



두 사람 바자(Bazza)와 샤자(Shazza)가 다음 게임을 하고 있다. 게임판은 셀로 이루어진 그리드로 되어 있고,  $R$  개의 행이 차례로 번호가  $0, \dots, R-1$  로 매겨져 있고  $C$  개의 열이 차례로 번호가  $0, \dots, C-1$  로 매겨져 있다.  $(P, Q)$  는 행  $P$ , 열  $Q$  에 있는 셀을 나타낸다. 각 셀에는 음이 아닌 정수가 쓰여 있고, 게임이 시작될 때 모든 셀의 값은 0이다.

이 게임은 다음과 같이 진행된다. 게임 진행 중에, 바자는 다음 두 가지 중 하나를 할 수 있다.

- 셀  $(P, Q)$  의 값을 업데이트한다. 즉, 이 셀에 쓰여진 정수를 바꾼다.
- 샤자에게 주어진 직사각형 모양의 셀들 안의 모든 정수들의 (직사각형의 마주보는 양 모퉁이  $(P, Q)$ ,  $(U, V)$  를 포함) 최대공약수(GCD)를 계산하도록 요청한다.

바자는 최대  $N_U + N_Q$  번 위와 같은 일을 (셀의 값을  $N_U$  번 업데이트하고  $N_Q$  번 질의를 함) 한 다음에는 지루해서 크리켓을 하러 간다.

당신의 임무는 정답을 구하는 것이다.

## 예시

$R=2$  이고  $C=3$  이라고 하고, 바자가 다음과 같이 업데이트를 한다고 하자.

- 셀  $(0, 0)$  을 20으로 업데이트
- 셀  $(0, 2)$  를 15로 업데이트
- 셀  $(1, 1)$  을 12로 업데이트

20	0	15
0	12	0

이 결과는 위 그림과 같다. 그리고 바자는 다음 직사각형에 대해 최대공약수가 무엇인지 질의한다.

- 양 모퉁이 (0, 0), (0, 2): 이 직사각형 안의 세 정수는 20, 0, 15이고 최대공약수는 5이다.
- 양 모퉁이 (0, 0), (1, 1): 이 직사각형 안의 네 정수는 20, 0, 0, 12이고 최대공약수는 4이다.

바자가 다음과 같이 업데이트를 했다고 하자.

- 셀 (0, 1) 을 6으로 업데이트
- 셀 (1, 1) 을 14로 업데이트

20	6	15
0	14	0

새로운 그리드는 위 그림과 같다. 그리고, 바자는 다음 직사각형에 대해서 다시 최대공약수 GCD 를 질의한다.

- 양 모퉁이 (0, 0), (0, 2): 이 직사각형 안의 세 정수는 이제 20, 6, 15이고 최대공약수는 1이다.
- 양 모퉁이 (0, 0), (1, 1): 이 직사각형 안의 네 정수는 20, 6, 0, 14이고 최대공약수는 2이다.

여기까지 바자가 한 업데이트는  $N_U = 5$  회, 질의는  $N_Q = 4$  회이다.

---

## 구현

다음 조건을 만족하는 함수 `init()`, `update()`, `calculate()` 를 구현하여 제출하여야 한다.

당신을 돕기 위해, 컴퓨터에 있는 솔루션 템플릿 파일들 (`game.c`, `game.cpp` and `game.pas`) 각각에는 함수 `gcd2(x, y)` 가 구현되어 있다. 이 함수는 음이 아닌 정수  $x$  와  $y$  가 주어질 때 그 최대공약수를 계산한다. 만약  $x = y = 0$  인 경우 `gcd2(x, y)` 역시 0을 리턴한다.

이 함수의 동작 시간은 만점을 얻을 수 있는 데에 충분할만큼 빠르다. 최악의 경우라도 동작 시간은  $\log(X + Y)$  에 비례한다.

구현해야 하는 함수: **init()**

C/C++ `void init(int R, int C);`

Pascal `procedure init(R, C : LongInt);`

설명

당신은 이 함수를 구현하여 제출해야 한다.

그리드의 초기 크기가 이 함수를 통해 당신에게 전달되며, 당신이 필요로 하는 전역변수 및 자료구조를 이 함수 내에서 초기화할 수 있다. 이 함수는 단 한 번만 호출되며, `update()` 또는 `calculate()`가 입력되기 전에 호출될 것이다.

파라미터

- `R`: 행의 개수.
- `C`: 열의 개수.

구현해야 하는 함수: **update()**

C/C++ `void update(int P, int Q, long long K);`

Pascal `procedure update(P, Q : LongInt; K : Int64);`

설명

당신은 이 함수를 구현하여 제출해야 한다.

