타일링 게임

바르친과 차로스는 단위 사각형 칸들로 이루어진 $2N \times 2M$ 격자에서 게임을 하고 있다. 행들은 위에서 아래로 0부터 $2N-1$까지 번호가 붙어있고, 열들은 왼쪽에서 오른쪽으로 0부터 $2M-1$까지 번호가 붙어있다. $0 \leq i < 2N$과 $0 \leq j < 2M$에 대해서, i행과 j열의 칸을 (i, j)로 나타낸다.

바르친은 차로스에게 $N \cdot M$개 **블록**들을 하나씩 준다. 각 블록은 4 개의 1×1 **타일**들로 이루어진 2×2 정사각형이다. 바르친은 모든 타일을 검정색 또는 흰색으로 색칠한다. 이 때, 각 블록에 **적어도 하나의 타일이 흰색**임을 보장한다.

차로스는 미래에 어떤 블록을 받게 될 지 모르는 상태에서 **각 블록을 받자마자** 격자에 놓아야 한다. 블록은 회전할 수 없다. 각 블록은 정확히 4개의 격자 칸을 덮으면서 격자 안에 완전히 놓여야만 한다. 또한 각 블록의 왼쪽 위 타일은 행과 열의 번호가 모두 짝수인 칸을 덮어야 한다. 격자의 각 칸은 많아야 하나의 블록에 의해서 덮여야 한다.

임의의 블록이 놓여진 후, 4개 검은색 타일로 덮이는 2×2 정사각형 영역이 존재하면, 바르친이 이긴다. 다시 말해서, 어떤 $0 \leq a < 2N-1$와 $0 \leq b < 2M-1$에 대해서, 칸 (a, b), $(a+1, b)$, $(a, b+1)$, $(a+1, b+1)$가 검은색 타일들로 모두 덮이면, 바르친이 이긴다. 여기서, a와 b는 짝수일 필요는 없다. 4개 검은색 타일은 같은 블록에 속할 수도 있고, 다른 블록에 속할 수도 있다.

차로스는 바르친이 이기지 못하게 하면서 $N \cdot M$개의 블록을 모두 놓으면 승리한다. 블록을 $N \cdot M$ 개 놓으면 격자가 완전히 덮인다는 점에 유의하자.

당신은 차로스가 게임을 이기는 전략을 구현해야 한다. 주어진 제약 조건 하에서 차로스는 미래에 받을 블록의 색깔과 관계없이 항상 이길 수 있도록 블록을 배치할 수 있다는 것을 증명할 수 있다.

Implementation Details

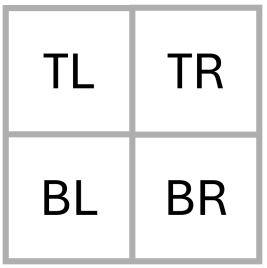
당신은 두 개의 함수를 구현해야 한다:

```
void init(int N, int M)
```

- N : 격자의 행 개수의 절반.
- M : 격자의 열 개수의 절반.
- 이 함수는 프로그램 실행 시작 시, 테스트 케이스당 정확히 한 번 호출된다.

```
std::pair<int, int> receive_block(int TL, int TR, int BL, int BR)
```

- TL , TR , BL , BR : 아래 그림과 같이 현재 블록의 왼쪽 위, 오른쪽 위, 왼쪽 아래, 오른쪽 아래 타일의 색상을 각각 나타낸다. 각 값은 0 (흰색) 또는 1 (검은색)이다.
- 이 함수는 `init` 의 최초 호출 이후 테스트 케이스당 정확히 $N \cdot M$ 번 호출된다.



이 함수는 정수 쌍 (i, j)를 반환해야 한다. 여기서, i 는 이 블록의 왼쪽 위 타일을 놓아야 하는 칸의 행 번호이고 j 는 열 번호이다. i 와 j 모두 짝수여야 하며, 블록이 덮는 2×2 영역은 이전에 배치된 어떤 블록과도 겹치지 않아야 한다.

`receive_block` 이 이러한 요구 사항들을 만족하지 않는 쌍을 반환하거나, 블록을 배치한 후 어떤 2×2 정사각형 영역이 검은색 타일로 완전히 덮이는 경우, 그레이더는 즉시 프로그램을 종료하고 해당 테스트 케이스에 대한 판정은 `Output isn't correct`가 될 것이다.

그레이더의 동작은 **적응적이지 않다(not adaptive)**. 즉, 바르친이 차로스에게 주는 블록 순서는 `init`이 호출되기 전에 고정된다.

Constraints

각 블록에 대해, 해당 블록을 구성하는 네 개의 타일 중 검은색 타일의 개수를 S 라고 하자. 즉, $S = TL + TR + BL + BR$.

