행성 탐사

승현이와 상수는 최근 발견되어 많은 관심을 받고 있는 *지학행성*을 탐사하고 있습니다. 이 둘은 편의상 지학행성을 2222×2222 크기의 격자판으로 나타냈습니다. 행과 열에는 0 이상 2221 이하의 정수 번호가 차례대로 붙어 있습니다. 행의 번호가 증가하는 방향을 남쪽, 행의 번호가 감소하는 방향을 북쪽, 열의 번호가 증가하는 방향을 동쪽, 열의 번호가 감소하는 방향을 서쪽으로 생각합니다. 지학행성은 도넛 모양으로 형성되었기 때문에 2221행의 남쪽으로 인접한 행은 0행이며, 2221열의 동쪽으로 인접한 열은 0열입니다. 편의상 이 격자판의 r행 c열에 있는 격자를 (r, c)로 표기합니다.

탐사 과정에는 여러 가지 장치들이 사용됩니다. 대부분의 장치들은 우주선에 달려 있으므로, 승현이와 상수가 가지고 있는 장치들만을 생각해 봅시다. 안타깝게도 이 장치들에는 GPS와 같은 기능이 없어서 자신들의 절대적인 위치(행과 열 번호)를 알지 못하기 때문에, 각 장치는 자신의 위치가 (0, 0)이라고 가정하고 모든 기능을 수행합니다.

승현이는 *페인트 발사 장치*를 가지고 있습니다. 승현이와 페인트 발사 장치는 항상 같은 격자에 있습니다. 이 장치는 아래와 같은 한 가지 기능을 수행합니다.

- $\text{paint}(a, b)$ ($0 \le a, b < 2222$): 페인트를 발사하여 승현이가 있는 격자로부터 a칸 남쪽, b칸 동쪽에 위치한 격자를 색칠합니다. 다시 말해, 장치가 (r, c)에 있다면, 이 장치는 $((r + a) \mod 2222, (c + b) \mod 2222)$를 색칠합니다. 예를 들어, 페인트 발사 장치가 $(7, 15)$에 있는 상태에서 $\text{paint}(19, 42)$를 호출하면 $(26, 57)$이, $\text{paint}(691, 2213)$을 호출하면 $(698, 6)$이 색칠됩니다.

상수는 *페인트 측정 장치*를 가지고 있습니다. 상수와 페인트 측정 장치는 항상 같은 격자에 있습니다. 이 장치는 아래와 같은 두 가지 기능을 수행합니다.

- $\text{count\_row}(a)$ ($0 \le a < 2222$): 상수가 있는 격자로부터 a칸 남쪽에 위치한 행의 격자들 중에서 승현이의 페인트 발사 장치로 인해 색칠된 격자들이 몇 개인지 알려줍니다. 다시 말해, 장치가 (p, q)에 있다면, 이 장치는 $(p + a) \mod 2222$행에서 승현이의 장치로 인해 색칠된 격자들의 수를 알려줍니다.
- $\text{count\_col}(b)$ ($0 \le b < 2222$): 상수가 있는 격자로부터 b칸 동쪽에 위치한 열의 격자들 중에서 승현이의 페인트 발사 장치로 인해 색칠된 격자들이 몇 개인지 알려줍니다. 다시 말해, 장치가 (p, q)에 있다면, 이 장치는 $(q + b) \mod 2222$열에서 승현이의 장치로 인해 색칠된 격자들의 수를 알려줍니다.

4,937,284개나 되는 수많은 격자들을 탐사하기에는 시간이 부족하기 때문에, 승현이와 상수는 흩어져서 탐사를 진행하기로 했습니다. 문제는 그렇게 하면 위 장치들이 필요가 없어진다는 것입니다. 공들여 만든 장치들을 버리기에는 아쉬웠기 때문에 어떻게 해야 할지 고심하던 승현이는, 묘안을 생각해 냅니다. 바로 이 장치들을 연락이 되지 않을 때 사용할, 서로 만나기 위한 임시 연락 수단으로 사용하자는 것입니다. 상수는 자신들의 위치도 모르는 장치들로 연락하는 것이 가능할지에 대한 의구심이 들었습니다. 이에 승현이는 모든 가능한 경우에 대하여 상수와 승현이가 만날 수 있도록 하는 방법을 찾겠다며 호언장담했습니다.

