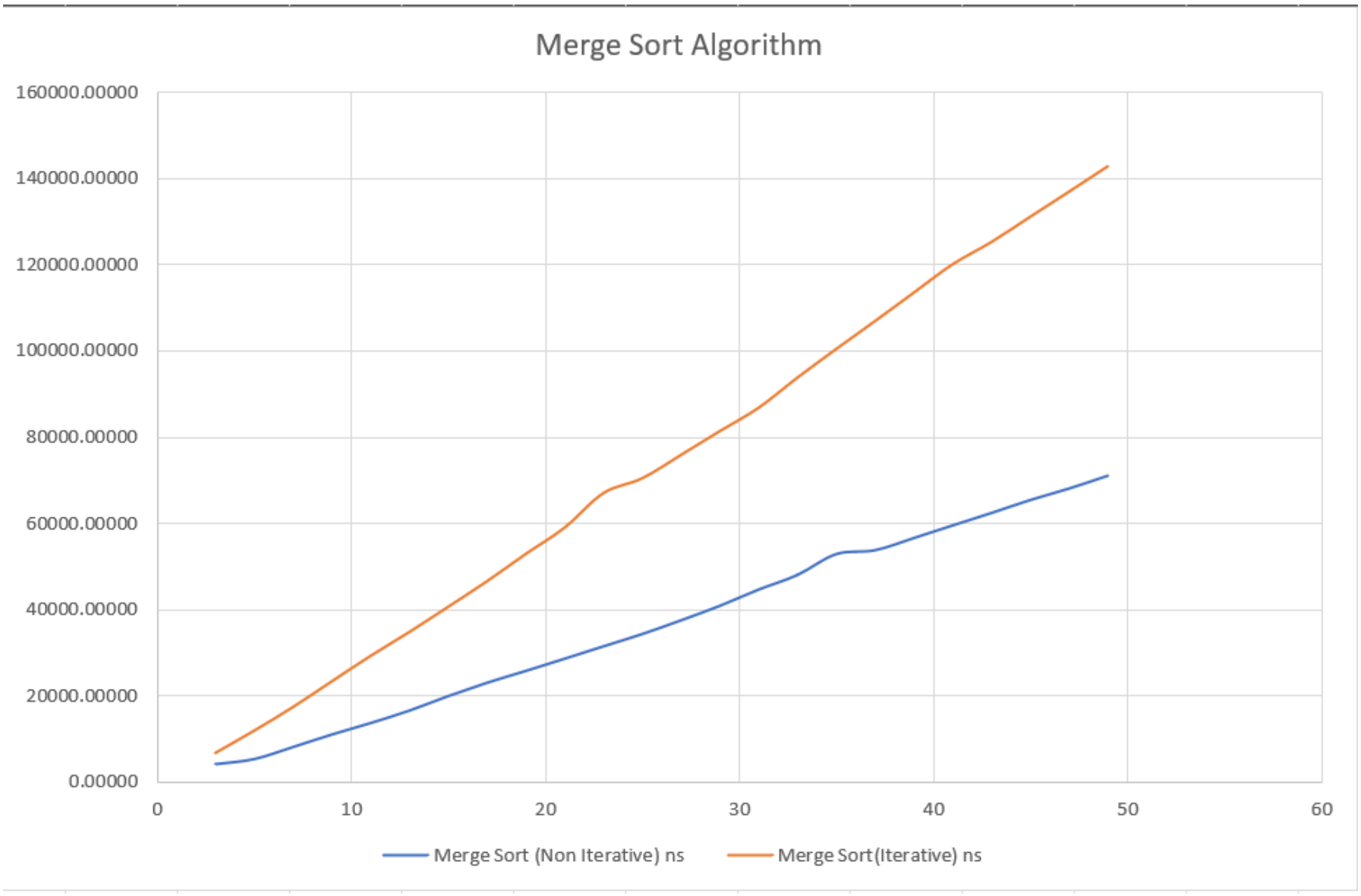
