Distributed Finals 2015

Submissions

Testrun
0pt | Not attempted **0/6 users** correct (0%)

kolakoski
8pt | Not attempted **5/7 users** correct (71%)
17pt | Not attempted **2/5 users** correct (40%)

necklace
16pt | Not attempted **10/10 users** correct (100%)
29pt | Not attempted **9/10 users** correct (90%)

rocks
7pt | Not attempted **2/2 users** correct (100%)
53pt | Not attempted **0/1 users** correct (0%)

shipping
26pt | Not attempted **2/6 users** correct (33%)
44pt | Not attempted **0/1 users** correct (0%)

Top Scores

bmerry	103
Marcin.Smulewicz	71
shik	70
MiSawa	60
ZbanIlya	53
WJMZBMR	45
simonlindholm	45
mk.al13n	45
wan92hy	45
dreamoon	24

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.

Distributed Finals 2015

A. Testrun
B. kolakoski
C. necklace
D. rocks
E. shipping

Contest Analysis
Questions asked 2

Submissions

Testrun
0pt | Not attempted
0/6 users correct (0%)

kolakoski
8pt | Not attempted
5/7 users correct (71%)
17pt | Not attempted
2/5 users correct (40%)

necklace
16pt | Not attempted
10/10 users correct (100%)
29pt | Not attempted
9/10 users correct (90%)

rocks
7pt | Not attempted
2/2 users correct (100%)
53pt | Not attempted
0/1 users correct (0%)

shipping
26pt | Not attempted
2/6 users correct (33%)
44pt | Not attempted
0/1 users correct (0%)

Top Scores

bmerry	103
Marcin.Smulewicz	71
shik	70
MiSawa	60
ZbanIlya	53
WJMZBMR	45
simonlindholm	45
mk.al13n	45
wan92hy	45
dreamoon	24

Problem B. kolakoski

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

Problem

The Kolakoski sequence is defined as follows, where A(i) is the i-th term in the sequence:

- A(0) = 1
- A(1) = 2
- The sequence is composed entirely of alternating runs of 1's and 2's
- A(i) is the length of the i-th run.

This completely and uniquely defines the sequence.

The first twenty terms of the sequence are as follows, where the lines mark the alternating runs of 1's and 2's:

1 2 2 1 1 2 1 2 2 1 2 2 1 1 2 1 1 2 2 1

1 2 2 1 1 2 1 2 2 1 2 1 1 2 1 2 2 1

By collecting the lengths of each run, we obtain the same sequence again.

You become mystified contemplating the elegance of the Kolakoski sequence and after staring at its 1's and 2's for far too long, you begin to wonder if maybe you should spice it up a little and introduce some more numerical variety to the terms.

So you decide to assign an arbitrary coefficient to each index in a manner such as the following:

C(0)=1
C(1)=3
C(2)=1
C(3)=5
C(4)=2
C(5)=2

By multiplying the first 6 terms each by their coefficient and summing, we get

1*1 + 3*2 + 1*2 + 5*1 + 2*1 + 2*2 = 20.

Given a mapping from index to coefficient, find the dot product of the first **N** terms of the Kolakoski sequence and their respective coefficients.

Input

The library "kolakoski" will contain two functions:

- GetIndex() which returns **N**, the number of terms we wish to sum; and
- GetMultiplier(i) which takes an index i and returns the coefficient (a number from 0 to 50) for that index.

A single call to GetMultiplier will take approximately 0.005 microseconds.

Output

Output one number: the weighted sum of the elements of the Kolakoski sequence.

Limits

Each node will have access to 700MB of RAM.
Your solution will run on 100 nodes in both inputs.

Small input

GetMultiplier(i) will always return 1, for all the inputs.
$1 \le \text{GetIndex}() \le 10^9$
Each node will have a time limit of 10 seconds.

Large input

$1 \le$ GetMultiplier(i) ≤ 50 for all i
$1 \le$ GetIndex() $\le 3 \times 10^9$
Each node will have a time limit of 12 seconds.

Sample

Input	Output
See below for sample input files.	For sample input 1: 1 For sample input 2: 15 For sample input 3: 50

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

Problem C. necklace

small 16 points *2 minute timeout*	The contest is finished.
large 29 points *10 minute timeout*	The contest is finished.

