Order of removal in C++

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

class OrderOfRemoval {
public:
    static int orderOfRemoval(vector<int>& arr) {
        int n = arr.size();
        sort(arr.begin(), arr.end()); // Sorting the array

        int ans = 0;
        for (int i = 0; i < n; i++) {
            int temp = arr[i] * (n - i);
            ans += temp;
        }

        return ans;
    }
};

int main() {
    // Hardcoded input array
    vector<int> arr = {1, 2, 3, 4, 5};
    int n = arr.size();

    // Calling orderOfRemoval function to calculate the
order of removal
    int result = OrderOfRemoval::orderOfRemoval(arr);

    // Printing the result
    cout << "Order of removal: " << result << endl;

    return 0;
}
```

Let's perform a **detailed dry run** of your orderOfRemoval function using the input array:

arr = {1, 2, 3, 4, 5}

Step-by-step Dry Run:

1. **Sort the array**: The input array {1, 2, 3, 4, 5} is already sorted, so no changes are made.

   Sorted array: {1, 2, 3, 4, 5}

2. **Initialize Variables**:
   - n = arr.size() = 5
   - ans = 0 (This will hold the final result)
3. **Iterate and calculate the result**: For each element arr[i] in the array, the contribution of that element to the ans is calculated by multiplying arr[i] with the remaining elements (i.e., arr[i] * (n - i)).

Dry Run Table:

i	arr[i]	n - i	arr[i] * (n - i)	Cumulative ans
0	1	5	1 * 5 = 5	0 + 5 = 5
1	2	4	2 * 4 = 8	5 + 8 = 13
2	3	3	3 * 3 = 9	13 + 9 = 22
3	4	2	4 * 2 = 8	22 + 8 = 30
4	5	1	5 * 1 = 5	30 + 5 = 35

Final Result:

After the loop finishes, the value of ans is 35.

So, the output of the program is:

Order of removal: 35

```
Order of removal: 35
```