Experiment 2.1

Student Name: Utkarsh Joshi
UID: 21BCS9158
Branch: BE(CSE)
Section/Group: 802-A
Semester: 5th
Date of Performance:25/9/23
Subject Name: DAA
Subject Code: 21CSH-311

1. **Aim:**
   Sort a given set of elements using the Quick sort method and determine the time required to sort the elements. Repeat the experiment for different values of n, the number of elements in the list to be sorted. The elements can be read from a file or can be generated using the random number generator.

2. **Objective:**
   To employ the Quick sort algorithm for sorting a provided set of elements, measure the time it takes for sorting, and conduct the experiment across various 'n' values, denoting the number of elements in the list, with elements sourced from either a file or a random number generator.

3. **Algorithm:**

Quick Sort is a divide-and-conquer sorting algorithm that works as follows:

1. Choose a pivot element from the array.
2. Partition the array into two subarrays: elements less than the pivot and elements greater than the pivot.
3. Recursively apply Quick Sort to the subarrays.
4. Repeat this process until the entire array is sorted.

5. **Input/Apparatus Used:**
    a. Visual Studio
    b. C++/Java Programming Language
    c. C++/Java Compiler

6. Sample Code:

```
#include <iostream>
#include <vector>
#include <ctime>
#include <cstdlib>
#include <algorithm>

using namespace std;

// Function to perform the Quick Sort algorithm void
quickSort(vector<int>& arr, int low, int high) {
  if (low < high) { int
    pivot = arr[high]; int
    i = low - 1;

    for (int j = low; j < high; j++) {
      if (arr[j] < pivot) {
        i++; swap(arr[i],
        arr[j]);
      }
    }

    swap(arr[i + 1], arr[high]); int

    pi = i + 1; quickSort(arr, low,
```

```
        pi - 1); quickSort(arr, pi + 1,

        high);

    }
}

int main() {
    srand(static_cast<unsigned>(time(0)));
    vector<int> n_values = {10};

    for (int n : n_values) {
        vector<int> data(n); for
        (int i = 0; i < n; i++) {
            data[i] = rand() % 1000; // Generate random data
        }

        cout << "Original array for n=" << n << ":" << endl; for
        (int value : data) {
            cout << value << " ";
        }   cout   <<   endl;

        quickSort(data, 0, n - 1);

        cout << "Sorted array for n=" << n << ":" << endl; for
        (int value : data) {
            cout << value << " ";

        }
        cout << endl << endl;
    }
```

```
    return 0;
}
```

//Utkarsh 21BCS9158

7. Outcome:

```
/tmp/AmcefmLZhQ.o
Original array for n=10:
376 616 172 727 552 329 739 574 282 949
Sorted array for n=10:
172 282 329 376 552 574 616 727 739 949
```

6. Time Complexity:

1) Best Case: O(N log(N))
2) Average Case: O(N log(N))
3) Worst Case: O(N^2)