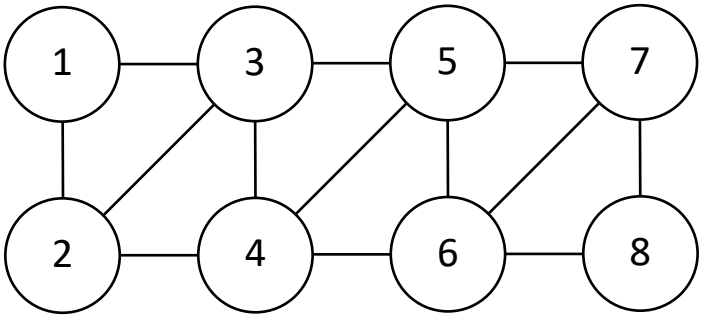
Васи най-накрая реши да посети Русе и да разгледа тамошния парк. От своя екскурзовод знае, че паркът е с правоъгълна форма, като дългите му страни са дълги N единици. На всяка от тези две страни са разположени $N + 1$ интересни места (барчета, тенис кортове, чешми и т.н.), т.е. общо има $2N + 2$ такива места. Те са номерирани и свързани с алеи така, както е показано на фигурата по-долу (в случая $N = 3$). Паркът има единствен вход (място с номер 1) и единствен изход (място с номер $2N + 2$ – на фигурата това е място с номер 8).



Всички алеи в парка са еднопосочни, като при разходка е задължително да се спазва разрешената посока на всяка алея. Алеите са така ориентирани, че **не се образуват цикли**. Освен това входът е единственото място, **от което само излизат алеи**, а изходът е единственото място, **в което само влизат алеи**. За всички останали интересни места важи правилото, че **има както влизащи, така и излизащи алеи**.

Ще казваме, че дадена алея има положителна посока, ако тя води от място с по-малък номер до място с по-голям номер, иначе ще казваме, че алеята е с отрицателна посока. Васи е наясно, че двете алеи, които излизат от входа и двете алеи, които влизат в изхода са с положителна посока (тъй като входът е с номер 1, а изходът – с номер $2N + 2$). Тя не знае нищо за посоките на другите алеи, а това ѝ е нужно, за да планира своята разходка в парка.

За любителите на математически и информатични задачи на входа на парка е сложен компютър, на който работи специална програма, която отговаря на въпроси от следния странен тип: към програмата се подава произволен списък от алеи чрез номерата на интересните места, които те свързват, и тя връща резултата от операция XOR на посоките им (приема се, че положителната посока се означава с 1, а отрицателната с 0). *Да напомним, че XOR на две еднобитови числа дава резултат 1, ако те са различни, и 0, ако са еднакви. В случай на повече от два операнда, първо се прилага XOR към първите два, след това към получения резултат и третия операнд и т.н.* **Програмата е така направена, че ако в подадения към нея списък има само една алея, то тя просто връща нейната посоката**.

Васи иска да узнае посоките на всички алеи без да задава твърде много въпроси към програмата.

Журито е копирало програмата от входа на парка и сега тя се намира в системата. Помогнете на Васи като напишете функция `run`, която ще се компилира с програмата на журито и ще комуникира с нея, като задава въпроси от гореописания вид и съобщава откритите посоки на алеи. След края на изпълнението си, Вашата функция трябва вярно да е открила и съобщила посоките на всички алеи в парка.

Детайли по имплементацията

Вашата функция `run` трябва да има следния прототип:

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

Тя ще бъде извикана само веднъж и ще получи като аргумент цялото положително число N - дължината на дългата страна на парка (общият брой на интересните места в парка е $2N + 2$).

За комуникацията с програмата на журито можете да използвате следните две функции:

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

При всяко извикване на функция `get_xor` ще бъде върнат резултатът от операция XOR на посоките на алеите във вектор `edges`. Всяка алея е представена като двойка от номерата на двата върха, които

свързва (независимо от реда). Обърнете внимание, че сложността на `get_xor` (и по време, и по памет) е линейна по броя алеи в запитването.

