Understanding Vector Assignment in C++

Problem Statement: Rotate Array

Given: An integer array `nums`, rotate the array to the right by `k` steps.

Example:
Input: `nums = [1,2,3,4,5,6,7]`, `k = 3`
Output: `nums = [5,6,7,1,2,3,4]`

Code Implementation

```
class Solution {
public:
    void rotate(vector<int>& nums, int k) {
        vector<int> temp(nums.size());  // Creates a new vector of same size as
nums

        // Storing values in temp at rotated positions
        for(int i = 0; i < nums.size(); i++) {
            temp[(i + k) % nums.size()] = nums[i];
        }

        // Copying temp vector into nums
        nums = temp;
    }
};
```

Common Doubts and Explanations

1 What does `vector<int> temp(nums.size());` do?

- This line creates a new vector `temp` of **the same size as** `nums`.
- Initially, all elements in `temp` are set to **default values** (which is `0` for integers).
- Example:

```
vector<int> temp(nums.size()); // If nums = [1,2,3,4,5], temp = [0, 0, 0, 0,
0]
```

2 How does `nums = temp;` copy all elements without a loop?

- In C++, when assigning one vector to another (`nums = temp;`), it **automatically copies** all elements.
- This is because `std::vector` **overloads the assignment operator (`=`)** to handle deep copying.

- **Internally, it loops over `temp` and assigns values to `nums`.**

3️⃣ What happens under the hood when `nums = temp;` is executed?

- **Memory Reallocation (if required):**
  - If `nums` has enough capacity, it **reuses the existing memory**.
  - If `temp` is larger, `nums` **allocates new memory** and copies elements.
- **Element-wise Copying:**
  - Each element of `temp` is copied into `nums` using an **implicit loop**.

Example:

```
std::vector<int> temp = {5, 6, 7, 1, 2, 3, 4};
std::vector<int> nums = {1, 2, 3, 4, 5, 6, 7};
nums = temp;  // All elements are copied
```

4️⃣ Is `nums = temp;` better than a manual loop?

☑ **Yes**, because it is optimized by the STL and handles memory efficiently.
☑ No need to manually iterate over elements.
☑ Cleaner and more readable code.

Key Takeaways

✔ `vector<int> temp(nums.size());` initializes a vector of the same size as `nums`.
✔ `nums = temp;` **copies all elements** internally without an explicit loop.
✔ The `std::vector` class **manages memory and iteration automatically**.
✔ This is an efficient way to copy one vector to another in C++.

◇ **Would you like a deep dive into how `std::vector` handles memory under the hood?** 🚀