Submissions

Testrun

0pt Not attempted **0/74 users** correct (0%)

again

1pt Not attempted **401/409 users** correct (98%)

14pt Not attempted **368/399 users** correct (92%)

lisp_plus_plus

3pt Not attempted **390/399 users** correct (98%)

17pt Not attempted **355/385 users** correct (92%)

asteroids

5pt Not attempted **283/305 users** correct (93%)

25pt Not attempted **91/170 users** correct (54%)

gas_stations

8pt Not attempted **191/233 users** correct (82%)

27pt Not attempted **28/95 users** correct (29%)

Top Scores

eatmore	100
Marcin.Smulewicz	100
tozangezan	100
Errichto.rekt	100
mnbvmar	100
qwerty787788	100
sevenkplus	100
tczajka	100
fhlasek	100
wata	100

Problem B. again

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 1 points *2 minute timeout*	The contest is finished.
large 14 points *10 minute timeout*	The contest is finished.

Problem

... we did it again

You know the issue: just as in the Oops problem from Distributed Round 1, we have lost our problem statement and the correct solutions, and we only have these two correct but slow solutions, one per supported language. Once again, we still have our test data. Can you still solve the problem?

Notice that in this problem 20 nodes are used to run both the Small and the Large datasets, which is not the usual number for Distributed Code Jam problems. 20 nodes were also used to run the solutions and produce the answers for the examples.

The C++ solution:

```
#include <message.h>
#include <stdio.h>
#include "again.h"

#define MASTER_NODE 0
#define SENDING_DONE -1
#define LARGE_PRIME 1000000007

int main() {
  if (MyNodeId() == MASTER_NODE) {
    long long result = 0;
    for (int node = 1; node < NumberOfNodes(); ++node) {
      while (true) {
        Receive(node);
        long long value = GetLL(node);
        if (value == SENDING_DONE) {
          break;
        } else {
          result = (result + value) % LARGE_PRIME;
        }
      }
    }
    printf("%lld\n", result);
    return 0;
  } else {
    for (long long i = 0; i < GetN(); ++i) {
      for (long long j = 0; j < GetN(); ++j) {
        long long value = GetA(i) * GetB(j);
        if ((i + j) % NumberOfNodes() == MyNodeId()) {
          PutLL(MASTER_NODE, value);
          Send(MASTER_NODE);
        }
      }
    }
    PutLL(MASTER_NODE, SENDING_DONE);
    Send(MASTER_NODE);
  }
  return 0;
}
```

The Java solution:

```
public class Main {
  static int MASTER_NODE = 0;
  static long SENDING_DONE = -1;
  static long LARGE_PRIME = 1000000007;

  public static void main(String[] args) {
    if (message.MyNodeId() == MASTER_NODE) {
      long result = 0;
      for (int node = 1; node < message.NumberOfNodes(); ++node
        while (true) {
          message.Receive(node);
          long value = message.GetLL(node);
          if (value == SENDING_DONE) {
            break;
          } else {
            result = (result + value) % LARGE_PRIME;
          }
        }
      }
      System.out.println(result);
    } else {
      for (long i = 0; i < again.GetN(); ++i) {
        for (long j = 0; j < again.GetN(); ++j) {
          long value = again.GetA(i) * again.GetB(j);
          if ((i + j) % message.NumberOfNodes() == message.MyNo
            message.PutLL(MASTER_NODE, value);
            message.Send(MASTER_NODE);
          }
        }
      }
      message.PutLL(MASTER_NODE, SENDING_DONE);
      message.Send(MASTER_NODE);
    }
  }
}
```

Input

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

- **GetN()**:
  - Takes no argument.
  - Returns a 64-bit number.
  - Expect each call to take 0.05 microseconds.
- **GetA(i)**:
  - Takes a 64-bit number in the range 0 ≤ i < GetN().
  - Returns a 64-bit number.
  - Expect each call to take 0.05 microseconds.
- **GetB(i)**:
  - Takes a 64-bit number in the range 0 ≤ i < GetN().
  - Returns a 64-bit number.
  - Expect each call to take 0.05 microseconds.

Output

Output what either of the solutions above would output, if they ran on 20 nodes without any limits on memory, time, number of messages or total size of messages.

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.
Number of nodes: 20.
$0 \le \text{GetA}(i) \le 10^9$, for all i.
$0 \le \text{GetB}(i) \le 10^9$, for all i.

Small input

$1 \le \text{GetN}() \le 30{,}000$.

Large input

$1 \le \text{GetN}() \le 10^8$.

Sample

Input	Output
See input files below.	For sample input 1: 999979007 For sample input 2: 2390 For sample input 3: 0

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