```cpp
#include <iostream>
#include <ctime>
#include <vector>
#include <stdint.h>
#include <stdio.h>
#include <iostream>
#include <string>
#include <algorithm>
#include <chrono>


using namespace std;

void radixSort(uint16_t * A, uint16_t digit, int len);

int main() {
    int arrSize[11] = {100, 500, 1000, 5000, 10000, 50000, 100000, 500000, 1000000, 5000000, 10000000};
    printf("Radix Sort\n");
    for(int i = 0; i < 11; ++i) {
        // Create the Array to be sorted with random data
        srand(clock());
        uint16_t * A = new uint16_t[arrSize[i]];
        for(int j = 0; j < arrSize[i]; ++j) {
            A[j] = rand();
        }
        // Get the start time
        auto init = chrono::high_resolution_clock::now();
        // Run the algorithm
        radixSort(A, 16, arrSize[i]);
        // Get the end time
        auto end = chrono::high_resolution_clock::now();
        // Calculate the elepsed time
        auto duration = end - init;
        int sec = chrono::duration_cast<chrono::seconds>(duration).count();
        int nano = chrono::duration_cast<chrono::nanoseconds>(duration).count() % 1000000000;

        printf("%i, %i.%09i\n", arrSize[i], sec, nano);
        // Make sure the output was sorted
        for (int j = 1; j < arrSize[i]; j++) {
            if (A[j] < A[j - 1]) {
                cout << "WRONG " << j;
            }
        }
        delete[] A;
    }
    // Wait for user input.
    string tmp;
    getline(cin, tmp);
}

```

```
51  void radixSort(uint16_t * A, uint16_t digit, int len) {
52      if (len > 1 && digit > 0) {
53          int zeroTop = 0;
54          int oneTop = 1;
55          for (int i = 0; i < len; ++i) {
56              // Check if the digit bit is zero
57              if (!(A[i] & (1 << digit - 1))) {
58                  // Put the item into the zero "Bin"
59                  swap(A[zeroTop], A[i]);
60                  zeroTop++;
61                  oneTop++;
62              }
63              else {
64                  // Put the item into the one "Bin"
65                  swap(A[zeroTop], A[i]);
66                  oneTop++;
67              }
68
69          }
70
71          // Sort the next digit for items in the zero "Bin" and the one "Bin"
72          radixSort(A, digit - 1, zeroTop);
73          radixSort(A + zeroTop, digit - 1, len - zeroTop);
74      }
75  }
76
```