Tower Breakers

Two players are playing a game of Tower Breakers! Player 1 always moves first, and both players always play optimally.The rules of the game are as follows:

- Initially there are n towers.
- Each tower is of height m.
- The players move in alternating turns.
- In each turn, a player can choose a tower of height x and reduce its height to y, where $1 \leq y < x$ and y evenly divides x.
- If the current player is unable to make a move, they lose the game.

Given the values of n and m, determine which player will win. If the first player wins, return 1. Otherwise, return 2.

Example. $n = 2$

$m = 6$

There are 2 towers, each 6 units tall. Player 1 has a choice of two moves:

- remove 3 pieces from a tower to leave 3 as $6 \bmod 3 = 0$

- remove 5 pieces to leave 1

Let Player 1 remove 3. Now the towers are 3 and 6 units tall.

Player 2 matches the move. Now the towers are both 3 units tall.

Now Player 1 has only one move.

Player 1 removes 2 pieces leaving 1. Towers are 1 and 2 units tall.

Player 2 matches again. Towers are both 1 unit tall.

Player 1 has no move and loses. Return 2.

Function Description

Complete the towerBreakers function in the editor below.

towerBreakers has the following paramter(s):

- int n: the number of towers
- int m: the height of each tower

Returns

- int: the winner of the game

Input Format

The first line contains a single integer t, the number of test cases.

Each of the next t lines describes a test case in the form of 2 space-separated integers, n and m.

Constraints

- $1 \leq t \leq 100$
- $1 \leq n, m \leq 10^6$

Sample Input

```
STDIN Function
----- --------
2  t=2
22  n=2,m=2
14  n=1,m=4
```

Sample Output

```
2
```

Author forthright48

Difficulty Easy

Max Score 100

Submitted By 5229

NEED HELP?

View discussions

View editorial

View top submissions

RATE THIS CHALLENGE

MORE DETAILS

Download problem statement

Download sample test cases

Suggest Edits

```
1
```

Explanation

We'll refer to player $\mathbf{1}$ as $\boldsymbol{P1}$ and player $\mathbf{2}$ as $\boldsymbol{P2}$

In the first test case, $\boldsymbol{P1}$ chooses one of the two towers and reduces it to $\mathbf{1}$. Then $\boldsymbol{P2}$ reduces the remaining tower to a height of $\mathbf{1}$. As both towers now have height $\mathbf{1}$, $\boldsymbol{P1}$ cannot make a move so $\boldsymbol{P2}$ is the winner.

In the second test case, there is only one tower of height $\mathbf{4}$. $\boldsymbol{P1}$ can reduce it to a height of either $\mathbf{1}$ or $\mathbf{2}$. $\boldsymbol{P1}$ chooses $\mathbf{1}$ as both players always choose optimally. Because $\boldsymbol{P2}$ has no possible move, $\boldsymbol{P1}$ wins.

Change Theme Language Rust

```rust
use std::env;
use std::fs::File;
use std::io::{self, BufRead, Write};

/*
 * Complete the 'towerBreakers' function below.
 *
 * The function is expected to return an INTEGER.
 * The function accepts following parameters:
 *  1. INTEGER n
 *  2. INTEGER m
 */

fn towerBreakers(n: i32, m: i32) -> i32 {

}

fn main() {
    let stdin = io::stdin();
    let mut stdin_iterator = stdin.lock().lines();

    let mut fptr = File::create(env::var("OUTPUT_PATH").unwrap()).unwrap();

    let t = stdin_iterator.next().unwrap().unwrap().trim().parse::<i32>().unwrap();

    for _ in 0..t {
        let first_multiple_input: Vec<String> = stdin_iterator.next().unwrap().unwrap()
            .split(' ')
            .map(|s| s.to_string())
            .collect();
```

Line: 41 Col: 1

Upload Code as File Test against custom input Run Code Submit Code