서로 연락이 불가능하기 때문에, 승현이와 상수는 정해진 시각에 자신들이 가지고 다니는 장치를 사용해야 합니다. 만약 이 둘이 서로 흩어졌다면, 0시(자정)이 될 때까지 기다리다가, 0시가 되면 승현이는 0초부터 시작해서 매 1분마다, 상수는 30초부터 시작해서 매 1분마다 장치를 단 한 번 사용하기로(명령을 내리기로) 했습니다. 즉 승현이는 0시, 0시 1분, 0시 2분, ...에 장치를 사용하고, 상수는 0시 30초, 0시 1분 30초, 0시 2분 30초, ... 에 장치를 사용하는 것입니다.

승현이는 상수로부터 어떤 정보도 제공받지 못하기 때문에 자신이 위치한 격자로부터 절대로 움직이지 않습니다. 상수 또한 승현이가 어디 있는지 알아내기 전까지는 움직이지 않습니다. 승현이는 상수가 가능한 한 빨리 자신의 위치를 찾을 수 있도록 하는 방법을 개발하고자 합니다.

문제

승현이를 도와 페인트 발사 장치와 페인트 측정 장치를 사용하여 상수가 승현이의 위치를 찾을 수 있도록 하는 프로그램을 작성하세요. 상수는 승현이와 만나기만 하면 되므로, 승현이가 자신이 있는 격자로부터 x칸 남쪽, y칸 동쪽에 있다는 사실만 알아내면 됩니다.

구현

여러분은 승현이의 역할을 하는 함수 `ainta()` 와 상수의 역할을 하는 함수 `sangsoo()` 을 C/C++로 구현하여 그 소스 코드를 제출해야 합니다. 이 함수들에는 파라미터가 없으며, 실제 채점 시 한 번 실행할 때 각각 최대 222,222번까지 호출될 수 있습니다. 각 호출은 독립적이므로, 이전의 호출로 인해 현재의 호출이 영향을 받는 일이 없도록 해야 할 것입니다. (초기화 등)

채점기 함수

아래의 함수들은 채점기에 구현되어 있는 함수들이며, `planet.h` 에서 프로토타입을 확인할 수 있습니다. 이 함수들은 각 테스트 케이스마다 독립적으로 동작하며, 상수(constant) 시간에 동작합니다.

- `void paint(int a, int b)` ($0 \leq a, b < 2222$): 위의 $\mathrm{paint}(a, b)$와 같은 기능을 합니다. 승현이만이 호출할 수 있습니다.
- `int count_row(int a)` ($0 \leq a < 2222$) : 위의 $\mathrm{count_row}(a)$와 같은 기능을 합니다. 상수만이 호출할 수 있습니다.

- `int count_col(int b)` ($0 \le b < 2222$): 위의 $\mathrm{count_col}(b)$와 같은 기능을 합니다. 상수만이 호출할 수 있습니다.
- `void report(int x, int y)` ($0 \le x, y < 2222$): 상수가 승현이가 어디 있는지 찾으면 호출합니다. 승현이가 있는 격자를 (r, c), 상수가 있는 격자를 (p, q)로 둘 때 $x = (r - p + 2222) \mod 2222, y = (c - q + 2222) \mod 2222$를 만족해야 정답 처리됩니다.

허용되지 않는 함수 호출(ex: 승현이가 `count_row` 함수 호출, 상수가 `paint` 함수 호출)을 할 시 프로그램이 강제 종료되며 오답 처리됩니다.