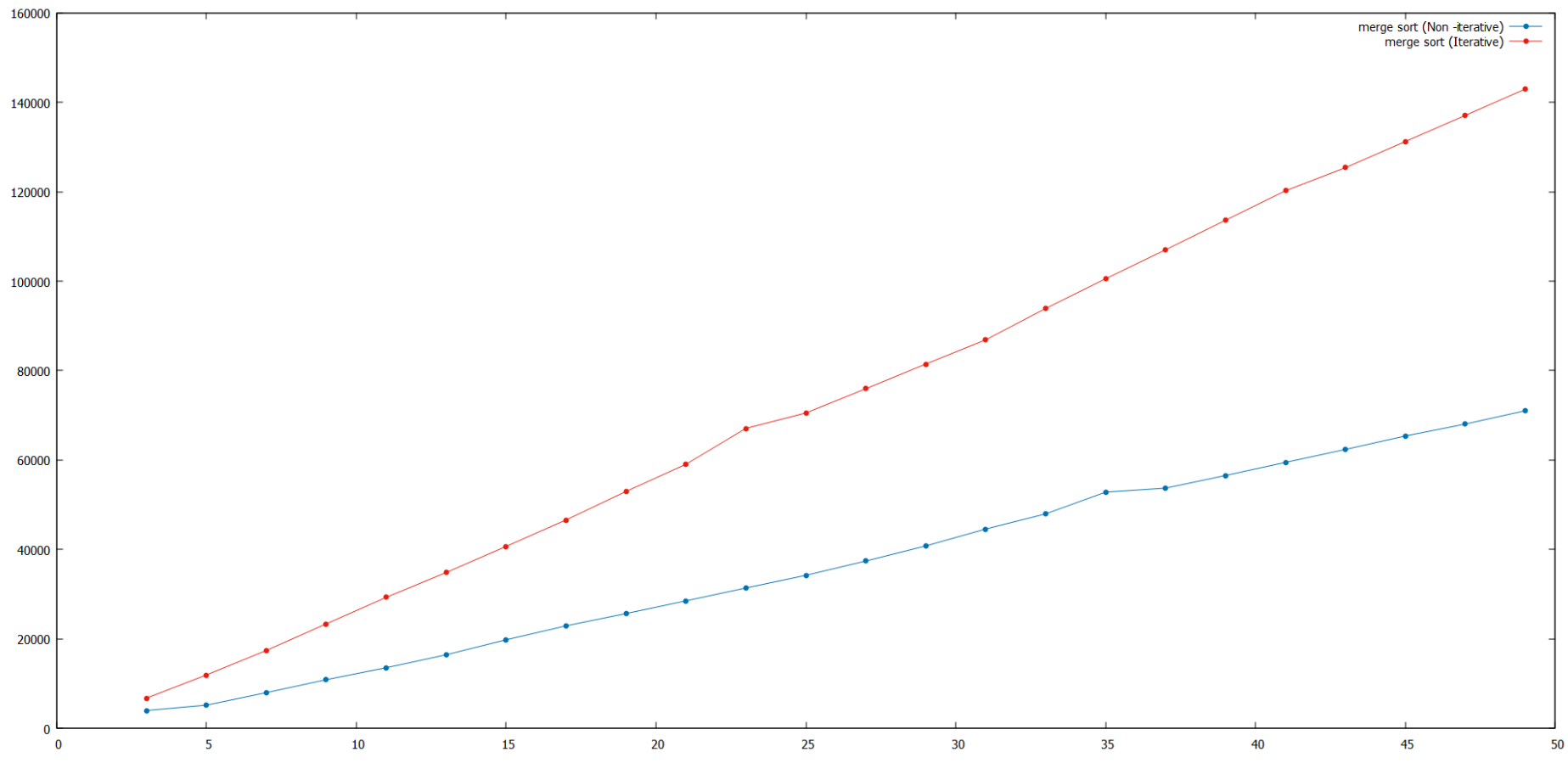
Lab3 (Merge Sort)

