Задача A. Реверси

Ограничение по времени:	10 секунд
Ограничение по памяти:	256 мебибайт

Реверси — игра на квадратной клетчатой доске 8×8. В игре используются 64 специальных камня, окрашенных с разных сторон в белый и чёрный цвета. Один из игроков играет белыми, другой — чёрными. Делая ход, игрок ставит фишку на клетку доски «своим» цветом вверх.

Будем нумеровать горизонтали и вертикали целыми числами от нуля до семи. В начале игры в центр доски выставляются четыре камня: чёрные на $(3, 3)$ и $(4, 4)$, белые на $(3, 4)$ и $(4, 3)$. Первый ход делают чёрные. Далее игроки ходят по очереди.

Делая ход, игрок должен поставить свой камень на одну из клеток доски таким образом, чтобы между этим поставленным камнем и одним из уже имеющихся на доске камней его цвета находился непустой непрерывный ряд камней соперника — горизонтальный, вертикальный или диагональный (другими словами, чтобы непрерывный ряд камней соперника оказался «закрыт» камнями игрока с двух сторон). Все камни соперника, входящие в «закрытый» на этом ходу ряд, переворачиваются на другую сторону (меняют цвет) и переходят к ходившему игроку.

Если в результате одного хода «закрывается» одновременно более одного ряда камней противника, то переворачиваются все камни, оказавшиеся на всех «закрытых» рядах.

Игрок вправе выбирать любой из возможных для него ходов. Если игрок имеет возможные ходы, он не может отказаться от хода. Если игрок не имеет допустимых ходов, то ход передаётся сопернику.

Игра прекращается, когда на доску выставлены все камни или когда ни один из игроков не может сделать хода. По окончании игры проводится подсчёт камней каждого цвета, и игрок, чьих фишек на доске выставлено больше, объявляется победителем. В случае равенства количества камней засчитывается ничья.

Описание правила игры основано на материале из Википедии — свободной энциклопедии

Протокол взаимодействия

Для взаимодействия с проверяющей системой вам предстоит реализовать функцию `move`, имеющую следующую сигнатуру:

```
void move(const int a[8][8], const int c, int* row, int* column); // C++
public void move(int[][] a, int c, int[] move); // java
```

- `a` — это двумерный массив из 8×8 элементов, нумерующийся с нуля, описывающий текущее положение на доске. Значение -1 соответствует пустому полю, значение 0 — камню, лежащему белой стороной вверх, значение 1 — камню, лежащему чёрной стороной вверх.
- `c` — цвет камней игрока: 0 — белый, 1 — чёрный. Гарантируется, что в процессе одного запуска этот параметр не будет меняться.
- Остальные параметры служат для возврата координаты клетки, в которую следует сделать ход.

  В `C++` ответ следует поместить по соответствующим указателям,

  В `Java` параметр `move` является массивом размером два, в который стоит поместить ответ, сначала номер строки, потом номер столбца.

Если после очередного хода противника окажется, что у вас нет корректных ходов, то функция `move` запускаться не будет.

Замечание

Вы можете посылать свои решения в ejudge, где они будут запускаться на 4 тестах, два теста за белых, два теста за чёрных.

Программа проходит тест, если она корректно доигрывает до конца игры, победить стратегию жюри не требуется.

Если ваша программа пройдёт все 4 теста, то она получит вердикт `OK` в ejudge.

Периодически, жюри будет выкачивать для каждого участника последнюю `OK` посылку из ejudge и играть из них турнир, начисляя за это баллы. Обратите внимание, что некорректный ход приводит к досрочному поражению в матче.

Также есть возможность играть с программами других участников — для этого воспользуйтесь специальной страничкой.

Вы можете скачать архив с материалами.

Внутри архива есть файл для локального запуска двух стратегий, `interact.py`, убедитесь что у вас стоит питон достаточно свежей версии.

Чтобы запустить тестирование передайте программе ровно два аргумента — команды для запуска двух стратегий, ключ `--verbose` добавляет печать дополнительных сообщений о текущем состоянии поля.

Файл `grader.cpp` читает запросы из `stdin` и вызывает вашу функцию, чтобы получить корретный исполняемый файл слинкуйте его вместе с вашей стратегией, например можно сделать:

```
g++ -Wall -Wextra -std=c++11 grader.cpp solution.cpp -o solution
./interact.py "./solution" "./solution" # запустить решение само с собой.
```

Если вы пишете на Java, то обратите внимание на файлы `Grader.java` и `Solution.java`, ваше решение должно называться `Solution.java`, для запуска скомпилируйте оба файла через `javac` и запускайте класс `Grader`.

```
javac Grader.java Solution.java
./interact.py "java Grader" "java Grader" # запустить решение само с собой.
```