3_array_processing.cpp

```cpp
/*
  Program for Question 3: Array Processing (Eliminatino of three largest values)
  We are given an array a of length n. Our task is to write a function reduce(a[], n) that
  deletes all values that are equal to the three largest numbers in the array.

  For example, if a = { 9,1,1,6,7,1,2,3,3,5,6,6,6,6,7,9 }
    reduce(a, n) should change array a to { 1,1,1,2,3,3,5 }
    the three largest numbers are 6,7, and 9 and were removed from the array

  My solution is optimal because it has a runtime complexity of O(n).
  It does 1 initial pass through the array to find the top 3 largest numbers.
  It then stores these numbers into a set which have a lookup complexity of O(1).

  The algorithm then loops through the array again and checks if the current value
  is equal to one of the 3 largest numbers. If it is, the value is deleted, then the next value
  is shifted forward based on how many times an element in the array has been deleted.
*/

#include <iostream>
#include <vector>
#include <unordered_set>
#include <limits.h>

std::unordered_set<int> find3Largest(std::vector<int> a, int n) {
    // max1 > max2 > max3
    int max1 = INT_MIN;
    int max2 = INT_MIN;
    int max3 = INT_MIN;

    std::unordered_set<int> maximums;

    for (int& number : a) {
        if (number > max1) {
            max3 = max2;
            max2 = max1;
            max1 = number;
        }
        else if (number > max2 && number < max1) {
            max3 = max2;
            max2 = number;
        }
        else if (number > max3 && number < max2) {
            max3 = number;
        }
    }

    // Insert into a set
    maximums.insert(max1);
    maximums.insert(max2);
    maximums.insert(max3);

    return maximums;
```

```
53  }
54
55  std::vector<int> reduce(std::vector<int> &a, int n) {
56      std::unordered_set<int> maximums = find3Largest(a, n);
57      int deletions = 0;
58
59      for (int i = 0; i < n; i++) {
60          if (maximums.find(a[i]) != maximums.end())  // Maximum found
61              deletions++;                  // Value is "deleted"
62          else
63              a[i-deletions] = a[i];          // If not maximum, shift it forward in the array based on
    number of deleetions
64      }
65
66      a.resize(a.size() - deletions);        // Array is now (n - deletions) shorter
67
68      return a;
69
70  }
71
72  double sec() {
73      return double(clock())/double(CLOCKS_PER_SEC);
74  }
75
76  void timeTestCase() {
77      std::vector<int> a = { 9,1,1,6,7,1,2,3,3,5,6,6,6,6,7,9 };
78
79      std::cout << "a = ";
80      for (int& value : a) {
81          std::cout << value << " ";
82      }
83      std::cout << "n = " << a.size() << ", reduce(a, n) = ";
84
85      double T1 = sec();
86      reduce(a, a.size());
87      double T2 = sec();
88
89      for (int& value : a) {
90          std::cout << value << " ";
91      }
92      std::cout << std::endl;
93
94
95      std::cout << "Run time of reduce(a, n): " << T2 - T1 << "s" << std::endl << std::endl;
96  }
97
98  int main() {
99      timeTestCase();
100     return 0;
101 }
```