Distributed Round 1 2016

Problem A. Testrun

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
0 points
2 minute timeout

The contest is finished.

Problem

This is a way to test your solutions, not a real problem!

When you submit a solution to this problem, it will run one testcase on a 100 nodes. This will allow you to estimate how fast your solution will run on our system.

Remember to change your solution appropriately before submitting it for real, so you don't fail because of a compilation error! The best way to check is to run your solution on the small input before submitting to the large input.

Input

There is no input for this problem. This means you should not include / import an input library.

Output

Doesn't really matter what you output. If your solution runs successfully to completion, it will be judged as "Wrong Answer".

Limits

Each node will have access to 1 GB of RAM, and a time limit of 26 seconds. The maximum number of messages a single node can send is 5000, and the maximum sum of the sizes of those messages is 8MB.
This problem only has one small test case. It will run on 100 nodes.

Submissions

Testrun

0pt	Not attempted **0/422 users** correct (0%)

oops

2pt	Not attempted **893/925 users** correct (97%)
12pt	Not attempted **756/882 users** correct (86%)

rps

1pt	Not attempted **789/857 users** correct (92%)
15pt	Not attempted **585/783 users** correct (75%)

crates

8pt	Not attempted **557/673 users** correct (83%)
25pt	Not attempted **258/433 users** correct (60%)

winning_move

3pt	Not attempted **635/700 users** correct (91%)
34pt	Not attempted **49/309 users** correct (16%)

Top Scores

simonlindholm	100
tomconerly	100
eatmore	100
cgy4ever	100
bmerry	100
Simon.M	100
Klockan	100
tczajka	100
tkociumaka	100
Zlobober	100

Submissions

Testrun
0pt Not attempted **0/422 users** correct (0%)

oops
2pt Not attempted **893/925 users** correct (97%)
12pt Not attempted **756/882 users** correct (86%)

rps
1pt Not attempted **789/857 users** correct (92%)
15pt Not attempted **585/783 users** correct (75%)

crates
8pt Not attempted **557/673 users** correct (83%)
25pt Not attempted **258/433 users** correct (60%)

winning_move
3pt Not attempted **635/700 users** correct (91%)
34pt Not attempted **49/309 users** correct (16%)

Top Scores

simonlindholm	100
tomconerly	100
eatmore	100
cgy4ever	100
bmerry	100
Simon.M	100
Klockan	100
tczajka	100
tkociumaka	100
Zlobober	100

Problem B. oops

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 2 points *2 minute timeout*	The contest is finished.
large 12 points *10 minute timeout*	The contest is finished.

Problem

Oops

The team preparing the Distributed Code Jam made a mess and needs your help. The statement and solutions for this problem were lost minutes before the contest, and all that we could recover were these two correct but far too slow (and misguided) solutions, one per language. Fortunately, we still have the test data. Can you reconstruct the statement and solve the problem properly based on the recovered slow solutions?

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 "oops.h"

#define MASTER_NODE 7
#define DONE -1

int main() {
  long long N = GetN();
  long long nodes = NumberOfNodes();
  long long my_id = MyNodeId();
  long long best_so_far = 0LL;
  for (long long i = 0; i < N; ++i) {
    for (long long j = 0; j < N; ++j) {
      if (j % nodes == my_id) {
        long long candidate = GetNumber(i) - GetNumber(j);
        if (candidate > best_so_far) {
          best_so_far = candidate;
          PutLL(MASTER_NODE, candidate);
          Send(MASTER_NODE);
        }
      }
    }
  }
  PutLL(MASTER_NODE, DONE);
  Send(MASTER_NODE);

  if (my_id == MASTER_NODE) {
    long long global_best_so_far = 0;
    for (int node = 0; node < nodes; ++node) {
      long long received_candidate = 0;
      while (true) {
        Receive(node);
        received_candidate = GetLL(node);
        if (received_candidate == DONE) {
          break;
        }
        if (received_candidate > global_best_so_far) {
          global_best_so_far = received_candidate;
        }
      }
    }
    printf("%lld\n", global_best_so_far);
  }
  return 0;
}
```

The Java solution:

