Problem Title

Nim (misère version) — determine if first player has a forced win
Company: Google

Scenario & Description

The game of Nim is played on several heaps of stones. Two players alternate turns. On a turn a player removes **one or more** stones from **exactly one** heap. In this variant the player **who takes the last stone loses** (this is called *misère Nim*).

Given three non-zero starting heap sizes `[a, b, c]`, decide whether the **first player** (the player who moves first) has a forced win assuming both players play optimally.

Input Format

- A list/array of three positive integers: `[a, b, c]` (each > 0).

Output Format

- Return `True` if the first player has a forced win, otherwise `False`.

Examples

- Input: `[3, 4, 5]`
  Output: `True`
  Explanation: Under optimal play first player can force a win.
- Input: `[1, 1, 1]`
  Output: `False`
  Explanation: With three heaps of 1, first player loses under misère rules.
- Input: `[1, 1, 2]`
  Output: `True`

Key Observation (Theory — Misère Nim)

For misère Nim the optimal-play rule is:

- If **all heaps have size 1** (i.e., every heap is 1), then:
  - First player wins **iff** the number of heaps is **even**.
  - (If number of heaps is odd → first player loses.)

- Otherwise (there exists a heap with size ≥ 2), play is the **same as normal Nim**:
    - Compute the XOR (nim-sum) of heap sizes.
    - If `nim-sum != 0` → first player has a winning strategy.
    - If `nim-sum == 0` → first player loses.

This well-known result is the standard misère Nim solution.

Why? Intuition:

- When all heaps are size 1, each move removes exactly one heap; since last removal loses, parity of heaps determines outcome.
- When some heap ≥ 2 exists, you can force play into positions where the parity-of-ones trick no longer matters — nim-sum determines winning positions.

Algorithm

1. Let `a, b, c` be the heap sizes.
2. If `a == b == c == 1` (or in general **all** heaps equal 1):
    - Return `True` if number of heaps is even, else `False`.
    - For three heaps: return `False` (since 3 is odd).
3. Else:
    - Compute `xor = a ^ b ^ c`.
    - If `xor != 0` → return `True`
    - Else → return `False`

Time complexity: `O(1)` for three heaps (generally `O(n)` for `n` heaps).
Space: `O(1)`.

Practice Links:

- [LeetCode – Nim Game]
- [GeeksforGeeks – Game of Nim]

Video Solution:

- [YouTube – Nim Game Explanation]