# Find the First Negative Integer in Every K Sized Window of Array CodeStudio

This program finds the first negative element in each k-sized window of a given array using two different approaches: a brute force approach and an optimized approach using a deque (double-ended queue).

**Approach 1: Function to find the first negative element in each k-sized window using a brute force approach**

- **Description:** This approach uses a nested loop to find the first negative element in each k-sized window.
- **Steps:**
    1. Initialize a vector **ans** to store the results, initially filled with zeros.
    2. Iterate through all possible k-sized windows in the input array.
    3. For each window, iterate through its elements and check for the first negative element. If found, store it in the **ans** vector.
- **Time Complexity: O(n * k), where n is the size of the input array and k is the window size. This is because, in the worst case, for each of the n-k+1 windows, we iterate through k elements.**
- **Space Complexity: O(n-k+1), as the ans vector stores the results for each window.**

**Approach 2: Function to find the first negative element in each k-sized window using a dequ**e

- **Description:** This approach optimizes the process using a deque to efficiently find the first negative element in each k-sized window.
- **Steps:**
    1. Initialize a deque **negativeindices** to store indices of negative elements.
    2. Initialize a vector **ans** to store the results.
    3. Iterate through the input array, and for each element:
        - Remove indices from the front of the deque that are no longer in the current window (i.e., outside the range of k elements).
        - If the current element is negative, add its index to the deque.

    - Once the window size reaches k, find and store the first negative element in the deque (if it exists) or store 0 (if no negative element is found).

- **Time Complexity: O(n), where n is the size of the input array. We iterate through the array once, and each element is added and removed from the deque at most once.**
- **Space Complexity: O(k), as the deque stores at most k indices.**