이 함수는 바자가 어떤 셀의 숫자를 바꿀 때 호출된다.

파라미터

- `P`: 그리드 셀의 행 번호 ( $0 \leq P \leq R-1$ ).
- `Q`: 그리드 셀의 열 번호 ( $0 \leq Q \leq C-1$ ).
- `K`: 이 그리드 셀이 가지게 되는 새로운 값 ( $0 \leq K \leq 10^{18}$ ). 이전에 가졌던 값과 같을 수 있다.

구현해야 하는 함수: `calculate()`

C/C++ `long long calculate(int P, int Q, int U, int V);`

Pascal `function calculate(P, Q, U, V : LongInt) : Int64;`

### 설명

당신은 이 함수를 구현하여 제출해야 한다.

이 함수는  $(P, Q)$  와  $(U, V)$  를 마주보는 모퉁이로 하는 직사각형 내에 포함된 모든 숫자들의 최대공약수를 계산하여야 한다. 이 직사각형의 범위는 테두리를 포함한다. 즉,  $(P, Q)$  와  $(U, V)$  가 포함되어 있어야 한다.

직사각형 범위 내에 모든 숫자가 0인 경우에는, 이 함수 역시 0을 리턴하여야 한다.

### 파라미터

- $P$ : 직사각형의 가장 왼쪽 위 셀의 행 번호 ( $0 \leq P \leq R-1$ ).
- $Q$ : 직사각형의 가장 왼쪽 위 셀의 열 번호 ( $0 \leq Q \leq C-1$ ).
- $U$ : 직사각형의 가장 오른쪽 아래 셀의 행 번호 ( $P \leq U \leq R-1$ ).
- $V$ : 직사각형의 가장 오른쪽 아래 셀의 열 번호 ( $Q \leq V \leq C-1$ ).
- 리턴값: 직사각형 내 모든 숫자들의 최대공약수 (단, 모든 숫자들이 0인 경우에는 0).

---

## 예제 세션

다음 예제 세션은 위의 예시를 나타낸 것이다:

함수 호출	리턴값
<code>init(2, 3)</code>	
<code>update(0, 0, 20)</code>	
<code>update(0, 2, 15)</code>	
<code>update(1, 1, 12)</code>	
<code>calculate(0, 0, 0, 2)</code>	5
<code>calculate(0, 0, 1, 1)</code>	4
<code>update(0, 1, 6)</code>	
<code>update(1, 1, 14)</code>	
<code>calculate(0, 0, 0, 2)</code>	1
<code>calculate(0, 0, 1, 1)</code>	2

---

## 제약 조건

- 시간 제한: 서브태스크를 참조하십시오.
- 메모리 제한: 서브태스크를 참조하십시오.
- $1 \leq R, C \leq 10^9$
- $0 \leq K \leq 10^{18}$ . 여기서  $K$ 는 바자가 그리드 셀에 써넣는 숫자이다.

---

## 서브태스크

서브태스크의 파라미터는 영어판 문제를 참조하십시오.

서브태스크	배점	R	C	$N_U$	$N_Q$	시간 제한	메모리 제한

---

## 테스트용 입력 형식

당신의 컴퓨터에 있는 샘플 그레이더는 입력을 파일 `game.in` 에서 읽어들이는데, 포맷은 다음과 같아야 한다.

- Line 1: `R C N`
- Next `N` lines: 일이 발생하는 순서대로 한 줄마다 일 하나씩

각 일은 다음 중 하나의 형식을 따라야 한다.

- `update(P, Q, K)` 의 표현: `1 P Q K`
- `calculate(P, Q, U, V)` 의 표현: `2 P Q U V`

예를 들어, 위의 예시는 다음과 같은 형식을 따라야 한다:

```
2 3 9
1 0 0 20
1 0 2 15
1 1 1 12
2 0 0 0 2
2 0 0 1 1
1 0 1 6
1 1 1 14
2 0 0 0 2
2 0 0 1 1
```

---

## 언어 유의사항

- C/C++ 당신의 프로그램은 `#include "game.h"` 명령어를 통해 헤더 파일을 추가시켜야 한다.
- Pascal 당신의 프로그램은 `unit Game` 를 정의해야 한다. 모든 배열의 인덱스는 `1` 이 아닌 `0` 부터 시작한다.

그리드 셀에 들어가는 숫자들이 매우 클 수 있기 때문에, C/C++ 사용자들은 `long long` 자료형을, Pascal 사용자들은 `Int64` 자료형을 이용하는 것을 권장한다.