CS2023 - DATA STRUCTURES AND ALGORITHMS IN-CLASS LAB EXERCISE
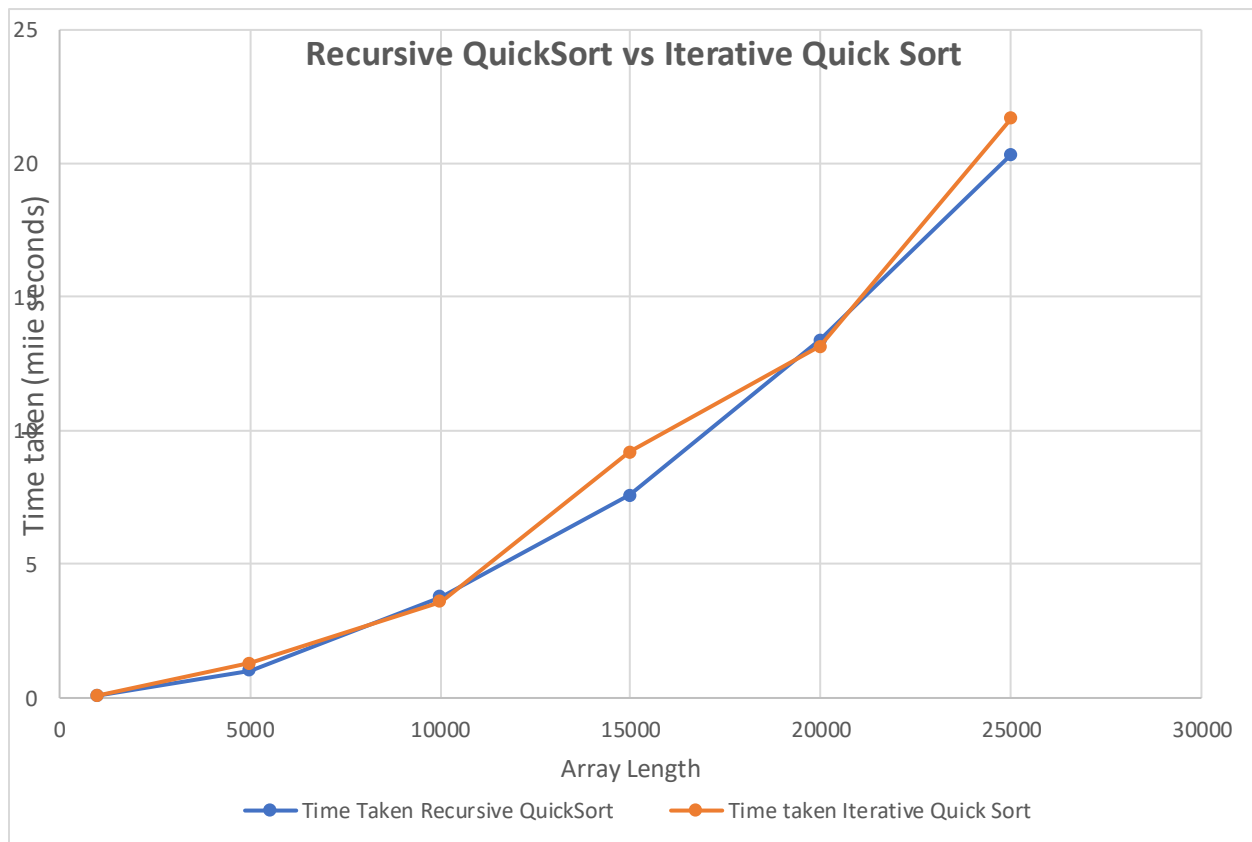
NAME:MUTHUWANA M.A.N.R

INDEX:210400N

Question 1

• Compare time taken for execution and plot the graph.

