NAME:	Prithvi Singh
UID:	2022301014
SUBJECT	DAA
EXPERIMENT NO:	1B
DATE OF PERFORMANCE	9/2/23
DATE OF SUBMISSION	10/2/23
AIM:	To implement the various functions e.g. linear, non-linear, quadratic, exponential etc.
ALGORITHM	• **Insertion sort-** a. procedure insertionSort(A: list of sortable items) b. n = length(A) c. for i = 1 to n - 1 do d. j = i e. while j > 0 and A[j-1] > A[j] do f. swap(A[j], A[j-1]) g. j = j - 1 h. end while i. end for j. end procedure • **Selection sort-** k. Repeat Steps b and c for i = 0 to n-1 l. CALL SMALLEST(arr, i, n, pos) m. [INITIALIZE] SET key = i n. Repeat for j = i+1 to n o. if (arr[key] > arr[j]) p. SET key=j q. [END OF if]

	r. [END OF LOOP] s. SWAP arr[i] with arr[pos] t. [END OF LOOP] u. EXIT
PROGRAM:	1.1b.cpp

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

int main()
{
    for(int i =0; i<100000; i++)
    {
        cout<<rand() << "\n";
    }
}
```

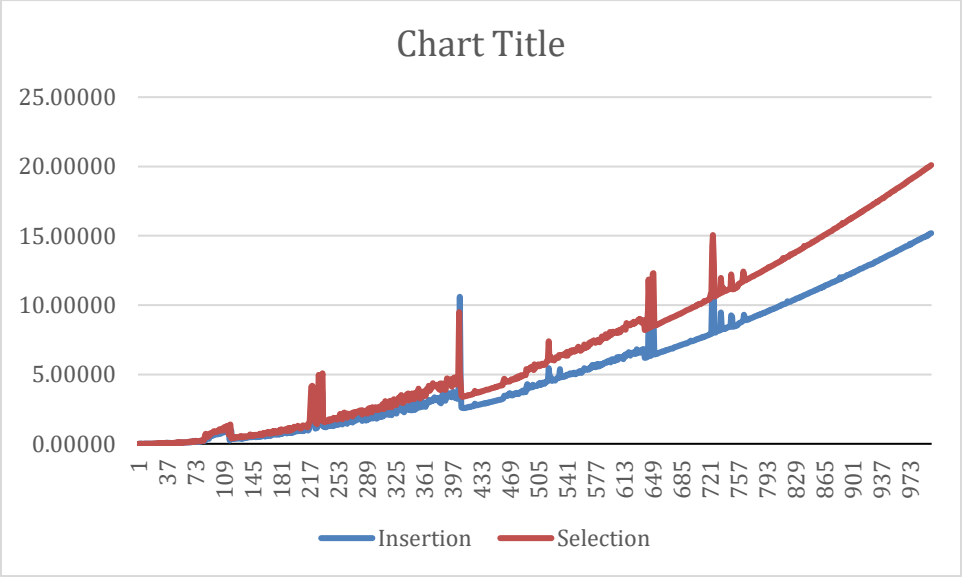
2. Main1.cpp

```cpp
#include <bits/stdc++.h>
#include <fstream>
using namespace std;

void insertionsort(vector<int> arr, int num)
{
    int key, j;
    for (int i = 1; i < num; i++)
    {
        key = arr[i];
        j = i - 1;
        while (j >= 0 && arr[j] > key)
        {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
}

void selectionsort(vector<int> arr, int num)
```

```
{
    int key;
    for (int i = 0; i < num - 1; i++)
    {
        key = i;
        for (int j = i + 1; j < num; j++)
        {
            if (arr[j] < arr[key])
            {
                key = j;
            }
        }
        int temp;
        temp = arr[key];
        arr[key] = arr[i];
        arr[i] = temp;
    }
}

int main()
{
    vector<int> arr;
    clock_t t1, t2, t3, t4;
    string filename("values.txt");
    ifstream fin(filename);
    if (!fin.is_open())
    {
        cerr << "Could not open the file - '"
             << filename << "'" << endl;
        return EXIT_FAILURE;
    }

    while (!fin.eof())
    {
        int tmp;
        fin >> tmp;
        arr.push_back(tmp);
    }

    int num = 100;
    for (int i = 0; i <1000; i++)
    {
        t1 = clock();
        insertionsort(arr, num);
```

```
        t2 = clock();
        t3 = clock();
        selectionsort(arr, num);
        t4 = clock();
        double insertiontime = double(t2 - t1) /
double(CLOCKS_PER_SEC);
        double selectiontime = double(t4 - t3) /
double(CLOCKS_PER_SEC);
        cout << endl;
        cout << i+1 <<" "<<fixed << insertiontime << setprecision(5)
<< '\t';
        cout << fixed << selectiontime << setprecision(5);

        num += 100;
    }
    fin.close();
    return 0;
}
```

Graphs and Observation:



1. Insertion Sort:

Here as we can see in the graph for insertion sort that as the values change time required for sorting changes too. This can be understood by seeing throttling at different stages in the execution. Sometimes

	also due to different values in each block the time to sort those blocks also differs. We can also accumulate from the graph that insertion sort for the most amount of running time takes less time than compared to selection sort. Despite of throttling we can observe that graph moves almost evenly for most amount of running time. Also, the running time differs from system to system as each system has different characteristics and specifications. 2. **Selection Sort:** Here as we can see in the graph for insertion sort that as the values change time required for sorting changes too. This can be understood by seeing throttling at different stages in the execution. Sometimes also due to different values in each block the time to sort those blocks also differs. We can also acknowledge from the graph that selection sort for the most amount of running time takes more time than compared to insertion sort. Despite of throttling we can observe that graph moves almost evenly for most amount of running time and also almost linear to insertion sort at the end stages of the execution. Also, the running time differs from system to system as each system has different characteristics and specifications.
Conclusion:	The experiment on finding the running time of insertion sort and selection sort shows that the running time of both algorithms is dependent on the size of the input data. The running time of insertion sort is quadratic in nature, while that of selection sort is also quadratic but performs better on small inputs.