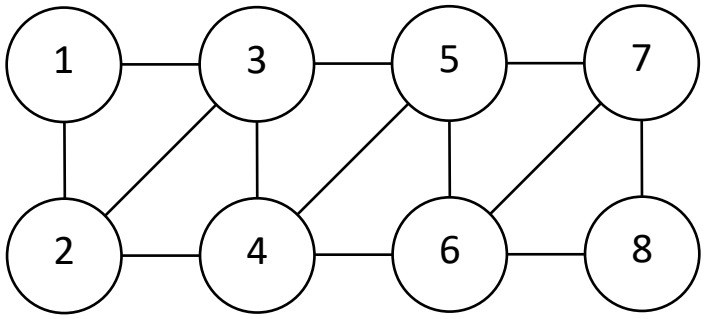
Vasi vrea să viziteze Ruse și să se plimbe în parcul acestuia. Din ghidul turistic ea știe ca parcul are o formă dreptunghiulară cu lungimea de N unități. Pe fiecare dintre cele doua laturi de lungime există $N + 1$ locații interesante (cafenele, terenuri de tenis, fântâni etc.), deci $2N + 2$ locații interesante în total. Acestea sunt numerotate și conectate prin alei, exact cum este prezentat în figura de mai jos ($N = 3$ în acest caz). Parcul are o singură intrare (numerotată cu 1) și o singură ieșire (numerotată cu $2N + 2$).



Toate aleile din parc sunt orientate și când cineva merge pe ele trebuie să respecte direcțiile lor. Aleile sunt orientate astfel încât **nu exista cicluri.** în plus, intrarea este singura locație unde **toate aleile conectate la ea ies din aceasta**, și ieșirea este singura locație unde **toate aleile conectate la ea intra în aceasta**. Pentru toate celelalte locații următoarea regula este respectata: **exista alei care intră în aceasta și alei care ies din locația respectiva**.

Vom spune că o anumită alee are o poziție pozitivă dacă pleacă de la un număr mai mic la un număr mai mare, iar altfel vom spune că are o direcție negativă. Vasi știe ca atât cele două alei pornind de la intrare cât și cele doua alei ajungând la ieșire au o direcție pozitivă (deoarece intrarea e numerotată cu 1 și ieșirea - cu numărul $2N + 2$), dar ea nu știe nimic despre direcțiile celorlalte alei și are nevoie să își planifice plimbarea în parc.

Pentru entuziasmații problemelor de matematică și informatică, la intrarea în parc exista un computer cu un program special pe el. Acesta răspunde întrebărilor de următorul tip: fiind dată o lista arbitrara de alei reprezentate prin numerele conectate de către ele, acesta returnează rezultatul operației de tip XOR asupra direcțiilor acestora (unde direcția pozitiva va fi reprezentata prin 1 și cea negativa prin 0). *Va reamintim că operația XOR aplicată asupra două numere de un bit este 1 daca aceste numere sunt diferite, și 0 în cazul în care sunt egale. în cazul în care există mai mult de doi operanzi, vom calcula XOR-ul primelor două, apoi cel dintre rezultatul operației și al treilea operand s.a.m.d.* **Programul este făcut în așa fel încât data fiind o lista ce conține o singura alee, acesta va returna doar direcția ei.**

Vasi vrea să afle direcțiile tuturor aleilor fără a pune prea multe întrebări programului.

Comisia a copiat programul de la intrarea în parc și acum îl uploadeaza la sistemul de testare. Ajutați-o pe Vasi să scrie o funcție run, care va fi compilată cu programul juriului și va comunica cu acesta prin întrebări asemenea celor descrise mai sus, și ne transmite direcțiile aleilor găsite. După ce s-a terminat de executat, funcția voastră va trebui să fi găsit corect și să comunice toate direcțiile aleilor.

Detalii de implementare

Funcția ta run trebuie să aibă următorul prototip:

```
void run(int n);
```

Aceasta va fi apelată doar o singură dată și va primi ca argumente numărul natural nenul *N* – lungimea parcului (numărul total de locații interesante este $2N + 2$).

Pentru a comunica cu programul juriului puteți folosi următoarele 2 funcții:

```
bool get_xor(const std::vector<std::pair<int, int>>& edges);
void state_direction(const std::pair<int, int>& edge, bool direction);
```

Pentru fiecare apel al lui `get_xor` va fi returnat rezultatul operației XOR pe direcțiile aleilor în vectorul `edges`. Fiecare alee este reprezentată printr-o pereche ce reprezintă indicii celor două locații ce sunt conectate de aceasta (neținând cont de ordine). A se nota ca funcția `get_xor` are o complexitate (si de timp și de spațiu) în raport liniar cu numărul de alei în întrebare.