구현해야 할 함수 : `ainta()`

```
void ainta();
```

이 함수는 승현이의 역할을 대신하는 함수입니다. 이 함수에서는 `paint(a, b)` 함수를 호출할 수 있습니다. 최초의 시각은 0시이며, `paint` 함수를 호출할 때마다 1분의 시간이 흐릅니다. (물론 프로그램이 1분 동안 실행되는 것은 아닙니다.) 여러분은 `ainta()` 함수를 종료하기 전까지 `paint` 함수를 최소 0번, 최대 5,000번 호출할 수 있습니다. 5001번 이상 호출할 시 프로그램이 강제 종료되며 오답 처리됩니다.

구현해야 할 함수 : `sangsoo()`

```
void sangsoo();
```

이 함수는 상수의 역할을 대신하는 함수입니다. 이 함수에서는 `count_row(a)` 함수와 `count_col(b)` 함수를 호출하여 승현이의 위치를 추측할 수 있습니다. 최초의 시각은 0시 30초이며, `count_row` 함수 또는 `count_col` 함수를 호출할 때마다 1분의 시간이 흐릅니다. (물론 프로그램이 1분 동안 실행되는 것은 아닙니다.) 상수가 승현이의 위치를 알아내면, `report(x, y)` 함수를 호출하여 결과를 반환한 뒤 함수를 종료해야 합니다. 만약 `report` 함수를 호출하지 않고 함수를 종료하거나, `report` 함수를 호출한 이후 `count_row` 또는 `count_col` 함수를 호출하거나, `report` 함수를 두 번 이상 호출할 시 프로그램이 강제 종료되며 오답 처리됩니다.

실행 과정

실제 채점 과정과 예제 채점기 모두에서 실행 과정은 같습니다. 프로그램이 한 번 실행할 때마다 여러 개의 테스트 케이스가 주어집니다. 각 테스트 케이스마다, 우선 `ainta()` 가 호출되어 승현이의 계획을 받고, 그 이후 `sangsoo()` 가 호출되어 상수가 승현이의 위치를 찾도록 도와줍니다. 만약 실행 과정에서 문제가 발생했거나, 비정상적인 함수 호출을 했거나, 오답이 발생한 경우(승현이의 위치를 잘못 찾은 경우) 그 즉시 프로그램이 종료됩니다. 여러분의 코드는 표준 입력 또는 출력, 파일 입출력 등을 활용해서는 안 되며, 그렇게 할 시 오답 처리됩니다.

예시

$r = 1, c = 3, p = 2, q = 2$라고 가정합니다. 아래와 같은 코드를 작성했다고 합시다.

```
#include "planet.h"

void ainta() {
        paint(0, 1);
    paint(2, 1);
    paint(2221, 2220);
}

void sangsoo() {
        int a = count_col(1);
    int b = count_col(2);
    int c = count_row(0);
    int d = count_row(2221);
    report(2221, 1);
}
```

편의상 $N = 2222$로 둡니다.

현재 시각	ainta() 에서의 함수 호출	sangsoo() 에서의 함수 호출	설명
0시 정각	paint(0,1)	-	$((r+0) \mod N, (c+1) \mod N) = (1, 4)$가 색칠됩니다.
0시 0분 30초	-	count_col(1)	$(q+1) \mod N = 3$열에는 색칠된 칸이 없으므로 0을 반환합니다.
0시 1분	paint(2, 1)	-	$((r+2) \mod N, (c+1) \mod N) = (3, 4)$이 색칠됩니다.
0시 1분 30초	-	count_col(2)	$(q+2) \mod N = 4$열에는 $(1, 4)$와 $(3, 4)$만이 색칠되어 있으므로 2을 반환합니다.
0시 2분	paint(2221, 2220)	-	$(r+2221) \mod N, (c+2220) \mod N) = (0, 1)$