```
public class Main {
  static int MASTER_NODE = 7;
  static int DONE = -1;
  public static void main(String[] args) {
    long N = oops.GetN();
    long nodes = message.NumberOfNodes();
    long my_id = message.MyNodeId();
    long best_so_far = 0L;
    for (long i = 0; i < N; ++i) {
      for (long j = 0; j < N; ++j) {
        if (j % nodes == my_id) {
          long candidate = oops.GetNumber(i) - oops.GetNumber(j
          if (candidate > best_so_far) {
            best_so_far = candidate;
            message.PutLL(MASTER_NODE, candidate);
            message.Send(MASTER_NODE);
          }
        }
      }
    }
    message.PutLL(MASTER_NODE, DONE);
    message.Send(MASTER_NODE);

    if (my_id == MASTER_NODE) {
      long global_best_so_far = 0;
      for (int node = 0; node < nodes; ++node) {
        long received_candidate = 0;
        while (true) {
          message.Receive(node);
          received_candidate = message.GetLL(node);
          if (received_candidate == DONE) {
            break;
          }
          if (received_candidate > global_best_so_far) {
            global_best_so_far = received_candidate;
          }
        }
      }
      System.out.println(global_best_so_far);
    }
  }
}
```

Input

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

- **GetN()**:
  - Takes no argument.
  - Returns a 64-bit number.
  - Expect each call to take 0.05 microseconds.
- **GetNumber(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: 6 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.
$-10^{18} \leq$ GetNumber(i) $\leq 10^{18}$, for all i.

Small dataset

1 ≤ GetN() ≤ 30,000.

Large dataset

$1 \leq$ GetN() $\leq 10^9$.

Sample

