1_plateau.cpp

```cpp
/*
  Program for Question 1: Plateau Program (Max sequence length)
  We are given an array a of length n. This array contains sequences of
  repeating numbers. Our task is to write a function maxLen(a[], n) that
  returns the length of the longest sequence of repeating numbers.

  For example, if a = { 1,1,1,2,3,3,5,6,6,6,6,7,9 }, n = 3, maxLen(a, 13) should return 4
    since the longest sequence: 6,6,6,6 has a length of 4

  My solution is optimal because it has a runtime complexity of O(n).
  When it loops through the array, it keeps count of the number of repeated
  values until it finds the next different value. If this count is greater
  than the max, it will become the new max
*/

#include <iostream>

double sec() {
  return double(clock())/double(CLOCKS_PER_SEC);
}

/*
 * Return the length of the longest sequence of identical numbers
 * in array a of size n
 */
int maxLen(int a[], int n) {
  int maxLength = 1;
  int currentLength = 1;      // Start after the first character

  for (int i = 1; i < n; i++) {
    if (a[i] == a[i-1])     // Sequence is still increasing in length
      currentLength++;
    else
      currentLength = 1;    // Start of a new sequence

    if (currentLength > maxLength)  // This sequence is now the longest
      maxLength = currentLength;
  }

  return maxLength;
}

int main() {
  int arr1[13] = { 1,1,1,2,3,3,5,6,6,6,6,7,9 };

  double T1 = sec();
  for (int i = 0; i < 1000000; i++) {
    maxLen(arr1, 13);
  }
  double T2 = sec();

  std::cout << "a = ";
```

```
53    for (int& value : arr1) {
54      std::cout << value << " ";
55    }
56    std::cout << "n = " << 13 << std::endl;
57    std::cout << "maxLen(a, 13) = " << maxLen(arr1, 13) << std::endl;
58    std::cout << "Run time of maxlen(arr[], n) repeated 1000000 times: " << T2-T1 << "s" <<
  std::endl;
59
60    return 0;
61  }
62
```