현재 시각	`ainta()` 에서의 함수 호출	`sangsoo()` 에서의 함수 호출	설명
0시 2분 30초	-	`count_row(1)`	$(p+1) \mod N = 3$행에는 $(3, 4)$만이 색칠되어 있으므로 1을 반환합니다.
0시 3분	-	-	`ainta()` 가 종료되었으므로 색칠하지 않습니다. 승현이가 상수보다 더 오랫동안 색칠을 할 수도 있습니다.
0시 3분 30초	-	`count_row(2221)`	$(p+2221) \mod N = 1$행에는 $(1, 4)$만이 색칠되어 있으므로 1을 반환합니다.
0시 4분 30초	-	`report(2221, 1)`	상수가 정확한 위치를 어떻게든 알아냈으므로(?) `report` 함수를 호출합니다. $(r-p+N) \mod N = (1-2+N) = 2221$이고, $(c-q+N) \mod N = 3-2 = 1$이므로 정답입니다.

부분문제

이 문제는 두 개의 부분문제로 이루어져 있습니다. 각 부분문제의 정보는 아래와 같습니다.

부분문제	점수	한 데이터당 테스트 케이스 수	참고
1	40	22,222	1번 부분문제에서 0점을 받을 시 2번 부분문제는 0점 처리됩니다.
2	60	222,222	가능한 모든 경우가 주어집니다.

만약 여러분이 실행 시간이 오래 걸리는 풀이를 생각했다면, 1번 부분문제에서 점수를 받고 2번 문제에서 제한 시간 초과를 받을 수도 있습니다.

채점 방식

1개 이상의 데이터에서 오답이 발생하거나 프로그램이 정상적으로 실행되지 않았다면 그 부분문제는 0점 처리됩니다.

프로그램이 정상적으로 실행되었고 오답이 발생하지 않았다면, 여러분의 점수는 함수 호출 횟수에 따라 결정됩니다. k번 부분문제의 모든 데이터에 대해, 승현이가 장치를 사용한 횟수와 상수가 장치를 사용한 횟수 중 최댓값을 C로 둡시다.

$$P = \begin{cases} 2 & \text{if } C \leq 101 \\ 2^{101/C} & \text{if } 101 < C \leq 5000 \\ 0 & \text{if C} > 5000 \end{cases}$$

로 두면, k번 부분문제의 점수는
$10 \times (k+1) \times P$를 소수점 아래 첫째 자리 (10^{-1}의 자리)까지 반올림한 값이 됩니다.

예를 들어, $k = 2, C = 222$라면 $P = 2^{ 101 / 222 } = 1.37... $이고, 이 때 여러분이 2번 부분문제에서 받는 점수는 $10 \times (2+1) \times 1.37.. \approx 41.1$점이 됩니다.

구현 세부 사항

여러분은 `ainta()` 와 `sangsoo()` 를 구현한 소스 코드 `planet.c` 또는 `planet.cpp` 를 제출해야 합니다. 편의를 위해 예제 채점기를 내려받을 수 있습니다. 이 압축 파일에는 `grader.c(pp)` , `planet.h` , `planet.c(pp)` 가 들어있으며, 사용하시는 언어에 따라 적절히 컴파일하시면 됩니다. (모든 파일을 한꺼번에 컴파일해야 합니다) 반드시 코드의 윗부분에 `planet.h` 를 `#include` 해야 합니다. 예제 채점기는 표준 입력(`stdin`)으로부터 입력을 받아서 표준 출력(`stdout`)에 실행 결과를 출력합니다.

예제 채점기의 입력 형식

- 첫 번째 줄: 테스트 케이스의 수 T
- 다음 T개의 줄: r, c, p, q. ($0 \leq r, c, p, q < 2222$)
    - 이것은 승현이는 (r, c)에 있고, 상수는 (p, q)에 있는 상황을 나타냅니다. 이 위치는 서로 달라야 합니다.

예를 들어, 위 예시의 경우 아래와 같이 입력할 수 있습니다.

```
1
1 3 2 2
```