```
Input                      Output

See input files below.     For sample input 1:
                           93
                           For sample input 2:
                           0
                           For sample input 3:
                           14
```

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

Submissions

Testrun

0pt	Not attempted **0/422 users** correct (0%)

oops

2pt	Not attempted **893/925 users** correct (97%)
12pt	Not attempted **756/882 users** correct (86%)

rps

1pt	Not attempted **789/857 users** correct (92%)
15pt	Not attempted **585/783 users** correct (75%)

crates

8pt	Not attempted **557/673 users** correct (83%)
25pt	Not attempted **258/433 users** correct (60%)

winning_move

3pt	Not attempted **635/700 users** correct (91%)
34pt	Not attempted **49/309 users** correct (16%)

Top Scores

simonlindholm	100
tomconerly	100
eatmore	100
cgy4ever	100
bmerry	100
Simon.M	100
Klockan	100
tczajka	100
tkociumaka	100
Zlobober	100

Problem C. rps

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 15 points *10 minute timeout*	The contest is finished.

Problem

Remarkably Parallel Scenario

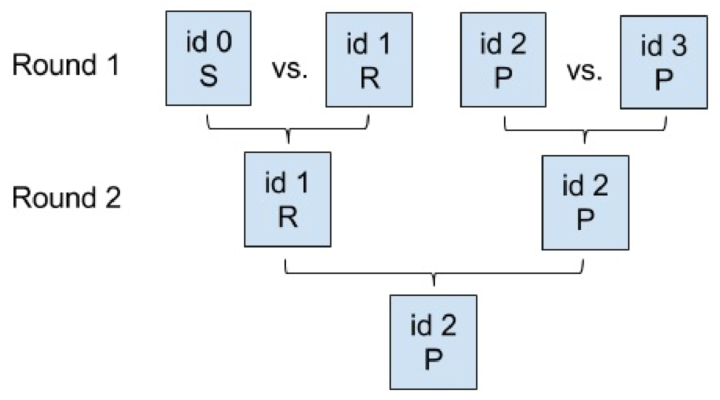
Your Rock-Paper-Scissors tournament yesterday went so well that you've been asked to organize another one. It will be a single-elimination tournament with **N** rounds. $2^{\mathbf{N}}$ players will participate, and they will have unique ID numbers in the range 0 through $2^{\mathbf{N}}$-1, inclusive.

Initially, the players will be lined up from left to right, in increasing order by ID number. In each round, the first and second players in the lineup (starting from the left) will play a match against each other, and the third and fourth players in the lineup (if they exist) will play a match against each other, and so on; all of these matches will occur simultaneously. The winners of these matches will remain in the lineup, in the same relative order, and the losers will leave the lineup and go home. Then a new round will begin. This will continue until only one player remains in the lineup; that player will be declared the winner.

In each Rock-Paper-Scissors match, each of the two players secretly chooses one of *Rock*, *Paper*, or *Scissors*, and then they compare their choices. Rock beats Scissors, Scissors beats Paper, and Paper beats Rock. If one player's choice beats the other player's choice, then that player wins and the match is over. You are tired of worrying about ties, so you have decided that if the players make the same choice, the player on the left wins the match.

You know that the players this year are not very strategic, and each one has a preferred move and will only ever play that move. Fortunately, you know every player's preferred move, so you can figure out: what is the ID number of the player who will win the tournament?

Here's an example tournament with **N** = 2:



In Round 1, player 1 beats player 0 (since Rock always beats Scissors), and player 2 beats player 3 (since the player with the lower number wins a tie). In Round 2, player 2 beats player 1 (since Paper always beats Rock). So, player 2 is the winner.

Input

The input library will be called "rps"; see the sample inputs below for examples in your language. It defines two methods: GetN(), which returns the number **N** of rounds in the tournament, and GetFavoriteMove(id), which returns the favorite move of the player with ID number id, for 0 ≤ id < $2^{\text{GetN()}}$. This move will always be either R, P, or S, representing Rock, Paper, or Scissors, respectively.

- **GetN()**:
    - Takes no argument.
    - Returns a 64-bit number: the **N** value from the statement.

    - Expect each call to take 0.09 microseconds.
- **GetFavoriteMove(id)**:
    - Takes a 64-bit number in the range $0 \leq i < 2^{\mathrm{GetN()}}$.
    - Returns a character.
    - Expect each call to take 0.09 microseconds.

Output

Output one value: the ID number of the winning player.

Limits

Time limit: 3 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.
GetFavoriteMove(id) is always one of R, P, or S for all valid values of id.

Small dataset

Number of nodes: 10.
1 ≤ GetN() ≤ 10.

Large dataset

Number of nodes: 100.
1 ≤ GetN() ≤ 28.

Sample

Input	Output

`See input files below.`	`For sample input 1:`
	`5`
	`For sample input 2:`
	`0`
	`For sample input 3:`
	`2`

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

Submissions

Testrun
0pt Not attempted **0/422 users** correct (0%)

oops
2pt Not attempted **893/925 users** correct (97%)
12pt Not attempted **756/882 users** correct (86%)

rps
1pt Not attempted **789/857 users** correct (92%)
15pt Not attempted **585/783 users** correct (75%)

crates
8pt Not attempted **557/673 users** correct (83%)
25pt Not attempted **258/433 users** correct (60%)

winning_move
3pt Not attempted **635/700 users** correct (91%)
34pt Not attempted **49/309 users** correct (16%)

Top Scores

simonlindholm	100
tomconerly	100
eatmore	100
cgy4ever	100
bmerry	100
Simon.M	100
Klockan	100
tczajka	100
tkociumaka	100
Zlobober	100

Problem D. crates

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

Problem

Rearranging Crates

You are the manager of the warehouse of the largest wharf in the area. The warehouse is tall and long, but narrow, so you have created a single row of stacks of crates. Unfortunately, the ship unloaders are usually sloppy and in a rush, so the number of crates can vary wildly between stacks, making them look like an uneven skyline of buildings.

