needle that starts at the same fixed point. That node will move its data-reading needle to read and return the data i places away (clockwise) from the fixed point, and then the data-reading needle will go back to the fixed point. Since the disk is a circle, it is allowed for i to be greater than or equal to N; the node will return the (i % N)th piece of data. For example, for N = 5, calls to GetData(0) and GetData(5) would return the same piece of data.
6. A contestant can also call GetN() to learn the value of N.

Our DCJ (Data Curating Janitor) completed these tasks the night before the Finals, and, to celebrate, we threw a party and brought in a DCJ (Disc Controlling Jockey) to play some music. Unfortunately, this morning, we learned that the Disc Controlling Jockey mistook our array of nodes for the Large dataset for a very elaborate turntable, and might have spun any or all of the disks! This means that, for the Large dataset only, even though all the nodes' disks have the same data in the same clockwise order, *their fixed points might no longer be the same*. For example, a piece of data that one node thinks is the 0th might be the 3rd on another node!

Moreover, we forgot to ban flavors of smoothie other than peach, and our Disc Controlling Jockey also spilled a strawberry smoothie all over the array of nodes for the Large dataset. Because of this, for the Large dataset only, the GetN() function no longer returns any useful data.

The Finals have already started, and we do not have time to fix the system. Can you find the median of the data anyway?

Input

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

- **GetN()**:
  - Takes no argument.
  - Returns a 32-bit integer.
  - Expect each call to take 0.1 microseconds.
- **GetData(i)**:
  - Takes exactly one 64-bit integer argument: an index i, $0 \le i \le 10^{18}$.
  - Returns a 32-bit integer: the number that is i places away clockwise from the node's fixed point, as described in the problem statement.
  - Expect each call to take 0.1 microseconds.

The data is generated as follows: values for N, a data set of N integers X_0, X_1, ..., X_{N-1}, and 100 "delta" integers d_0, d_1, ..., d_{99} are chosen by the test setter. Then, a permutation P of the integers 0 through N-1 is chosen uniformly at (pseudo)-random*; all nodes use the same values of N, Xs, ds and P.
GetN() returns N in the Small dataset and -1 in the Large dataset, on all nodes.
GetValue(i) on node j returns $X_{P[(i + d_j) \% N]}$.

*: for technical reasons, the randomness used is weaker than something like `std::random_shuffle` in C++ or `java.util.Collections.shuffle` in Java. For transparency, this is the exact procedure used to retrieve the i-th (0-based) element out of N of a random permutation (you do not necesarilly need to fully understand this to solve the problem):

```
int index = i;
do {
   int MASK = (power == 32) ? ~0 : ((1 << power) - 1);
   index += add;
   index *= m1;
   index &= MASK;
   index ^= index >> ((power / 2) + 1);
   index += add2;
   index *= m2;
   index &= MASK;
   index ^= index >> (2 * power / 3);
} while (index >= N);
return index;
```

where `power` is ceil($\log_2$(N)), and `add`, `add2`, `m1` and `m2` are distinct constants between 0 and N - 1, inclusive, such that `m1` and `m2` are odd.

Output

Output one line with a single integer: the median of the data.

Limits

Number of nodes: 100 **(for both the Small and Large datasets)**.
Time limit: 7 seconds.
Memory limit per node: 32 MB. **(Notice this is less than usual.)**
Maximum number of messages a single node can send: 5000.
Maximum total size of messages a single node can send: 8 MB.
$1 \le X_i \le 10^9$, for all i.
N % 2 = 1. (N is odd.)
$1 \le N < 10^9$

Small dataset

$d_i = 0$, for all i. (The nodes all indexed relative to the same original fixed point.)

Large dataset

$0 \le d_i < N$, for all i.

Sample

Input	Output
See input files below.	For sample input 1: 6 For sample input 2: 1000000000 For sample input 3: 1

The sample input has access to GetN() for all cases, but contains values of d_i other than 0. To test the Small dataset, you can edit the files to have all d_i equal to 0 (the answer is of course the same). To test the Large, you can just not use the GetN() function.
Sample input libraries:
Sample input for test 1: median.h [CPP] median.java [Java]
Sample input for test 2: median.h [CPP] median.java [Java]
Sample input for test 3: median.h [CPP] median.java [Java]


Powered by
Google Cloud Platform