Astro a Nemzetközi Űrállomásról (ISS) vizsgálja a Mar Sara bolygót. Az ott élő Terran civilizáció lakosai N várost építettek. A városok 1-től N-ig sorszámozottak, és kezdetben egyetlen út sincs közöttük.

A Terran lakosai utakat kezdenek építeni a városok között méghozzá úgy, hogy egy lépésben két olyan város közé nem építenek közvetlen utat, amelyek már egy vagy több közvetlen úton keresztül elérhetők. Minden olyan alkalommal, amikor két város között új közvetlen út épül, Astro meg akarja határozni ezt a két várost egy SETI műszer segítségével. Ez a műszer képes megadni városok két diszjunkt halmazára, hogy van-e legalább egy **közvetlen** út, amely összeköt egy várost az első halmazból egy másik várossal a másik halmazból.

Egy út meghatározása a következő lépések szerint történik:

1. A Terránok építenek egy új utat.
2. Astro a SETI segítségével próbálja meghatározni az új utat.
3. Az új utat jelenti az ISS Tudományos Bizottságának, akik jóváhagyják a válaszát.
4. Ha még nincs N-1 közvetlen út építve, akkor az 1. ponttól ismétlődik a folyamat.

Írj olyan programot, amely megoldja Astro problémáját.

**Interakció**

A run() függvényt kell megvalósítanod, amelyet az értékelőprogram hív meg egyszer a program elején. Ebben a függvényben hívhatod a query() függvényt, amely a SETI műszert valósítja meg. A setRoad() függvény hívásával kell jelenteni az új közvetlen utat. A query() függvény többször is hívható a setRoad() függvény meghívása előtt.

**query() könyvtári függvény**

* 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;

A SETI műszer használatához ezt a függvényt kell meghívni. A size\_a és size\_b paraméterek két vizsgált halmaz elemszáma legyen. Az a[] és b[] tömb tartalmazza az első és a második halmazbeli városok sorszámait. Megjegyzendő, hogy a két halmaznak diszjunktnak kell lennie. A függvény 1-et ad vissza eredményül, ha van legalább egy közvetlen út, amely összeköt egy várost az első halmazból egy másik várossal a másik halmazból. Egyébként 0-t ad vissza.

**setRoad() könyvtári függvény**

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

Ezzel a függvénnyel kell közölnöd, hogy a és b város között épült az új közvetlen út. Ha az új út nem az a és b város között épült, akkor a program befejeződik és 0 pontot kapsz az aktuális tesztesetre. Ha ez az (N-1). hívása volt a függvények, akkor a program befejeződik és pozitív pontszámot ad az aktuális tesztesetre. Egyéb esetben a Terránok új utat építenek és folytatni kell ennek felderítését. Minden további query() hívás figyelembe veszi a legutóbbi setRoad() hívás óta épített új közvetlen utat.

**run() saját függvényed**

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

Ezt a függvényt kell megvalósítanod és beküldened. Ebben hívhatók a query() és setRoad() függvények. Az N bemenő paraméter a Terránok által épített városok számát adja. Ha ez a függvény befejeződik, mielőtt kiderítetted volna az összes új utat, 0 pontot kapsz az aktuális tesztesetre.

**Nyelvi megjegyzések**

* C/C++: #include “icc.h” használata
* Pascal: definiáld a unit icc-t, amely importálja az értékelő könyvtári függvényeket: uses graderhelplib

**Pontozás**

A tesztesetek 6 csoportba vannak sorolva. Az egy csoportba tartozó tesztesetek mindegyikénél N értéke ugyanaz. Egy csoportra akkor kapsz pontot, ha a csoportba tartozó minden tesztesetre helyes választ adtál legfeljebb M darab query() függvényhívással. Az N és M, valamint a csoportra kapott pontszám értékét az alábbi táblázat tartalmazza.

|  |  |  |  |
| --- | --- | --- | --- |
| Csoport | N = városok száma | M = *query()* hívási korlátja | Pontszám |
| 1 | 15 | 1500 | 7 |
| 2 | 50 | 2500 | 11 |
| 3 | 100 | 2250 | 22 |
| 4 | 100 | 2000 | 21 |
| 5 | 100 | 1775 | 29 |
| 6 | 100 | 1625 | 10 |

**Példa futás**

|  |  |  |
| --- | --- | --- |
| Versenyző művelete | Értékelő műv. | Megjegyzés |
| -query(1, 3, {1}, {2, 3, 4})query(1, 2, {2}, {3, 4})query(1, 1, {2}, {3})setRoad(2, 4)query(2, 2, {2, 4}, {1, 3})setRoad(1, 3)query(2, 2, {2, 4}, {1, 3})query(1, 2, {2}, {1, 3})query(1, 1, {4}, {3})setRoad(4, 1) | run(4)return 0return 1return 0-return 0-return 1return 0return 0exit |  - Az értékelő N=4 várossal indít. Az első út a 2-es és 4-es város közé épül. Ezt a versenyző nem tudja.  - A lekérdezés az {1} és {2, 3, 4} halmazokra vonatkozik, és az eredmény azt jelzi, hogy nincs út az 1-es és a többi város között. - Lekérdezés a {2} és {3, 4} halmazokra. A válasz 1, mert van út a 2-es és 4-es város között. - Nem a 2-es és 3-as város között van az út. - Helyes a válasz (2, 4). Az értékelő új utat generál az 1-es és 3-as város között. - Lekérdezés a {2, 4} és {1, 3} halmazokra. Eredménye 0. - Helyes válasz (1, 3). Az értékelő új utat generál az 1-es és 4-es város közé. - Az előző lekérdezés. 1-es válasz az újonnan generált út miatt. - Helyes válasz (4, 1), és az értékelő maximális pontot ad a tesztre. |