Social Engineering

Problem Name	Social Engineering
Input file	Interactive task
Output file	Interactive task
Time limit	5 seconds
Memory limit	256 megabytes

Ett socialt nätverkt är en oriktad sammanhängande graf med *n* noder och *m* kanter, där varje nod är en person, och två personer är vänner om det finns en kant mellan dem.

Maria är en del av detta sociala nätverk. Hon gillar att utmana sina vänner att göra diverse saker. Detta betyder att hon först utför en enkel uppgift, och därefter utmanar en av hennes vänner att göra samma sak. Denna vän kommer sedan utmana en av sina vänner, som sedan utmanar en av sina vänner, och så vidare. Det kan hända att samma person blir utmanad mer än en gång, men varje oordnat par av vänner kan bara delta i utmaningen en gång. (När en person *A* utmanat en person *B*, så får varken person *A* eller person *B* utmana den andra igen.) Med andra ord, utmaningarna kommer bilda en vandring i grafen som aldrig använder en kant mer än en gång. En person förlorar utmaningen om det är deras tur och de inte kan utmana någon av sina vänner. Utmaningarna är alltid startade av Maria, och hon förlorar sällan. Nu har de återstående *n*-1 personerna bestämt sig för att samarbete för att se till att Maria förlorar nästa utmaning, och det är ditt jobb att koordinera detta uppdrag.

Implementation

Du ska implementera en funktion:

```
void SocialEngineering(int n, int m, vector<pair<int,int>> edges);
```

som spelar spelet på en graf med *n* noder och *m* kanter. Denna funktion kommer anropas en gång av en grader. Listan `edges` kommer innehålla exakt *m* par av heltal (*u*,*v*), som betyder att det finns en kant mellan nod *u* och nod *v*. Noder är numrerade från 1 till *n*. Maria är alltid nod 1. Din funktion kan göra anrop till följande funktioner:

```
int GetMove();
```

Denna funktion ska anropas när det är Marias tur, som i början av spelet till exempel. Om du anropar denna funktion när det inte är Marias tur kommer du få `Wrong Answer`. Denna funktion kan returnera ett av följande värden:

- ett heltal v, där $2 \leq v \leq n$. Detta betyder att Maria utmanar personen med index v. Detta kommer alltid vara ett tillåtet drag.
- 0, om Maria ger upp. Maria ger alltid upp om hon inte har några tillåtna drag. När detta händer ska ditt program låta funktionen `SocialEngineering` returnera, och du får `Accepted`.

```
void MakeMove(int v);
```

Denna funktion ska anropas när det inte är Marias tur. Detta betyder att personen vars tur det är nu, utmanar person v. Om detta inte är ett tillåtet drag eller om det är Marias tur så får du `Wrong Answer`. Om Maria har en vinnande strategi i början av spelet ska ditt program låta `SocialEngineering` returnera *innan* första anropet till `GetMove`. Du får då `Accepted`.

Gränser

- $2 \leq n \leq 2 \cdot 10^5$.
- $1 \leq m \leq 4 \cdot 10^5$.
- Grafen är sammanhängande. Varje oordnat par av noder dyker upp högst en gång som en kant, och varje kant går mellan två olika noder.

Delpoäng

Maria spelar alltid perfekt i benämningen att hon kommer göra vinnande drag så fort hon har en vinnande strategi. Om hon inte har en vinnande strategi så kommer hon försöka lura ditt program att göra misstag, på olika smarta sätt. Hon ger bara upp om hon inte har några tillåtna drag, förutom i Subtask 3.

Subtask 1. (15 poäng) $n, m \leq 10$.

Subtask 2. (15 poäng) Alla utom Maria har som mest 2 vänner.

Subtask 3. (20 poäng) Maria kommer ge upp direkt om hon inte har en vinnande strategi.

Subtask 4. (25 poäng) $n, m \leq 100$.

Subtask 5. (25 poäng) Inga ytterligare begränsningar.

Exempel-interaktion

Du gör något	Grader gör något	Förklaring
-	`SocialEngineering(5, 6, {{1,4}, {1,5}, {2,4}, {2,5}, {2,3}, {3,5}})`	`SocialEngineering` anropas med en graf med 5 noder och 6 kanter.
`GetMove()`	`Returnerar 4`	Maria utmanar person 4.
`MakeMove(2)`	-	Person 4 utmanar person 2.
`MakeMove(5)`	-	Person 2 utmanar person 5.
`MakeMove(1)`	-	Person 5 utmanar Maria.
`GetMove()`	`Returns 0`	Maria har inga tillåtna drag, så hon ger upp.
`Returnerar`	-	Du har vunnit spelet, och ska låta `SocialEngineering` returnera.

Du gör något	Grader gör något	Förklaring
-	`SocialEngineering(2, 1, {{1,2}})`	`SocialEngineering` anropas med en graf med 2 noder och 1 kant.
`Returnerar`	-	Maria har en vinnande strategi på denna graf så du ska returnera direkt för att ge upp.

Exempel-Grader

Den givna exempel-gradern, `grader.cpp`, i uppgifts-bilagan `SocialEngineering.zip`, läser input från standard input i följande format:

- Den första raden innehåller antalet noder, *n*, och antalet kanter, *m*, i grafen.
- De följande *m* raderna innehåller två heltal, *u* och *v*, vilket betyder att det finns en kant mellan *u* och *v*.

Exempel-gradern läser input och anropar `SocialEngineering`-funktionern i din lösning. Notera att exempel-gradern inte implementerar Marias vinnande strategi och ges bara för en exempel-interaktion. För att kompilera exempel-gradern med din lösning kan du använda följande kommando i terminalen: `g++ -std=gnu++11 -O2 -o solution grader.cpp solution.cpp`

där `solution.cpp` är lösnings-filen som du ska submitta. För att köra programmet med exempel-input från bilagan, skriv följande kommando i terminalen:

```
./solution < input.txt
```