Array length	Time Taken Recursive QuickSort	Time taken Iterative Quick Sort
1000	0.089	0.0911
5000	1.0206	1.303
10000	3.7908	3.6191
15000	7.5758	9.2046
20000	13.3998	13.1619
25000	20.3332	21.6794

Screenshot 1 (top):

Code editor (main.cpp):

```cpp
            stack[++top] = p + 1;
            stack[++top] = h;
        }
    }
}

int main() {
    ///////////recursive quicksort//////////////
    //int a[] = {2, 6, 5, 1, 3, 4};
    vector <int> a;
    int length =1000;
    cout<<"Length of array :"<<length<<endl;
    for(int j=0;j<length;j++){
        a.push_back(rand()%100);
    }
//    for (int i = 0; i < length; i++)
//        cout << a[i] << " ";
//    cout<<endl;
```

Run output:
```
Length of array :1000
Recursive quicksort
Time taken is: 0.089000 milliseconds.

Length of array :1000
Non Recursive quick sort
Time taken is: 0.091100 milliseconds.

Process finished with exit code 0
```

Screenshot 2 (bottom):

Code editor (main.cpp):

```cpp
            stack[++top] = h;
        }
    }
}

int main() {
    ///////////recursive quicksort//////////////
    //int a[] = {2, 6, 5, 1, 3, 4};
    vector <int> a;
    int length =5000;
    cout<<"Length of array :"<<length<<endl;
    for(int j=0;j<length;j++){
        a.push_back(rand()%100);
    }
//    for (int i = 0; i < length; i++)
//        cout << a[i] << " ";
//    cout<<endl;
    cout<<"Recursive quicksort"<<endl;
```

Run output:
```
Length of array :5000
Recursive quicksort
Time taken is: 1.020600 milliseconds.

Length of array :5000
Non Recursive quick sort
Time taken is: 1.303000 milliseconds.

Process finished with exit code 0
```

First screenshot (top):

Code editor (main.cpp):
```
71              stack[++top] = h;
72          }
73      }
74  }
75
76  int main() {
77      ////////////recursive quicksort////////////
78      //int a[] = {2, 6, 5, 1, 3, 4};
79      vector <int> a;
80      int length =10000;
81      cout<<"Length of array :"<<length<<endl;
82      for(int j=0;j<length;j++){
83          a.push_back(rand()%100);
84      }
85  //   for (int i = 0; i < length; i++)
86  //       cout << a[i] << " ";
87  //   cout<<endl;
88      cout<<"Recursive quicksort"<<endl;
```

Run console output:
```
Length of array :10000
Recursive quicksort
Time taken is: 3.790800 milliseconds.

Length of array :10000
Non Recursive quick sort
Time taken is: 3.619100 milliseconds.

Process finished with exit code 0
```

Second screenshot (bottom):

Code editor (main.cpp):
```
71              stack[++top] = h;
72          }
73      }
74  }
75
76  int main() {
77      ////////////recursive quicksort////////////
78      //int a[] = {2, 6, 5, 1, 3, 4};
79      vector <int> a;
80      int length =15000;
81      cout<<"Length of array :"<<length<<endl;
82      for(int j=0;j<length;j++){
83          a.push_back(rand()%100);
84      }
85  //   for (int i = 0; i < length; i++)
86  //       cout << a[i] << " ";
87  //   cout<<endl;
88      cout<<"Recursive quicksort"<<endl;
```