WEERASIRI MKSL – 220689N

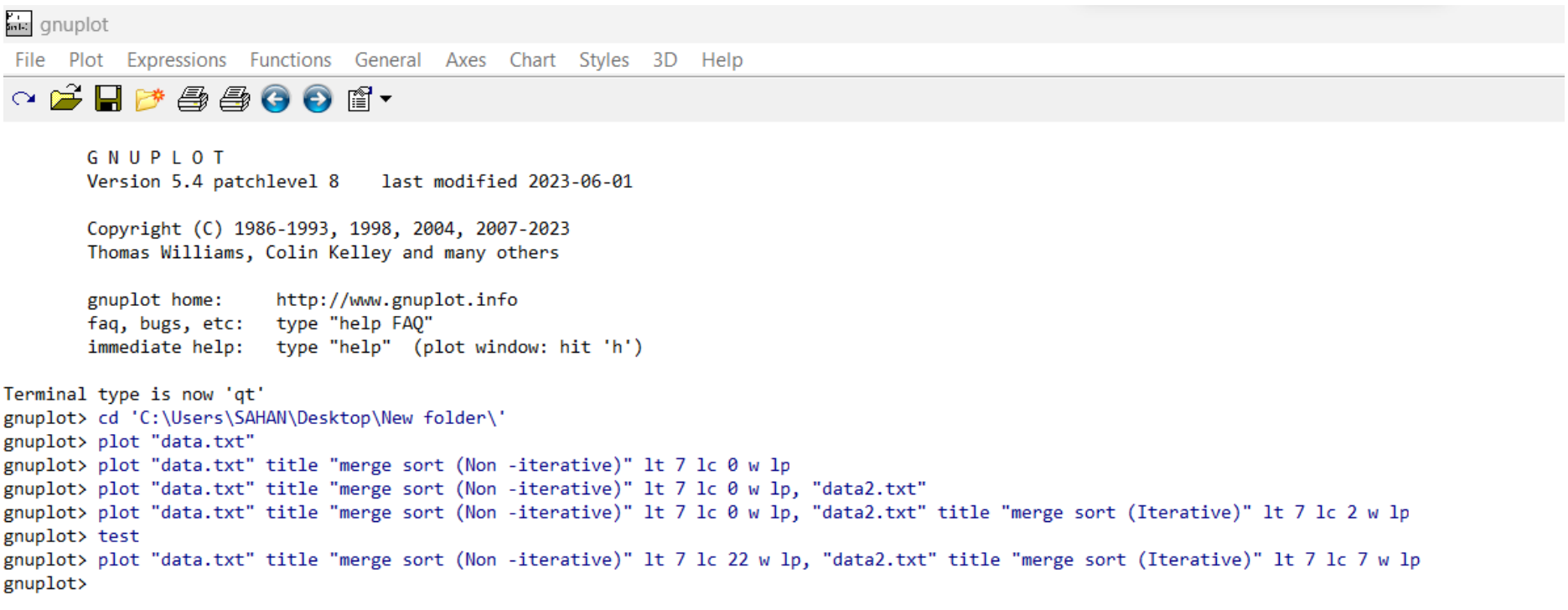
Graph by MS EXCEL



Graph by GNUPLOT



GNUPLOT CODE:



Data by the below code

n	Merge Sort (Non Iterative) ns	Merge Sort(Iterative) ns
3	4045.60000	6790.40000
5	5217.80000	11958.40000
7	8021.20000	17438.80000
9	10920.60000	23396.00000
11	13589.40000	29315.20000
13	16483.20000	34859.40000
15	19841.00000	40710.40000
17	22945.00000	46625.40000
19	25696.00000	52979.40000
21	28531.60000	59072.60000
23	31404.80000	67089.60000
25	34260.00000	70526.40000
27	37448.20000	75954.60000
29	40810.60000	81473.00000
31	44591.60000	86889.00000
33	47985.80000	93888.00000
35	52831.20000	100530.60000
37	53730.60000	107012.60000
39	56560.20000	113613.20000
41	59485.80000	120193.40000
43	62381.00000	125361.20000
45	65388.60000	131222.20000
47	68055.60000	137045.20000
49	71035.20000	142942.00000

Code: (By https://www.tutorialspoint.com/compile_cpp_online.php)

```
#include <iostream>
#include <vector>
#include <chrono>
#include <limits.h>
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;
}

vector<int> merge(vector<int> A, int l1,int r1,int l2,int r2){
   vector<int> temp;
   while(l1<=r1 && l2<=r2){
      if(A[l1]<A[l2]){
         temp.push_back(A[l1]);
         l1++;
      }else{
         temp.push_back(A[l2]);
         l2++;
      }
   }
```

```
    while(l1<=r1){
        temp.push_back(A[l1]);
        l1++;
    }

    while(l2<=r2){
        temp.push_back(A[l2]);
        l2++;
    }

    return temp;
}

vector<int> mergeSort(vector<int> arr, int left, int right){
    if(left<right){
        int mid=(left+right)/2;
        vector<int > L,R;
        L=mergeSort(arr,left,mid);
        R=mergeSort(arr,mid+1,right);
        vector<int> LR=L;
        LR.insert(LR.end(),R.begin(),R.end());
        vector<int> res= merge(LR,0,mid-left,mid+1-left,right-left);
        return res;
    }
    vector<int> temp={arr[left]};
    return temp;
}
void mergeLEC(vector<int> &A, int l,int m,int r){
    vector<int> L,R;
    for(int i=l;i<m+1;i++){
        L.push_back(A[i]);
    }
    for(int i=m+1;i<r+1;i++){
        R.push_back(A[i]);
    }
    L.push_back(INT_MAX);
    R.push_back(INT_MAX);
    int i=0,j=0;
    for(int k=l;k<=r;k++){
        if(L[i]<R[j]){
            A[k]=L[i];
```

```
            i++;
        }else{
            A[k]=R[j];
            j++;
        }
    }
}

void mergeSortLEC(vector<int> &arr, int left, int right){
    if(left<right){
        int mid=(left+right)/2;
        mergeSortLEC(arr,left,mid);
        mergeSortLEC(arr,mid+1,right);
        mergeLEC(arr,left,mid,right);
    }
}

void runtheProgramMerge(int n,vector<int> A){
    mergeSort(A,0,n-1);
}
void runtheProgramMergeNON(int n,vector<int> A){
    mergeSortLEC(A,0,n-1);
}
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:
                runtheProgramMergeNON(arrays[t].size(),arrays[t]);
                topic="\n\n\nMerge non Iterative\n#########\n";
                break;
                case 1:
```

```
    runtheProgramMerge(arrays[t].size(),arrays[t]);
    topic="\n\n\nMErge 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("%.20f\n",avg_duration[i]);
      }
    }


    return 0;
}
```