```cpp
/*
Shaya Ahmed
CS 3700                        Bubble Sort & Quick Sort
*/
#include <iostream>
#include <iomanip>
#include <fstream>
#include <cstdlib>
#include <ctime>
#include <algorithm>
#include <vector>

using namespace std;

void bubbleSort(vector<int>, int);
void quickSort(vector<int>, int);
void q_sort(vector<int>, int, int);

int main()
{
        srand(time(0));
        clock_t start, end;
        vector<int> data(10000);
        int Size = data.size();
        double duration;

        // Best Case
        ofstream bestCaseFile("best.txt");
        for(int i = 1; i <= Size; i++){
                bestCaseFile << i << endl;
        }
        bestCaseFile.close();

        // Worst Case
        ofstream worstCaseFile("worst.txt");
        for(int i = Size; i >= 1; i--){
                worstCaseFile << i << endl;
        }
        worstCaseFile.close();

        // Avg Case
        ofstream avgCaseFile("avg.txt");
        for(int i = 1; i <= Size; i++){
                avgCaseFile << (rand() %Size + 1) << endl;
        }
        avgCaseFile.close();

        // read from bestCase File to vector
        ifstream myFile1("best.txt");
        if(!myFile1.is_open()){
                cout << "File does not exist!...\n";
        }
        else{
                while(!myFile1.eof()){
                        for (int i = 0; i < Size; i++){
                                myFile1 >> data[i];
                        }
                }
                start = clock();
                bubbleSort(data, Size);
                end = clock();

                duration = ((double)(end-start))/CLOCKS_PER_SEC;
                cout << "\nBubble Sort Best Case took " << setprecision(3) << fixed << duration << endl;

                start = clock();
                quickSort(data, Size);
                end = clock();

                duration = ((double)(end-start))/CLOCKS_PER_SEC;
                cout << "Quick Sort Best Case took " << setprecision(3) << fixed << duration << endl;
        }
        myFile1.close();


        // read from worstCase File to vector
        ifstream myFile2("worst.txt");
        if(!myFile2.is_open()){
                cout << "File does not exist!...";
        }
        else{
                while(!myFile2.eof()){
                        for(int i = 0; i < Size; i++){
                                myFile2 >> data[i];
                        }
                }
                start = clock();
                bubbleSort(data, Size);
                end = clock();

                duration = ((double)(end-start))/CLOCKS_PER_SEC;
                cout << "\nBubble Sort Worst Case took " << setprecision(3) << fixed << duration << endl;

                start = clock();
                quickSort(data, Size);
                end = clock();

                duration = ((double)(end-start))/CLOCKS_PER_SEC;
                cout << "Quick Sort Worst Case took " << setprecision(3) << fixed << duration << endl;
        }
        myFile2.close();

        // read from avgCase File to vector
        ifstream myFile3("avg.txt");
        if(!myFile3.is_open()){
                cout << "File does not exist!...";
        }
        else{
                while(!myFile3.eof()){
                        for(int i = 0; i < Size; i++){
                                myFile3 >> data[i];
                        }
                }
                start = clock();
                bubbleSort(data, Size);
                end = clock();

                duration = ((double)(end-start))/CLOCKS_PER_SEC;
                cout << "\nBubble Sort Average Case took " << setprecision(3) << fixed << duration << endl;

                start = clock();
                quickSort(data, Size);
                end = clock();

                duration = ((double)(end-start))/CLOCKS_PER_SEC;
                cout << "Quick Sort Average Case took " << setprecision(3) << fixed << duration << endl;
        }
        myFile3.close();

        cout << "\n\n";
        return 0;
}

void bubbleSort(vector<int> num, int size)
{
        int temp;
        for(int i = 1; i < size; i++){
                for (int j = 1; j < size; j++){
//                              cout << "j: " << j << "\ni: " << i << "\ndata[" << j << "]= " << num[j] << endl;
                        if(num[j-1] > num[j]){
                                temp = num[j-1];
                                num[j-1] = num[j];
                                num[j] = temp;
                        }
                }
//                cout << num[i] << endl;
        }
}

void quickSort(vector<int> num, int size)
{
        q_sort(num, 0, size-1);
}

void q_sort(vector<int> num, int l, int h)
{
//        for(int i = 0; i < num.size(); i++){
//                cout << num[i] << endl;
//        }
//        cout << endl;
        int pivot = num[l],
        i = l, j = h;
        do
        {
//                cout << "left: " << l;
//                cout << "\nn of left: " << num[l];
//                cout << "\n\nright: " << h;
//                cout << "\nn of right: " << num[h];
                while ((num[h] >= pivot) && (l < h))
                         h--;
                if (l != h)
                {
                        num[l] = num[h];
                        l++;
                }
//                cout << "\n\n---------------\nleft: " << left;
//                cout << "\nn of left: " << numbers[left];
//                cout << "\n\nright: " << right;
//                cout << "\nn of right: " << numbers[right];
                while ((num[l] <= pivot) && (l < h))
                        l++;
//                cout << "\n\n---------------\nleft: " << left;
//                cout << "\nn of left: " << numbers[left];
//                cout << "\n\nright: " << right;
//                cout << "\nn of right: " << numbers[right];
                if (l != h)
                {
                        num[h] = num[l];
                        h--;
                }
//                cout << "\n\n---------------\nleft: " << left;
//                cout << "\nn of left: " << numbers[left];
//                cout << "\n\nright: " << right;
//                cout << "\nn of right: " << numbers[right] << endl;
        }while (l < h);
        num[l] = pivot;
        pivot = l;
        l = i;
        h = j;
        if (l < pivot)
                 q_sort(num, l, pivot-1);
        if (h > pivot)
                 q_sort(num, pivot+1, h);
}
```