Вашата функция трябва да извика `state_direction` за всяка алея (отново представена по същия начин) с откритата посока на алеята, която се съобщава чрез параметър `direction`. Ако в някой момент Вашата програма извика някоя от двете функции с невалидна алея или `state_direction` с грешна посока, ще получите нула точки за съответния тест. Нула точки ще получите и за тест, в който съществуват алеи, за които Вашата функция не е извикала `state_direction`.

Към системата изпратете файл **park.cpp**. В него, освен функцията `run`, може да има всякакви помощни функции, структури, променливи и т.н. Той само не трябва да съдържа функция `main` и трябва да включва хедър файла `park.h` чрез указване към предпроцесора `#include "park.h"` в началото си.

Ограничения

$1 \leq N \leq 100\,000$

Оценяване

Вашето решение ще получи точки, различни от 0, ако за всеки тест изпълни задачата в рамките на зададения time limit и броят на зададените въпроси чрез викане на функция `get_xor` за всеки тест не надвишава $3N$. Броят на получените точки ще бъде както следва (Q – брой извиквания на функция `get_xor` за един тест; $\lceil x \rceil$ – най-малкото цяло число $\geq x$):

15 точки : $Q \leq 3N$ за всички тестове, но $Q > 2N$ дори за един тест.

45 точки : $Q \leq 2N$ за всички тестове, но $Q > \left\lceil \frac{3}{2}N \right\rceil$ дори за един тест.

70 точки : $Q \leq \left\lceil \frac{3}{2}N \right\rceil$ за всички тестове, но $Q > \left\lceil \frac{16}{11}N \right\rceil$ дори за един тест.

100 точки : $Q \leq \left\lceil \frac{16}{11}N \right\rceil$ за всички тестове.

Локално тестване

Предоставен Ви е файлът `Lgrader.cpp`, който може да компилирате заедно с вашата програма, за да я тествате. При стартиране ще Ви пита за N, след което за посоките на всички алеи подред (0 или 1). Ще отпечатва комуникацията, която се извършва. Може да модифицирате този файл както искате.

Тестване на системата и адаптивни тестове

На определени (неизвестни за вас) тестове програмата на журито ще се адаптира към въпросите на вашата програма, т.е. посоките на алеите в парка ще се окажат различни в зависимост от това какви запитвания са изпратени. Ако тествате програмата си на системата като качите ваш входен файл, няма да имате достъп до адаптивната възможност. Формата на входния файл трябва да бъде следният: естественото число N, последвано от $4N + 1$ нули и единици – посоките на алеите подред (1-2, 1-3, 2-3, 2-4, ... , k-(k+1), k-(k+2), …).

Примерна комуникация

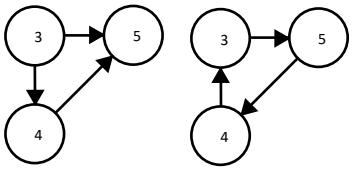
№	Действия на `run`	Действия и отговори на журито
1.		`run(2)`
2.	`state_direction({1,2}, 1)`	
3.	`state_direction({1,3}, 1)`	
4.	`state_direction({4,6}, 1)`	
5.	`state_direction({5,6}, 1)`	
6.	`get_xor({{3,4},{3,5},{4,5}})`	`1`
7.	`get_xor({{3,5}})`	`1`
8.	`state_direction({3,4}, 1)`	
9.	`state_direction({3,5}, 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.	Функцията приключва изпълнение.	

Обяснение на примерната комуникация

1. Програмата на журито вика `run` с $N = 2$.

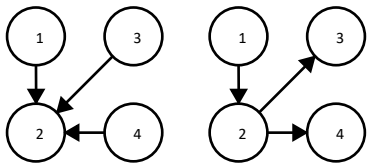
2-5. `Run` съобщава посоките на алеите от входа и към изхода.

6-7. `Run` пита за два XOR-а, и получава 1 и на двата. Нека видим двете възможности за алеите:



8-10. Втория вариант съдържа цикъл, но такива няма в парка, затова `run` съобщава посоките от първия вариант.

11. `Run` пита за XOR на посоките на две алеи и получава 0. Това значи, че двете имат еднаква посока:



12-13. В първия вариант всички алеи от/към място 2 влизат в него, но в парка има само едно такова място (изхода) и това не е то, затова `run` съобщава посоките от втория вариант.

14. `Run` приключва изпълнението си. Използвани са 3 запитвания.