PROBLEM-1

Two Sum

Easy

21805742Add to ListShare

Given an array of integers `nums` and an integer `target`, return *indices of the two numbers such that they add up to `target`*.

You may assume that each input would have ***exactly* one solution**, and you may not use the *same* element twice.

You can return the answer in any order.

Example 1:

```
Input: nums = [2,7,11,15], target = 9

Output: [0,1]

Output: Because nums[0] + nums[1] == 9, we return [0, 1].
```

Example 2:

```
Input: nums = [3,2,4], target = 6

Output: [1,2]
```

Example 3:

```
Input: nums = [3,3], target = 6

Output: [0,1]
```

Constraints:

- `2 <= nums.length <=` 10^4
- -10^9 `<= nums[i] <=` 10^9
- -10^9 `<= target <=` 10^9
- **Only one valid answer exists.**

Follow-up: Can you come up with an algorithm that is less than $O(n^2)$ time complexity?

PROBLEM-1

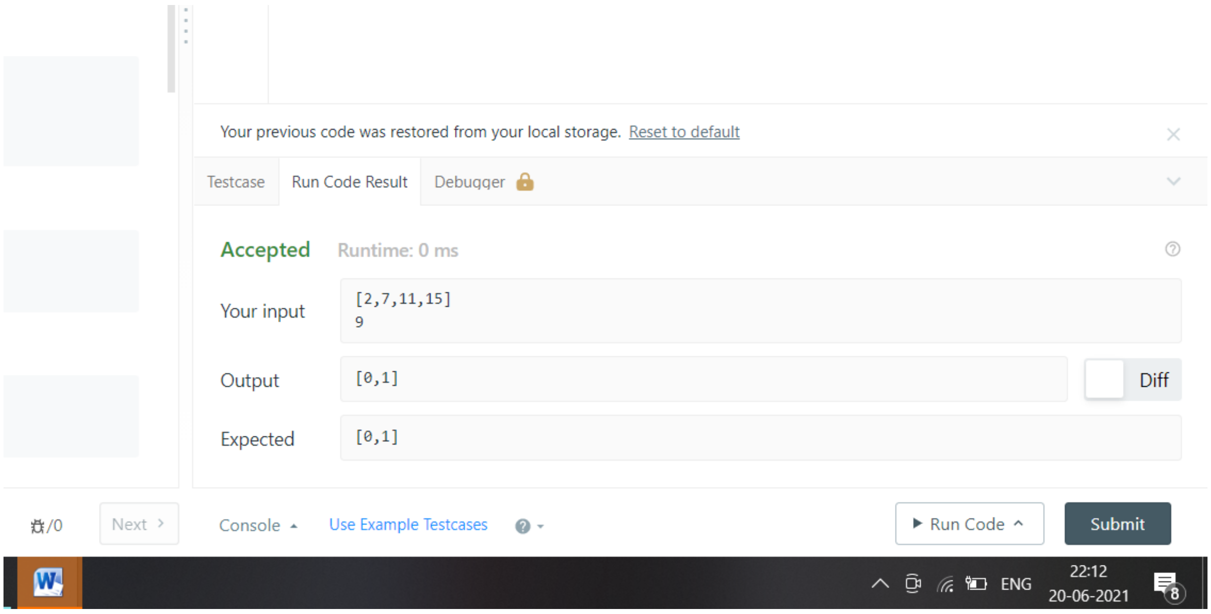
SOLUTION

```
class Solution {
  public int[] twoSum(int[] nums, int target) {
    for(int i=0;i<nums.length;i++)
    {
      for(int j=i+1;j<nums.length;j++)
        {
          if(target==nums[i]+nums[j]){
            return new int[] {i,j};
          }
        }
    }

    throw new IllegalArgumentException("No two sum solution");
  }
}
```

OUTPUT

PROBLEM-1



Your previous code was restored from your local storage. Reset to default

Testcase | Run Code Result | Debugger

Accepted Runtime: 0 ms

Your input
```
[2,7,11,15]
9
```

Output
```
[0,1]
```

Expected
```
[0,1]
```