The District Crate Judge (DCJ) is visiting the facilities tomorrow, and you want to get a perfect score in her assessment. You decided to use an old crane to rearrange the crates to make them into a neatly organized crate wall that will impress the DCJ, but this task may take a long time! Your crane can only handle one crate at a time. Moreover, the crane's wheels are not strong enough to move if the crane is handling a crate, so the crane can only drop off a crate on a stack adjacent to the one the crate came from.

The crane starts off at the leftmost stack, without any crates. The crane can carry out only the following sequence of actions, which we will call a *move*:

1. Position the crane at any stack with at least one crate. This may be the stack the crane is already at.
2. Pick up the top crate from the stack the crane is at. (Even if this causes the stack to have zero crates, it still counts as a stack.)
3. Put that crate on top of a stack directly adjacent to the stack the crane is at.

Notice that if you want to move a crate two stacks down to the right, for example, this will take two moves.

You must use some number of moves to transfer crates around so that all stacks are as even as possible, with all the excess distributed among the leftmost possible stacks. If there are N stacks and a total of C crates, you want the $C \% N$ leftmost stacks to have $\lceil C/N \rceil$ crates each and the rest of the stacks to have $\lfloor C/N \rfloor$ crates each. For instance, if there are 3 stacks and a total of 8 crates, you want the stack heights to be 3, 3, 2, from left to right.

What is the minimum number of moves you need to use to balance the stacks as specified above? Since the result can be really big, output it modulo 10^9+7 (1000000007).

Input

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

- **GetNumStacks()**:
  - Takes no argument.
  - Returns a 64-bit number: the number of stacks.
  - Expect each call to take 0.12 microseconds.
- **GetStackHeight(i)**:
  - Takes a 64-bit number in the range 1 ≤ i ≤ GetNumStacks().
  - Returns a 64-bit number: the starting number of crates in stack i, counting from left to right and starting from 1.
  - Expect each call to take 0.12 microseconds.

Output

Output a single line with a single integer, the remainder of dividing the minimum number of moves needed to balance the stacks by 10^9+7 (1000000007).

Limits

Time limit: 6 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.

1 ≤ GetStackHeight(i) ≤ 10^9, for all i.

Small dataset

Number of nodes: 10.
1 ≤ GetNumStacks() ≤ 10^6.

Large dataset

Number of nodes: 100.
1 ≤ GetNumStacks() ≤ 10^9.

Sample

Input	Output

`See input files below.`	`For sample input 1:`
	`3`
	`For sample input 2:`
	`5`
	`For sample input 3:`
	`0`

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

Submissions

Testrun

0pt	Not attempted **0/422 users** correct (0%)

oops

2pt	Not attempted **893/925 users** correct (97%)
12pt	Not attempted **756/882 users** correct (86%)

rps

1pt	Not attempted **789/857 users** correct (92%)
15pt	Not attempted **585/783 users** correct (75%)

crates

8pt	Not attempted **557/673 users** correct (83%)
25pt	Not attempted **258/433 users** correct (60%)

winning_move

3pt	Not attempted **635/700 users** correct (91%)
34pt	Not attempted **49/309 users** correct (16%)

Top Scores

simonlindholm	100
tomconerly	100
eatmore	100
cgy4ever	100
bmerry	100
Simon.M	100
Klockan	100
tczajka	100
tkociumaka	100
Zlobober	100

Problem E. winning_move

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 34 points *10 minute timeout*	The contest is finished.

Problem

The Only Winning Move

Perhaps you have played this game before: Every player submits a positive integer. The winner, if any, is the player who submitted the smallest positive integer that was submitted by no other player.

Your friends invited you to play this game, but you decided that game theory is complex and the only winning move is not to play. Instead, you volunteered to judge the game. Given the players' choices, can you determine what the winning number was?

Input

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

- **GetNumPlayers()**:
  - Takes no argument.
  - Returns a 64-bit number: the number of players in the game.
  - Expect each call to take 0.12 microseconds.
- **GetSubmission(playernum)**:
  - Takes a 64-bit number in the range 0 ≤ playernum < GetNumPlayers().
  - Returns a 64-bit number: the number chosen by the player with ID number playernum.
  - Expect each call to take 0.12 microseconds.

Output

Output one value: the winning number, or 0 if there was no winner.

Limits

Time limit: 4 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: 1 GB.
1 ≤ GetSubmission(playernum) ≤ 10^{18}, for all playernum.

Small dataset

Number of nodes: 10.
1 ≤ GetNumPlayers() ≤ 1000.

Large dataset

Number of nodes: 100.
1 ≤ GetNumPlayers() ≤ 35000000.

Sample

```
Input                    Output

See input files below.   For sample input 1:
                         4
                         For sample input 2:
                         3
                         For sample input 3:
                         0
```

Sample input libraries:
Sample input for test 1: winning_move.h [CPP] winning_move.java [Java]

Sample input for test 2: winning_move.h [CPP] winning_move.java [Java]
Sample input for test 3: winning_move.h [CPP] winning_move.java [Java]