# 题目

给你一个长度为n的整数数组nums。请你构建一个长度为2n的答案数组ans，数组下标从0开始计数，对于所有0 <= i < n的i，满足下述所有要求：

ans[i] == nums[i]

ans[i + n] == nums[i]

具体而言，ans 由两个 nums 数组串联形成。

返回数组 ans 。

示例 1：

输入：nums = [1,2,1]

输出：[1,2,1,1,2,1]

解释：数组 ans 按下述方式形成：

- ans = [nums[0],nums[1],nums[2],nums[0],nums[1],nums[2]]

- ans = [1,2,1,1,2,1]

示例 2：

输入：nums = [1,3,2,1]

输出：[1,3,2,1,1,3,2,1]

解释：数组 ans 按下述方式形成：

- ans = [nums[0],nums[1],nums[2],nums[3],nums[0],nums[1],nums[2],nums[3]]

- ans = [1,3,2,1,1,3,2,1]

提示：

n == nums.length

1 <= n <= 1000

1 <= nums[i] <= 1000

# 分析

## 方法一：模拟

思路：

我们顺序遍历修改前nums数组的元素，并按顺序添加至nums数组的尾部。最终，修改后的nums数组即为串联形成的数组，我们返回该数组作为答案。

代码：

class Solution {

public:

vector<int> getConcatenation(vector<int>& nums) {

int n = nums.size();

for (int i = 0; i < n; ++i){

nums.push\_back(nums[i]);

}

return nums;

}

};

复杂度分析

时间复杂度：O(n)，其中n为nums的长度。即为遍历与串联的时间复杂度。

空间复杂度：O(1)，输出数组不计入空间复杂度。