Problem

You've come up with the coolest idea ever for a new fashion trend: customizable necklaces made out of strings with beads that display letters and other characters! The beads appear only on the front of the necklace and read only in one direction, so the string of characters is not circular and irreversible. By itself, this is not really a new idea. The awesome new feature you have in mind is to add a button that lights up some of the beads so that they display a secret message consisting of characters that form a subsequence of the main string of characters. This will have so many applications... just think of the possibilities! And it's so shiny! People are going to love it! Everyone will want their own!

So you announce this product, allowing people to place orders for necklaces by specifying the string of characters to be displayed on the necklace as well as the secret message to be lit up when they press the button. The orders come pouring in! Your idea is even more popular than you expected! How exciting!

Unfortunately, after examining a few orders, you realize that you forgot to check the crucial constraint that the secret message has to be a subsequence of the main necklace string. Without that, the secret message can't always be lit up entirely.

You don't want to disappoint your customers by just telling them that it is impossible to light up their secret messages in the chosen necklace strings. So you decide to offer them an alternative message by finding a substring of their secret message that forms a subsequence of their necklace string, in case they would be satisfied with this shorter version. You want to maximize the length of such a substring.

Given a necklace string **N** and a secret message string **M**, find the maximum length of a substring of **M** that is also a subsequence of **N**.

Input

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

- GetNecklaceLength(), which returns the length of the necklace string
- GetNecklaceElement(i), which returns the i-th (0-indexed) element of necklace string
- GetMessageLength(), which returns the length of the secret message
- GetMessageElement(i), which returns the i-th (0-indexed) element of the secret message.

A single call of GetNecklaceElement or GetMessageElement will take up to 0.02 microseconds.

Output

Output one integer - the maximum length of a substring of Message that is also a subsequence of Necklace.

Limits

$0 \le$ GetNecklaceElement(i), GetMessageElement(i) $\le 10{,}000$
$1 \le$ GetNecklaceLength() $\le 10^9$
Each node will have access to 256MB of RAM and a time limit of 5 seconds.
Your solution will run on 100 nodes (both for the small and the large input).

Small input

$1 \le$ GetMessageLength() ≤ 100

Large input

$1 \le$ GetMessageLength() ≤ 3000

Sample

Input	Output
See below for sample input files.	For sample input 1: 3 For sample input 2: 1 For sample input 3: 4

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

Problem D. rocks

small 7 points *2 minute timeout*	The contest is finished.
large 53 points *10 minute timeout*	The contest is finished.

Problem

You own an almond farm in a region which for the last few years has been experiencing an extreme, record-breaking drought. Reservoirs are drying up, mandatory water rationing is in effect, and you are facing pressure to do something about this farm of yours that is consuming 4 liters of water per almond.

As you survey the regular grid-like arrangement of your farm, a brilliant idea strikes you. You should turn the land into a giant board where people can play life-sized versions of their favorite grid-based board games! So you take out all the almond trees and turn your farm into an amusement park by sectioning your land into a square **N**-by-**N** grid.

Unfortunately, the region you are in is also known for its earthquakes, and soon after you convert your land, an earthquake strikes, triggering a landslide that drops giant rocks onto your grid, each rock conveniently occupying the space of exactly one of the cells. No cell contains more than one rock.

You are now trapped in the bottom left corner (the cell with coordinates (0, 0)), and you need to get to the exit at the top right corner (the cell with coordinates (**N**-1, **N**-1)). You can move only up or to the right 1 cell at a time and cannot squeeze past any rocks or climb over them, so if you want to move into a cell that is occupied by a rock, you have to push it into next cell in the same direction as you are moving. If there is already a rock in that cell, it will also get pushed in the same direction, and so on, until finally there is a square without rocks. After every push, each rock occupies exactly one cell. You can push up to **K** rocks at the same time in this manner, but you cannot push rocks off the grid. Is it possible to reach the exit?

Input

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

- GetN(), which returns the number of rows of the grid (which is equal to the number of columns);
- GetK(), which returns the maximum number of rocks you can push in front of you; and
- IsRock(x, y), which returns true if there is a rock in the cell with coordinates (x, y).

A single call to IsRock() will take approximately 0.05 microseconds.

Output

Output one line containing the word "YES" if it is possible to reach the exit, or the word "NO" if it is impossible.

