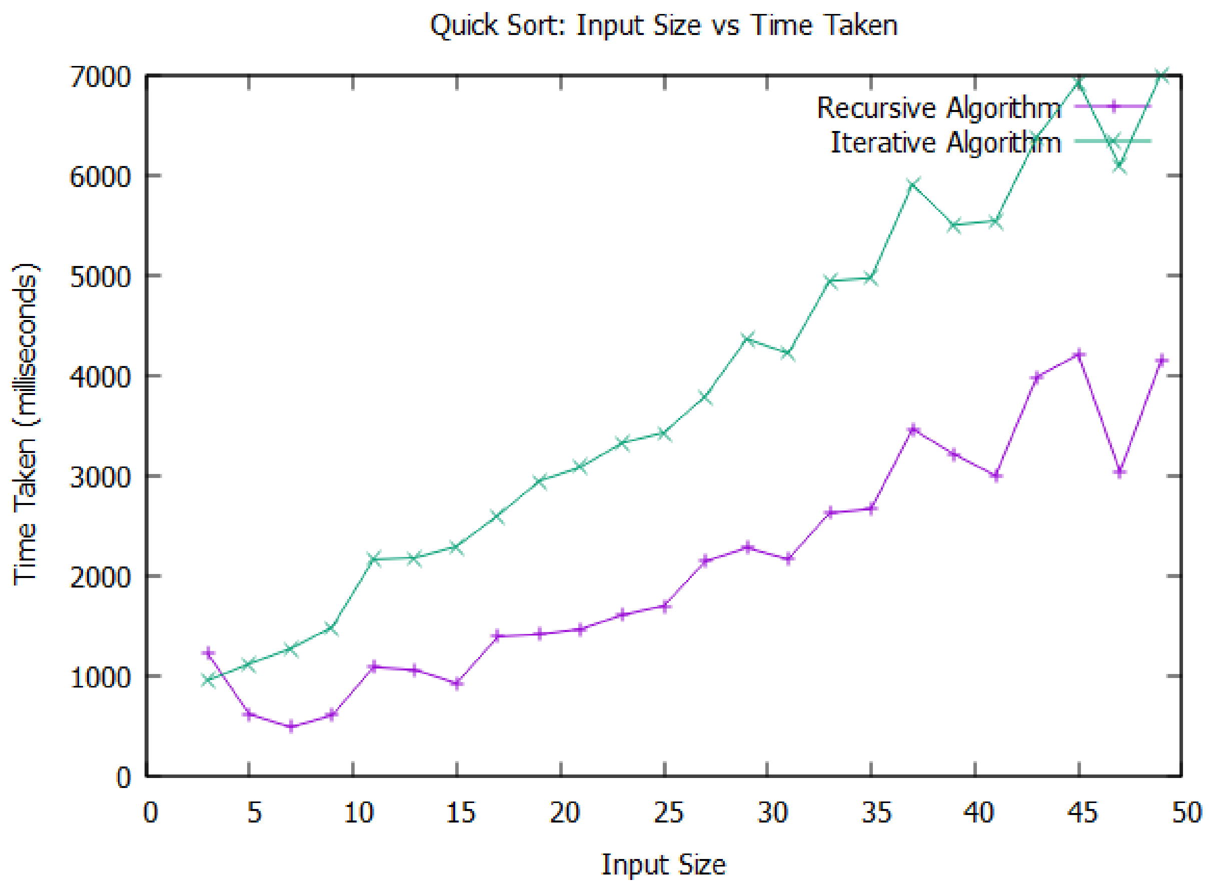
Lab 4 (Quick Sort)

Name : MKSL Weerasiri

Registration No. : 220689N

Graph



By GNUPlot and VSCode

Code (For plotting graph)

```
#include <iostream>
using namespace std;

int main() {

    FILE *gnuplotPipe = popen("gnuplot -persistent", "w");
    fprintf(gnuplotPipe, "set title 'Quick Sort: Input Size vs Time 
Taken'\n");
    fprintf(gnuplotPipe, "set xlabel 'Input Size'\n");
    fprintf(gnuplotPipe, "set ylabel 'Time Taken (milliseconds)'\n");
    fprintf(gnuplotPipe, "plot 'data_R.txt' with linespoints title 'Recursive Algorithm', 'data_I.txt' with linespoints title 'Iterative Algorithm'\n");
    fflush(gnuplotPipe);

    return 0;
}
```

