```cpp
#include <iostream>
#include <vector>
#include <chrono>
#include <random>

using namespace std;

bool get_line(const string& prompt, string& userinput){
    cout << prompt;
    getline(cin, userinput);
    return !userinput.empty();
}

void max_heapify(vector<int>& a, int i, int n){
    while((2 * i + 1) < n){
        int exch_idx = 2 * i + 1;
        // check to see if a right child exists and if so check if its bigger than the left child
        if(exch_idx + 1 < n && a[exch_idx] < a[exch_idx + 1])
            exch_idx++;
        if(a[i] > a[exch_idx]) break;
        swap(a[i], a[exch_idx]);
        i = exch_idx;
    }
}

void selection_sort(vector<int>& a){
    for(int i = 0;i < a.size();i++){
        int min_idx = i;
        for(int j = i + 1;j < a.size();j++)
            if(a[min_idx] > a[j]) min_idx = j;
        swap(a[i], a[min_idx]);
    }
}

void build_MaxHeap(vector<int>& a){
    for(int i = a.size() / 2;i >= 0;i--){
        max_heapify(a, i, a.size());
    }
}

void heap_sort(vector<int>& a){
    build_MaxHeap(a);
    int n = a.size();
    while(n > 1){
        swap(a[0], a[--n]);
        max_heapify(a, 0, n);
    }
}


void display(const vector<int>& a){
    for(int e : a) cout << e << " ";
    cout << endl;
}

int main() {
    string userinput;
    unsigned int seed = chrono::steady_clock::now().time_since_epoch().count();
    uniform_int_distribution<int> uniform_int_distribution(-100, 100);
    mt19937 gen(seed);
    while(get_line("(part a) Enter a positive integer n: ", userinput)) {
        int n = stoi(userinput);
        vector<int> a;
        for(int i = 0;i < n;i++)
            a.push_back(uniform_int_distribution(gen));

        get_line("Enter the number of trials: ", userinput);
        int trials = stoi(userinput);
        vector<double> heap_sort_trials;
        vector<double> selection_sort_trials;
        for(int i = 0;i < trials;i++){
            vector<int> heap_copy = a;
            auto start = chrono::steady_clock::now();
            heap_sort(heap_copy);
            auto end = chrono::steady_clock::now();
            chrono::duration<double> elapsed_seconds = end - start;
            heap_sort_trials.push_back(elapsed_seconds.count());
```

```
   78:
   79:             vector<int> selection_copy = a;
   80:             start = chrono::steady_clock::now();
   81:             selection_sort(a);
   82:             end = chrono::steady_clock::now();
   83:             elapsed_seconds = end - start;
   84:             selection_sort_trials.push_back(elapsed_seconds.count());
   85:         }
   86:         double heap_sort_avg = accumulate(heap_sort_trials.begin(), heap_sort_trials.end(), 0.0) / trials;
   87:         double selection_sort_avg = accumulate(selection_sort_trials.begin(), selection_sort_trials.end(), 0.0
) / trials;
   88:
   89:         cout << "The average runtime for heap sort is: " << heap_sort_avg << " seconds" << endl;
   90:         cout << "The average runtime for selection sort is: " << selection_sort_avg << " seconds" << endl;
   91:     }
   92:     while(get_line("(part b) Press any key followed by enter to continue: ", userinput)){
   93:         vector<int> A;
   94:         for(int i = 0;i < 10;i++)
   95:             A.push_back(uniform_int_distribution(gen));
   96:         display(A);
   97:         heap_sort(A);
   98:         display(A);
   99:     }
  100: }
```