Limits

Each node will have access to 600MB of RAM, and a time limit of 4 seconds.
IsRock(0, 0) and IsRock(GetN() - 1, GetN() - 1) will return false.
$0 \le$ GetK() $\le$ GetN()
Your solution will run on 100 nodes in both inputs.

Small input

$2 \le$ GetN() $\le 2{,}000$

Large input

$2 \le$ GetN() $\le 10{,}000$

Sample

Input	Output
See input files below.	For sample input 1: YES

```
                        For sample input 2:
                        NO
                        For sample input 3:
                        NO
```

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

Problem E. shipping

small 26 points *2 minute timeout*	The contest is finished.
large 44 points *10 minute timeout*	The contest is finished.

Problem

In a certain country that shall remain unnamed, people live in **N** villages along river valleys separated by mountain ranges. Due to the geography of the country, it is extremely difficult to cross over mountains. The only practical way to get from one village to another is by following rivers. All of the rivers are part of the same system and each river connects to another river downstream, until eventually all water in the system passes through a single point to enter the sea. There is a village at every point where two rivers join and at each river's source, and there is one village at the mouth of the entire river system, where it meets the sea. There may be also be villages at other points along rivers. No two villages are at the same location along a river, and there is exactly one path between any two villages. In other words, the villages and the paths between them form a tree.

You are in charge of managing a shipping company that delivers packages between villages by transporting them on boats along the rivers. Since there is only one path between any two villages, you fortunately do not have to worry about finding the shortest path to use. Unfortunately, however, the country is in the middle of a civil war, with **K** rival factions battling for control. Each village is under the control of exactly one of the factions. Luckily, conditions are stable at the moment and no village is going to change factions anytime soon.

As a neutral company, you are able to send your boats through every village, but only under the condition that each of the factions is allowed to use your shipping services for free, by loading additional packages onto any of your boats that pass between villages under the control of that faction. Every time one of your boats passes through a village (including the village at which your boat begins its route), you may be given an additional package occupying 1 unit of capacity on your boat, to be transported to another village further along on your boat's current route that is occupied by the same faction. Because the factions do not want their packages to be intercepted by other factions, you will be given a package to transport between two villages only if every village on the path between them (including the destination) is occupied by the same faction. Once you deliver a package, you can reuse the space for another package later on, but multiple packages at the same time require multiple units of capacity.

You are now faced with the problem of guaranteeing enough extra capacity on your boats to transport these extra packages. You have **Q** shipments, each of which has a source village and a destination village. You will use a different boat for each shipment. For each shipment, determine the number of units of capacity to reserve on your boat in order to carry all of the additional packages for the various factions in the worst case.

Input

The input library is called "shipping"; see the sample inputs below for examples in your language. It defines 6 methods:

- NumberOfVillages()
- VillageFaction(village_index)
- VillageImmediatelyDownstream(village_index)
- NumberOfShipments()
- GetShipmentSource(shipment_id)
- GetShipmentDestination(shipment_id)

Villages and shipments are both zero-indexed. VillageImmediatelyDownstream returns the index of the village that lies immediately downstream, except for the village at the mouth of the river system, whose return value is the index of this village (since no other village is downstream, just the sea).
A single call of VillageImmediatelyDownstream will take approximately 0.04 microseconds. A single call of VillageFaction will take approximately 0.02 microseconds.

Output

Output a space-separated list of integers, where the i-th integer is the minimum number of units of capacity needed for the boat delivering the i-th shipment to carry all additional packages in the worst case.

Limits

Each node will have access to 256MB of RAM, and a time limit of 5 seconds.
$1 \le$ NumberOfVillages() $\le 10^8$
$0 \le$ VillageFaction(i) $\le 10^6$ for all villages
$0 \le$ GetShipmentSource(i), GetShipmentDestination(i) $<$ NumberOfVillages() for all shipments
The VillageImmediatelyDownstream method will describe a tree (that is, there will be only one path between any two villages).
Your solution will run on 100 nodes in both inputs.

Small Input

$1 \le$ NumberOfShipments() ≤ 10
VillageImmediatelyDownstream(i) $\le$ i for all villages

Large input

$1 \le$ NumberOfShipments() $\le 20{,}000$

Sample

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

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