Problem 3379: Transformed Array

Problem Information

Difficulty: Easy
Acceptance Rate: 57.22%
Paid Only: No
Tags: Array, Simulation

Problem Description

You are given an integer array `nums` that represents a circular array. Your task is to create a new array `result` of the **same** size, following these rules:

For each index `i` (where `0 <= i < nums.length`), perform the following **independent** actions:

* If `nums[i] > 0`: Start at index `i` and move `nums[i]` steps to the **right** in the circular array. Set `result[i]` to the value of the index where you land. * If `nums[i] < 0`: Start at index `i` and move `abs(nums[i])` steps to the **left** in the circular array. Set `result[i]` to the value of the index where you land. * If `nums[i] == 0`: Set `result[i]` to `nums[i]`.

Return the new array `result`.

Note: Since `nums` is circular, moving past the last element wraps around to the beginning, and moving before the first element wraps back to the end.

Example 1:

Input: nums = [3,-2,1,1]

Output: [1,1,1,3]

Explanation:

* For `nums[0]` that is equal to 3, If we move 3 steps to right, we reach `nums[3]`. So `result[0]` should be 1. * For `nums[1]` that is equal to -2, If we move 2 steps to left, we reach

`nums[3]`. So `result[1]` should be 1. * For `nums[2]` that is equal to 1, If we move 1 step to right, we reach `nums[3]`. So `result[2]` should be 1. * For `nums[3]` that is equal to 1, If we move 1 step to right, we reach `nums[0]`. So `result[3]` should be 3.

Example 2:

Input: nums = [-1,4,-1]

Output: [-1,-1,4]

Explanation:

* For `nums[0]` that is equal to -1, If we move 1 step to left, we reach `nums[2]`. So `result[0]` should be -1. * For `nums[1]` that is equal to 4, If we move 4 steps to right, we reach `nums[2]`. So `result[1]` should be -1. * For `nums[2]` that is equal to -1, If we move 1 step to left, we reach `nums[1]`. So `result[2]` should be 4.

Constraints:

* `1 <= nums.length <= 100` * `-100 <= nums[i] <= 100`

Code Snippets

C++:

```
class Solution {
public:
vector<int> constructTransformedArray(vector<int>& nums) {

}
};
```

Java:

```
class Solution {
public int[] constructTransformedArray(int[] nums) {

}
}
```

Python3:

```
class Solution:
def constructTransformedArray(self, nums: List[int]) -> List[int]:
```