```
#include <algorithm>
class Sequence {
private:
  int length;
  int* pseq;

public:
  Sequence() : length(10), pseq(new int[length]) {
    for (int i = 0; i < length; i++) {
      pseq[i] = 0;
    }
  }

  Sequence(int lengthVal, int n1=0, int n2=0, int n3=0, int n4=0, int n5=0, int
n6=0, int n7=0,
      int n8=0, int n9=0, int n10=0) : length(lengthVal), pseq(new int[length]) {
    int nums[10] = {n1, n2, n3, n4, n5, n6, n7, n8, n9, n10};
    for (int i = 0; i < length; i++) {
      pseq[i] = nums[i];
    }
  }

  Sequence(const Sequence& s) : length(s.length), pseq(new int[length]) {
    for (int i = 0; i < length; i++) {
      pseq[i] = s.pseq[i];
    }
  }

  ~Sequence() {
    delete[] pseq;
  }

  int getLength() const {
    return length;
  }

  int* getSeq() const {
    return pseq;
}

void Sort(int n) {
    std::sort(pseq, pseq + n);
}

int RemoveDuplicates() {
    if (length <= 1) {
        return length;
    }

    int uniqueCount = 1;
    for (int i = 1; i < length; i++) {
        if (pseq[i] != pseq[uniqueCount - 1]) {
            pseq[uniqueCount++] = pseq[i];
        }
    }

    return uniqueCount;
}

void Rotate(int steps) {
    if (length <= 1 || steps <= 0) {
        return;
    }

    steps %= length;

    // Reverse the entire sequence
    std::reverse(pseq, pseq + length);

    // Reverse the first 'steps' elements
    std::reverse(pseq, pseq + steps);

    // Reverse the remaining elements
```