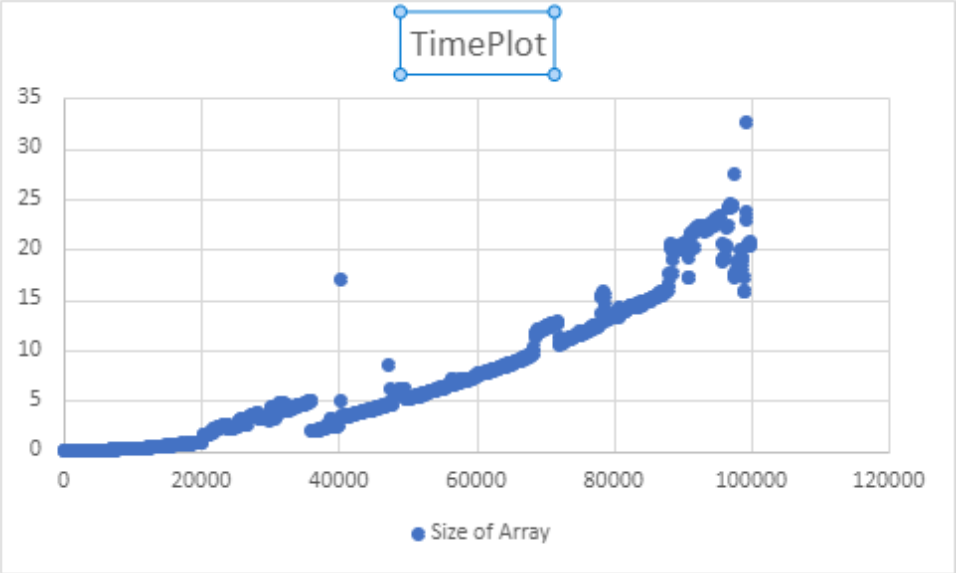
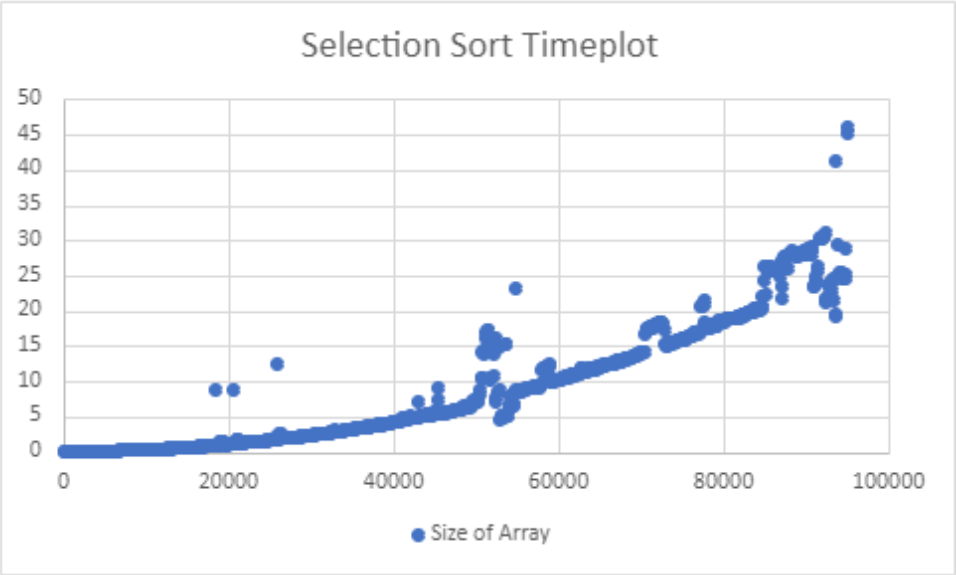
NAME:	Harshith Kunder
UID:	2021300068
SUBJECT	Design and Analysis of Algorithms.
EXPERIMENT NO :	1 B.
PROBLEM STATEMENT 1:	Experiment on finding the running time of Insertion and selection sort.

Program:

```
#include<bits/stdc++.h>

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 curri){
    if(curri==arr.size()) return;
    for(int i=curri;i<arr.size();i++){
        if(arr[i]<arr[curri]) swap(arr[i],arr[curri]);
    }
```

```
    selectionSort(arr,curri+1);
}
int main()
{
    vector<int> arr;
    clock_t start, end;
    vector<int> numbers;

    for(int i=36000;i<=100000;i+=100){
        numbers.push_back(i);
    }

    //cout<<numbers.size()<<" ";

    // for(int i=0;i<numbers.size();i++){
    //     cout<<numbers[i]<<" ";
    // }
    for(int i=0;i<numbers.size();i++){
        arr.clear();
        std::fstream myfile("numbers.txt", std::ios_base::in);
        int a;
        for(int j=0 ; j<numbers[i] ; j++){
            myfile>>a;
            arr.push_back(a);
        }
        /* Recording the time.*/
        start = clock();

        insertionsort(arr,arr.size());

        end = clock();

        // Calculating total time taken by the program.
        double time_taken = double(end - start) /
double(CLOCKS_PER_SEC);
        cout<<numbers[i]<<" "<< fixed
            << time_taken << setprecision(5);
        cout << "\n" ;
    }
```

	`` return 0; } ```
Algorithm:	**Insertion sort:** **To sort an array of size N in ascending order:** ● Iterate from arr[1] to arr[N] over the array. ● Compare the current element to its predecessor. ● Until an element lesser than or equal to key is found, keep shifting elements one space to the right. ● Replace the void formed after shifting with the key. **Selection sort:** **For array of size n:** ● Traverse the unsorted array and find the number with least value. ● Swap the number with least value with the element at the start. ● Perform the same procedure for the unsorted part of the array.

Data:	Graph For Insertion Sort: Y axis shows time in seconds  Data: Refer to the excel sheet attached along with the pdf. Graph For Selection sort: Y axis shows time in seconds  Data: Refer to the excel sheet attached along with the pdf.

Observation:	1. Fluctuations can be seen in the sorting algorithm performance due to Laptop overheating, Background apps and low battery. 2. In general we can see that the performance of both algorithms is almost identical up to array size of 20000. 3. However after that insertion sort was performing better as compared to selection sort. 4. This is because in insertion sort number of swap/shifting operations performed is lesser on average as compared to Selection Sort.
Conclusion:	I understood in depth the Insertion and Selection sorting algorithms and their relative performance.