Run console output:
```
Length of array :15000
Recursive quicksort
Time taken is: 7.575800 milliseconds.

Length of array :15000
Non Recursive quick sort
Time taken is: 9.204600 milliseconds.

Process finished with exit code 0
```
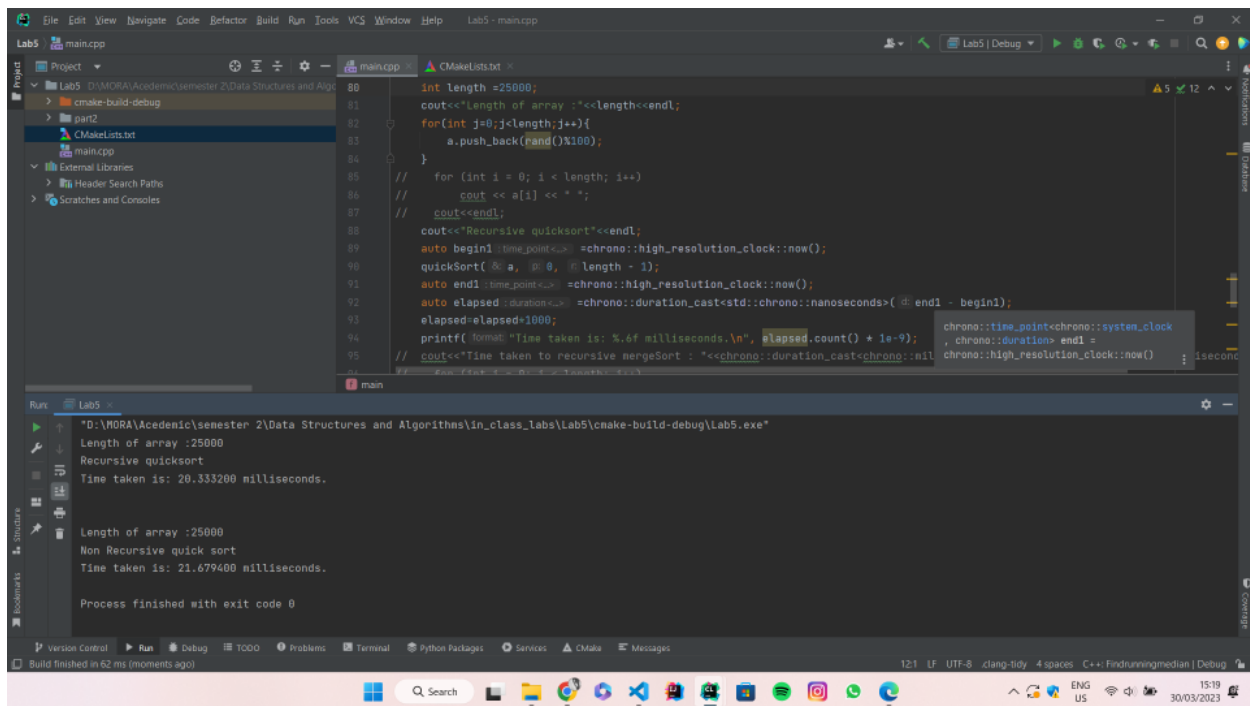
Screenshot 1 - Code:
```cpp
            stack[++top] = h;
        }
    }
}

int main() {
    /////////////recursive quicksort//////////////
    //int a[] = {2, 6, 5, 1, 3, 4};
    vector <int> a;
    int length =20000;
    cout<<"Length of array :"<<length<<endl;
    for(int j=0;j<length;j++){
        a.push_back(rand()%100);
    }
//    for (int i = 0; i < length; i++)
//        cout << a[i] << " ";
//    cout<<endl;
    cout<<"Recursive quicksort"<<endl;
```

Run output:
```
Length of array :20000
Recursive quicksort
Time taken is: 13.399800 milliseconds.


Length of array :20000
Non Recursive quick sort
Time taken is: 13.161900 milliseconds.


Process finished with exit code 0
```
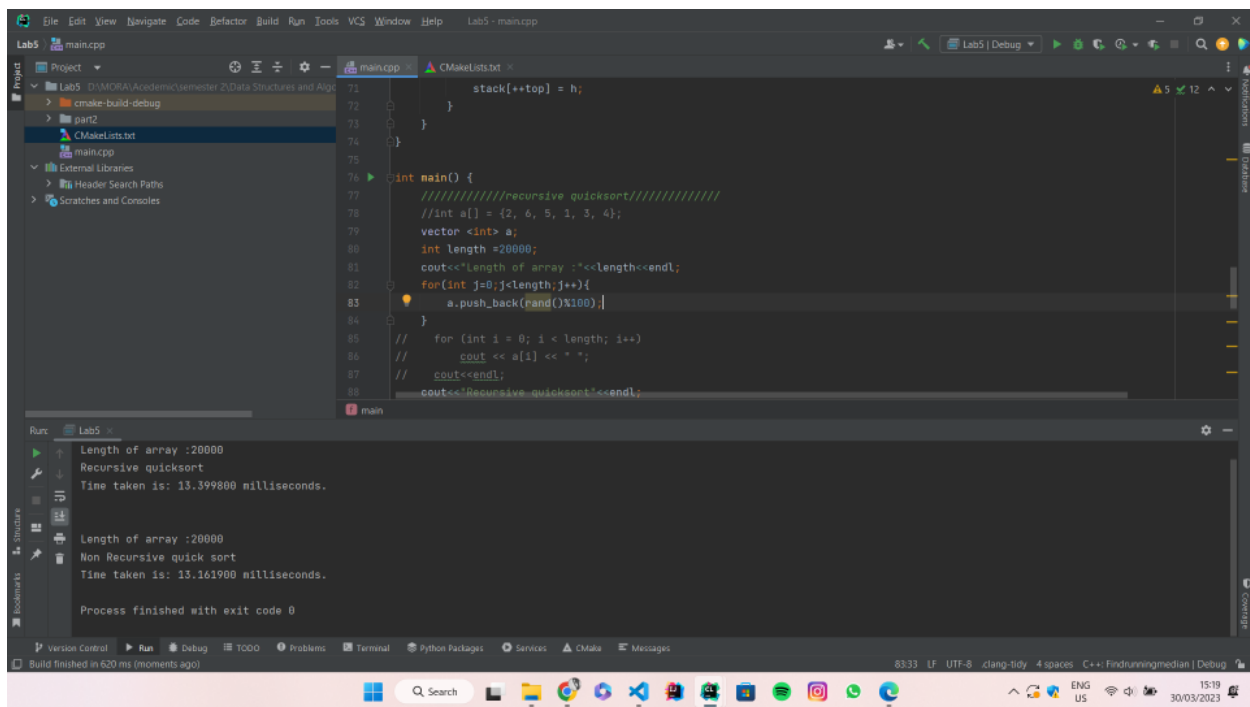