Funcția voastră trebuie să apeleze `state_direction` pentru fiecare alee în parte (reprezentate în același fel) cu orientările găsite și transmise prin intermediul parametrului `direction`. Dacă la un moment programul apelează vreuna dintre cele 2 funcții cu o alee invalidă sau `state_direction` cu o direcție greșită, veți primi zero puncte pe acel test. Veți obține zero puncte pe un test și daca exista alei pentru care funcția voastră nu apelează `state_direction`.

Trimiteți un fișier numit **park.cpp** pe sistemul de evaluare. În el, pe lângă funcția `run`, puteți adăuga orice alte funcții, clase sau variabile ajutătoare. Aveți grijă să nu conțină vreo funcție cu numele `main` și să includă headerul `park.h` cu instrucțiunea de pre-processor `#include "park.h"` la începutul acestuia.

Restricții

$1 \leq N \leq 100000$

Evaluarea

Soluția voastră va primi punctaj nenul daca pentru fiecare test va da răspunsul corect, se va încadra în limita de timp și numărul de apelări ale funcției `get_xor` pentru fiecare test nu este mai mare de $3N$. Numărul de puncte primite va fi după cum urmează (Q – numărul de apelări ale funcției `get_xor` pentru un singur test și $\lceil x \rceil$ - cel mai mic numar intreg $\geq x$):

15 puncte : $Q \leq 3N$ pentru toate testele și $Q > 2N$ pentru cel puțin un test.

45 puncte: $Q \leq 2N$ pentru toate testele și $Q > \lceil \frac{3}{2} N \rceil$ pentru cel puțin un test.

70 puncte: $Q \leq \lceil \frac{3}{2} N \rceil$ pentru toate testele și $Q > \lceil \frac{16}{11} N \rceil$ pentru cel puțin un test.

100 puncte: $Q \leq \lceil \frac{16}{11} N \rceil$ pentru toate testele.

Testare locala

Veți primi programul `Lgrader.cpp`, pe care îl puteți compila alături de programul de test. Rulat, va cere o valoare pentru N si, după acesta, direcțiile aleilor în ordinea lor (0 sau 1). Programul va afișa interacțiunea. Puteți modifica acest program după cum doriți.

Testare pe sistemul de evaluare și teste adaptive

Pentru niște teste (necunoscute vouă), programul comisiei se va adapta în funcție de query-urile puse de programul concurentului, mai exact, direcțiile aleilor din parc se vor alege în funcție de query-urile folosite. În cazul testării programului pe sistemul de evaluare, nu veți avea acces la opțiunea testelor adaptive. Formatul fișierului de input trebuie să fie următoarea: un număr natural N, urmat de $4N + 1$ de unu și zero – orientarea aleilor în ordinea lor (1-2, 1-3, 2-3, 2-4, …).

Exemplu de interacțiune

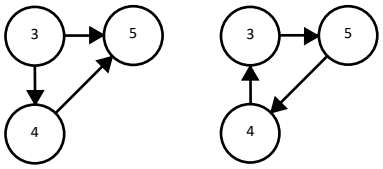
№	Acțiunile `run`	Acțiunile și răspunsurile programului comisiei
1.		`run(2)`
2.	`state_direction({1,2}, 1)`	
3.	`state_direction({1,3}, 1)`	
4.	`state_direction({4,6}, 1)`	
5.	`state_direction({5,6}, 1)`	
6.	`get_xor({{3,4},{3,5},{4,5}})`	`1`
7.	`get_xor({{3,5}})`	`1`
8.	`state_direction({3,4}, 1)`	
9.	`state_direction({3,5}, 1)`	
10.	`state_direction({4,5}, 1)`	
11.	`get_xor({{2,3},{2,4}})`	`0`
12.	`state_direction({2,3}, 1)`	
13.	`state_direction({2,4}, 1)`	
14.	Funcția se termina	

Explicarea interacțiunii pe exemplul de mai sus

1. Programul comisiei apelează `run` cu $N = 2$.

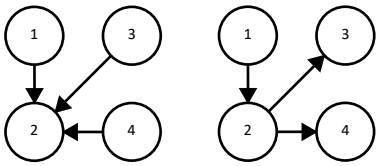
2-5. `Run` transmite direcția aleilor de la intrare la ieșire.

6-7. `Run` întreabă despre 2 XOR-uri și primește răspunsul 1 pentru ambele. în continuare, vom examina ambele posibilități pentru alei:



8-10. A doua varianta conține un ciclu, doar ca nu exista cicluri în parc, deci `run` transmite orientările pentru prima varianta.

11 `Run` întreabă care este XOR-ul direcțiilor celor 2 alei și primește răspunsul 0. Deci, direcțiile lor sunt aceleași:



12 -13 În prima variantă aleile din/spre locația 2 sunt orientate spre 2, doar că există o singură locație care ar putea garanta aceasta, ieșirea, iar locația 2 nu este ieșirea, deci `run` transmite direcțiile corespunzătoare celei de a doua variante.

14. `Run` se termina. 3 query-uri au fost folosite.