Sliding Window - First Negative Number

Problem Statement:

Given an array v[] of size n, and an integer k, find the first negative number in every window of size k that slides through the array. If a window does not contain any negative number, return 0 for that window.

Example:

```
Input:
v = {0, -1, -2, 3, -5, 6, 4, 7, -8};
k = 3;

Output:
-1 -1 -2 -5 -5 0 -8
```

Approach and Thought Process:

We use a queue to store indices of negative numbers. As the window moves, we remove out-of-bound indices. The front of the queue always holds the first negative number in the window.

Code Implementation:

```
#include<iostream>
#include<queue>
#include<vector>
using namespace std;

vector<int> near_next_negative(vector<int> &v, int k) {
    int n = v.size();
    queue<int> q;
    int i = 0;

    while(i <= v.size() - 1) {
        if(v[i] < 0) q.push(i);
        i++;
    }
    vector<int> ans;
    for(int i = 0; i <= n - k; i++) {
        while(q.size() > 0 && q.front() < i) q.pop();
                  if(q.size()  >  0  &&  q.front()  >=  i  &&  q.front()  <  i  +  k)
ans.push_back(v[q.front()]);
        else ans.push_back(0);
    }
    return ans;
}

int main() {
    vector<int> v;
    int n;
    cout << "Enter the number of elements: ";
    cin >> n;

    for(int i = 0; i <= n - 1; i++) {
        int temp;
        cin >> temp;
        v.push_back(temp);
    }

    vector<int> ans = near_next_negative(v, 3);
    for(int i = 0; i <= ans.size() - 1; i++) cout << ans[i] << " ";
    return 0;
}
```

Time Complexity Analysis:

Each element is processed at most once. Overall complexity: O(n).

Dry Run of Given Test Case:

```
Input:
v = {0, -1, -2, 3, -5, 6, 4, 7, -8}; k = 3;

Window       | First Negative
-------------|---------------
{0, -1, -2}  | -1
{-1, -2, 3}  | -1
{-2, 3, -5}  | -2
```

```
{3, -5, 6}   | -5

{-5, 6, 4}   | -5

{6, 4, 7}    | 0

{4, 7, -8}   | -8


Output: -1 -1 -2 -5 -5 0 -8
```