```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() - 1) / 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);
    }
}
```

```
49
50
51 void display(const vector<int>& a){
52     for(int e : a) cout << e << " ";
53     cout << endl;
54 }
55
56 int main() {
57     string userinput;
58     unsigned int seed = chrono::steady_clock::now().
   time_since_epoch().count();
59     uniform_int_distribution<int> uniform_int_distribution
   (-100, 100);
60     mt19937 gen(seed);
61     while(get_line("(part a) Enter a positive integer n: "
   , userinput)) {
62         int n = stoi(userinput);
63         vector<int> a;
64         for(int i = 0;i < n;i++)
65             a.push_back(uniform_int_distribution(gen));
66
67         get_line("Enter the number of trials: ", userinput
   );
68         int trials = stoi(userinput);
69         vector<double> heap_sort_trials;
70         vector<double> selection_sort_trials;
71         for(int i = 0;i < trials;i++){
72             vector<int> heap_copy = a;
73             auto start = chrono::steady_clock::now();
74             heap_sort(heap_copy);
75             auto end = chrono::steady_clock::now();
76             chrono::duration<double> elapsed_seconds = end
    - start;
77             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 }
101
```