- $1 \le N, M \le 100$
- 모든 블록에 대해, $0 \le S \le 3$

Subtasks

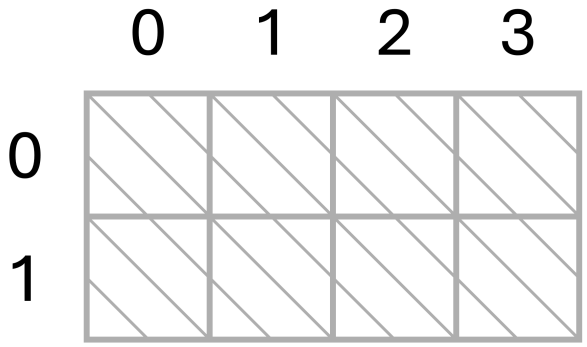
Subtask	Score	Additional Constraints
1	6	모든 블록에 대해 $S = 1, N = 2$.
2	16	모든 블록에 대해 $S = 3$, $N = M$, N은 짝수이고, 가능한 네 가지 블록 색칠 각각이 정확히 $\frac{N^2}{4}$ 번 나타난다.
3	10	모든 블록에 대해 $S = 1$.
4	29	모든 블록에 대해 $S \le 2$.
5	39	추가적인 제약이 없다.

Example

$N = 1$이고 $M = 2$인 게임을 생각하자. 따라서 격자는 2개 행과 4개 열을 가진다. 그레이더는 먼저 다음을 호출한다:

```
init(1, 2)
```

초기에, 모든 칸은 비어있다. 격자는 다음과 같다:

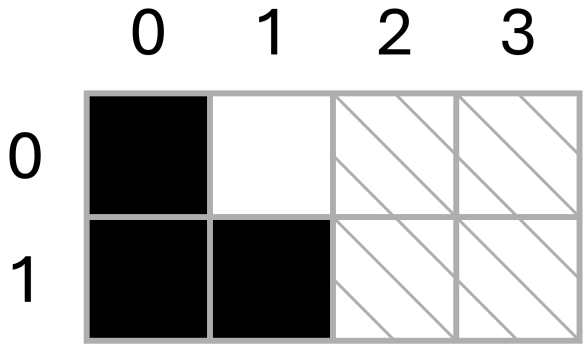


총 $N \cdot M = 2$개의 블록을 배치해야 한다. 바르친이 먼저 검은색 타일 세 개와 오른쪽 위 모서리에 흰색 타일 하나가 있는 블록을 준다고 가정해 보자.

```
receive_block(1, 0, 1, 1)
```

차로스는 $(0, 0)$을 반환하는 것으로 격자의 왼쪽에 이 블록을 놓도록 결정한다.

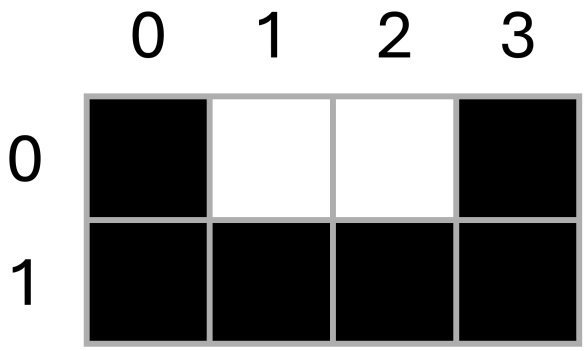
격자는 이제 다음과 같다:



그러면 바르친은 세 개의 검은색 타일과 왼쪽 위에 흰색 타일 하나를 가진 또 다른 블록을 준다:

```
receive_block(0, 1, 1, 1)
```

2×2 블록의 왼쪽 위 역할을 할 수 있는 짝수 행과 짝수 열을 가진 유일한 남은 칸은 $(0, 2)$ 이므로 차로스는 $(0, 2)$을 반환한다. 최종적으로 격자는 다음과 같다:



어떤 2×2 사각형 영역도 검은색 타일로 완전히 덮이지 않았으므로, 차로스는 바르친이 승리하지 못한 상태에서 모든 블록을 성공적으로 놓았다. 차로스가 게임에서 이긴다.

Sample Grader

Input format:

```
N M
TL[0] TR[0] BL[0] BR[0]
TL[1] TR[1] BL[1] BR[1]
...
TL[NM-1] TR[NM-1] BL[NM-1] BR[NM-1]
```

Output format:

```
R[0] C[0]
R[1] C[1]
...
R[NM-1] C[NM-1]
```

여기서, $R[k]$ 와 $C[k]$ 는 `receive_block` 에 대한 k 번째 호출에서 반환된 정수 쌍이다.