Astro the astronaut just started his internship on the International Space Station (ISS). His job is to study the Terran civilization from the planet Mar Sara. The Terrans already built `N` cities, numered from `1` to `N`, but they did not build any roads between them.

Just as Astro started his internship, the Terrans began building roads on Mar Sara. It's known that the Terrans want to connect the cities as fast as possible, so they won't build a road between 2 cities that are already connected by a path composed of one or more roads. Each time the Terrans build a new road, Astro wants to know what cities it connects before the next road is built.

Luckily for Astro, he has access to SETI Masterpieces, a new satellite that can look at Mar Sara and see what roads have been built. Given two disjoint sets of cities, SETI is able to tell whether there is at least one **direct** road that connects a city from the first set with another city from the second set.

The way the interaction works is the following:

1. The Terrans build a road.
2. Astro uses SETI to try to discover the newly built road.
3. He goes to the ISS Scientific Committee and shows them his answer for the newly created road.
4. If there are less than `N - 1` roads build, repeat from step 1.

Your job is to write a program that will solve Astro's problem.

Interaction

You should implement a function `run()` which will be called once, in the beginning of the program. From this function, you should call `query()` every time you use the SETI Masterpieces. Every time you determine the newest built road, you should call `setRoad()`. You can call `query()` multiple times before calling `setRoad()`.

Grader function: `query()`

- C/C++: `int query(int size_a, int size_b, int a[], int b[]);`
- Pascal: `function query(size_a, size_b : longint; a, b : array of longint) : longint;`

Call this function to make a query to the SETI Masterpieces. Arguments `size_a` and `size_b` should represent the sizes of the two sets that are queried. Arrays `a[]` and `b[]` should contain the indices of the cities from the first and second set, respectively. Note that these sets should be disjoint! The function returns `1` if there is at least one direct road between a city from the first set and a city from the second set, and `0` otherwise.

Grader procedure: `setRoad()`

- C/C++: `void setRoad(int a, int b);`
- Pascal: `procedure setRoad(a : longint, b : longint);`

Call this procedure to indicate that you discovered that the newest road connects cities `a` and `b`. If the road is incorrectly discovered, you will receive 0 points for the current test case and the program terminates. If this is the `(N-1)`th call to this procedure, the program terminates and returns the score for the current test case. Otherwise, a new road is built by the Terrans and the interaction continues. Any other subsequent call to `query()` will take into account the road built after the call to `setRoad()`.

Your procedure: `run()`

- C/C++: `void run(int N);`
- Pascal: `procedure run(N : longint);`

Your submission must implement this procedure. You must alternately call `query()` and `setRoad()` from this procedure. The received argument `N` represents the number of cities built by the Terrans. If this procedure finishes its execution before guessing all the roads, you will not receive credit for that test case.

Language notes

- C/C++: you must `#include "icc.h"`
- Pascal: you must define the `unit icc`, and you must also import the grader routines with the statement `uses graderhelplib`

Scoring

The test cases for this task are grouped into 6 batches. The tests belonging to the same batch will have the same value for `N`. You will get the score for a batch, if, for each test from the batch, you guess all the roads correctly using at most `M` calls to the function `query()`. The values for `N` and `M`, as well as the score for each batch, are listed in the table below.

Batch number	N = Number of cities	M = Maximum calls to *query()*	Score
1	15	1500	7
2	50	2500	11
3	100	2250	22
4	100	2000	21
5	100	1775	29
6	100	1625	10

Sample execution

Contestant action	Grader action	Note
-	run(4)	- The grader starts a run for N = 4 cities. The first road is built between cities 2 and 4, unknown to the contestant.
query(1, 3, {1}, {2, 3, 4})	return 0	- Query for sets {1} and {2, 3, 4}. The answer is "false": there is no road from 1 to any of the other cities.
query(1, 2, {2}, {3, 4})	return 1	- Query for sets {2} and {3, 4}. The answer is "true": there is a road from city 2 to city 4.
query(1, 1, {2}, {3})	return 0	
setRoad(2, 4)	-	- The contestant guesses correctly road (2, 4). The grader generates a new road between 1 and 3.
query(2, 2, {2, 4}, {1, 3})	return 0	- Query for sets {2, 4} and {1, 3}. The answer is "false".

setRoad(1, 3)	-	- The contestant guesses correctly road (1, 3). The grader generates a new road between 1 and 4.
query(2, 2, {2, 4}, {1, 3})	return 1	- Same query as the last one. The answer is now "true", because of the newly generated road.
query(1, 2, {2}, {1, 3})	return 0	
query(1, 1, {4}, {3})	return 0	
setRoad(4, 1)	exit	- The contestant guesses correctly the last road (4, 1). The grader accepts the last guess and gives full score for the test.