Screenshot 2 - Code:
```cpp
    int length =25000;
    cout<<"Length of array :"<<length<<endl;
    for(int j=0;j<length;j++){
        a.push_back(rand()%100);
    }
//    for (int i = 0; i < length; i++)
//        cout << a[i] << " ";
//    cout<<endl;
    cout<<"Recursive quicksort"<<endl;
    auto begin1 : time_point<...> =chrono::high_resolution_clock::now();
    quickSort( & a,  & 0,  & length - 1);
    auto end1 : time_point<...> =chrono::high_resolution_clock::now();
    auto elapsed : duration<...> =chrono::duration_cast<std::chrono::nanoseconds>( & end1 - begin1);
    elapsed=elapsed+1000;
    printf( format: "Time taken is: %.6f milliseconds.\n", elapsed.count() * 1e-9);
//    cout<<"Time taken to recursive mergeSort : "<<chrono::duration_cast<chrono::mil
```

Tooltip:
```
chrono::time_point<chrono::system_clock
, chrono::duration> end1 =
chrono::high_resolution_clock::now()
```

Run output:
```
"D:\MORA\Acedemic\semester 2\Data Structures and Algorithms\in_class_labs\Lab5\cmake-build-debug\Lab5.exe"
Length of array :25000
Recursive quicksort
Time taken is: 20.333200 milliseconds.


Length of array :25000
Non Recursive quick sort
Time taken is: 21.679400 milliseconds.


Process finished with exit code 0
```

Question 2: Find running median

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

int main() {
    cout<<"Quection 2"<<endl;
    vector<int> arr;
    int num;
    do{
        cout<<"Enter a number::";
        cin>>num;
        arr.push_back(num);
        for(int i=0;i<arr.size();i++){cout<<arr[i]<<" ";}cout<<endl;
        for(int i = 1; i < arr.size(); i++){
            int key=arr[i];
            int j=i-1;
            while(j>=0 && key<arr[j]){
                arr[j+1]=arr[j];
                j--;
            }
            arr[j+1]=key;
        }
        cout<<"Sorted number list by insertion sort:";
        for(int i=0;i<arr.size();i++){cout<<arr[i]<<" ";}cout<<endl;
        int k=arr.size()/2;
        if(arr.size()%2==0 && arr.size()>1){
            cout<< "The median of a set of integers:"<<(float)(arr[k]+arr[k-
1])/2<<endl;
        }else if(arr.size()%2==1 && arr.size()>1){
            cout<< "The median of a set of integers>>>"<<arr[k]<<endl;
        }else{
            cout<< "The median of a set of integers:"<<arr[0]<<endl;
        }
        cout<<endl;

    }while(num!=-1);
    return 0;
}
```