Data

```
QuickSort Recursive (Data Inside data_R.txt)
#########
3   1218.40000000000009094947
5   613.20000000000004547474
7   489.00000000000000000000
9   603.20000000000004547474
11  1086.00000000000000000000
13  1056.20000000000004547474
15  923.79999999999995452526
17  1390.59999999999990905053
19  1412.59999999999990905053
21  1466.79999999999995452526
23  1605.00000000000000000000
25  1697.40000000000009094947
27  2141.80000000000018189894
29  2274.19999999999981810106
31  2164.00000000000000000000
33  2622.80000000000018189894
35  2662.80000000000018189894
37  3460.40000000000009094947
39  3208.00000000000000000000
41  2997.40000000000009094947
43  3983.59999999999990905053
45  4201.80000000000018189894
47  3041.59999999999990905053
49  4151.80000000000018189894


QuickSort Iterative (Data Inside data_I.txt)
#########
3   953.60000000000002273737
5   1118.00000000000000000000
7   1270.20000000000004547474
9   1484.79999999999995452526
11  2160.19999999999981810106
13  2178.00000000000000000000
15  2290.19999999999981810106
17  2599.00000000000000000000
19  2943.40000000000009094947
```

```
21  3082.00000000000000000000
23  3322.19999999999981810106
25  3426.40000000000009094947
27  3789.19999999999981810106
29  4360.19999999999981810106
31  4216.00000000000000000000
33  4939.00000000000000000000
35  4969.39999999999963620212
37  5903.00000000000000000000
39  5496.39999999999963620212
41  5540.00000000000000000000
43  6372.00000000000000000000
45  6921.00000000000000000000
47  6087.39999999999963620212
49  6991.19999999999981810106
```

By https://www.tutorialspoint.com/compile_cpp_online.php

Code (For data)

```cpp
#include <iostream>
#include <vector>
#include <chrono>
using namespace std;
void print(int n,vector<int> arr)
{
    for(int i=0; i<n; i++) {
        std::cout<<arr[i]<<" ";
    }
    std::cout<<"\n";
}
vector<vector<int>> makeRandomArrays(int start_size,int end_size,int step, int
value_limit)
{
    vector<vector<int>> arrays;
    vector<int> sample;
    for(int i=start_size; i<end_size+1; i=i+step) {
        sample.clear();
        for(int j=0; j<i; j++) {
            sample.push_back(rand()%(value_limit+1));
        }
        arrays.push_back(sample);
    }
    return arrays;
}
```

```
void swap(int &a,int &b)
{
    int temp=a;
    a=b;
    b=temp;
}
int arrange(vector<int> &A,int p,int r){
    int pivoted=A[r];
    int i=p-1;
    for(int j=p;j<r;j++){
        if(A[j]<pivoted){
            i++;
            swap(A[j],A[i]);
        }
    }
    i++;
    swap(A[r],A[i]);
    return i;
}
void QuickSortR(vector<int> &A,int p,int r){
    if(p>=r){
        return;
    }
    int pivoted_index=arrange(A,p,r);
    QuickSortR(A,p,pivoted_index-1);
    QuickSortR(A,pivoted_index+1,r);
}
void push(vector<int>&A,int n){
    A.push_back(n);
}
int pop(vector<int>&A){
   int n=A[(int)A.size()-1];
   A.pop_back();
   return n;
}
void QuickSortI(vector<int>&A,int n){
    vector<int> stk;
    push(stk,0);
    push(stk,n-1);
    int last,first,pivot;
    while((int)stk.size()!=0){
        last=pop(stk);
        first=pop(stk);
        pivot=arrange(A,first,last);
        if (pivot - 1 > first){
          push(stk, first);
          push(stk,pivot-1);
        }
        if (pivot + 1 < last){
          push( stk,pivot + 1);
          push( stk,last);
        }
    }
}
void runtheProgramQuickSortRecursive(int n,vector<int> inputs){
    QuickSortR(inputs,0,n-1);
}
void runtheProgramQuickSortIterative(int n,vector<int> inputs){
    QuickSortI(inputs,n);
}


int main()
{
    //Get the values
    vector<vector<int>> arrays=makeRandomArrays(3,50,2,100);

    double sum_duration;
    vector<double> avg_duration;
    string topic;
    for(int sorting=0; sorting<2; sorting++) {
        avg_duration.clear();
        for(int t=0; t<arrays.size(); t++) {
            sum_duration=0.0f;

            for(int i=0; i<5; i++) { //5 times

                auto start = chrono::high_resolution_clock::now();


                switch(sorting) {
                case 0:
                    runtheProgramQuickSortRecursive(arrays[t].size(),arrays[t]
);

                    topic="\n\n\nQuickSort Recursive\n#########\n";
                    break;
                case 1:
                    runtheProgramQuickSortIterative(arrays[t].size(),arrays[t]
);

                    topic="\n\n\nQuickSort Iterative\n#########\n";
                    break;
                default:
                    break;
                }
                auto end = chrono::high_resolution_clock::now();

                // Calculating total time taken by the program.
                double time_taken =
                    chrono::duration_cast<chrono::nanoseconds>(end -
start).count();



                sum_duration=sum_duration+time_taken;



            }
            avg_duration.push_back(sum_duration/5.0f);
        }
        cout<<topic;
        for(int i=0; i<avg_duration.size(); i++) {
            printf("%i\t%.20f\n",i*2+3,avg_duration[i]);
        }
    }


    return 0;
}
```