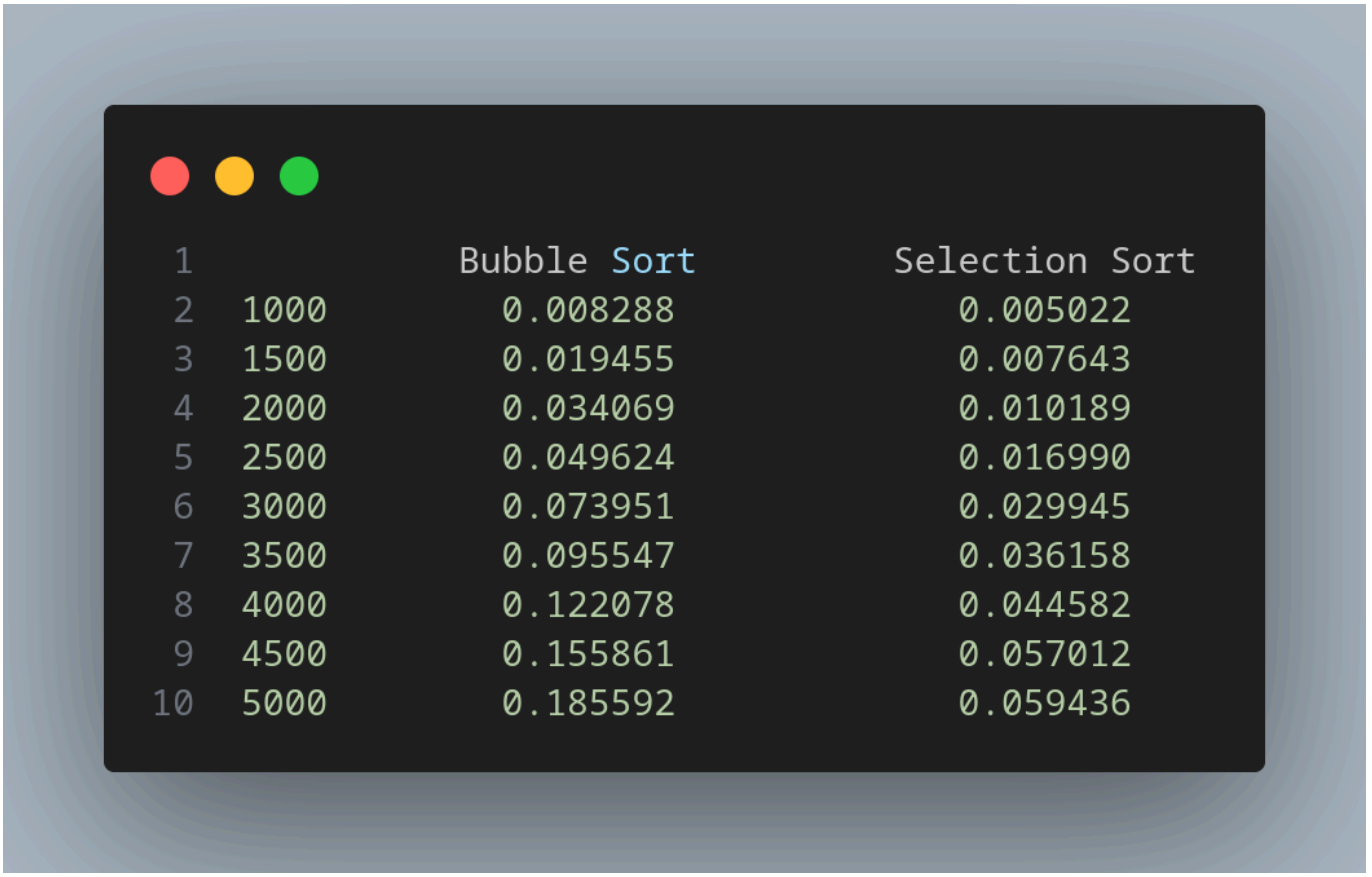
Practical no. 1

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

// Function to print array elements to a file
void printArr(vector <int> arr, int n, string filename){
    ofstream outputfile(filename);
    for(int i=0; i<n; i++){
        outputfile << arr[i]<< " ";
    }
    outputfile.close();
}

// Function to perform selection sort and print sorted array to a file
void selectionSort(vector <int> arr, int n){
    for(int i = 0; i<n; i++){
        int mini = i;
        for(int j= i+1; j<n; j++)
            if(arr[j]< arr[mini])
                mini = j;
        swap(arr[i], arr[mini]);
    }
    printArr(arr, n, "selectionSort.txt");
}

// Function to perform bubble sort and print sorted array to a file
void bubbleSort(vector <int> arr, int n){
    for(int i = 0; i<n-1; i++){
        for(int j = 0; j< n-1-i; j++)
            if(arr[j] > arr[j+1])
                swap(arr[j], arr[j+1]);
    }
    printArr(arr, n, "bubbleSort.txt");
}

// Function to generate random numbers and store them in a file
void randomNums(int n){
    vector <int> arr;
    ofstream outputfile("input.txt");
    for(int i=0; i<n; i++){
        outputfile << rand() % (n+1)<< " ";
    }
    outputfile.close();
}

// Function to run sorting algorithms and measure their execution time
void runner(int num){
    randomNums(num); // put number of numbers to generate
```

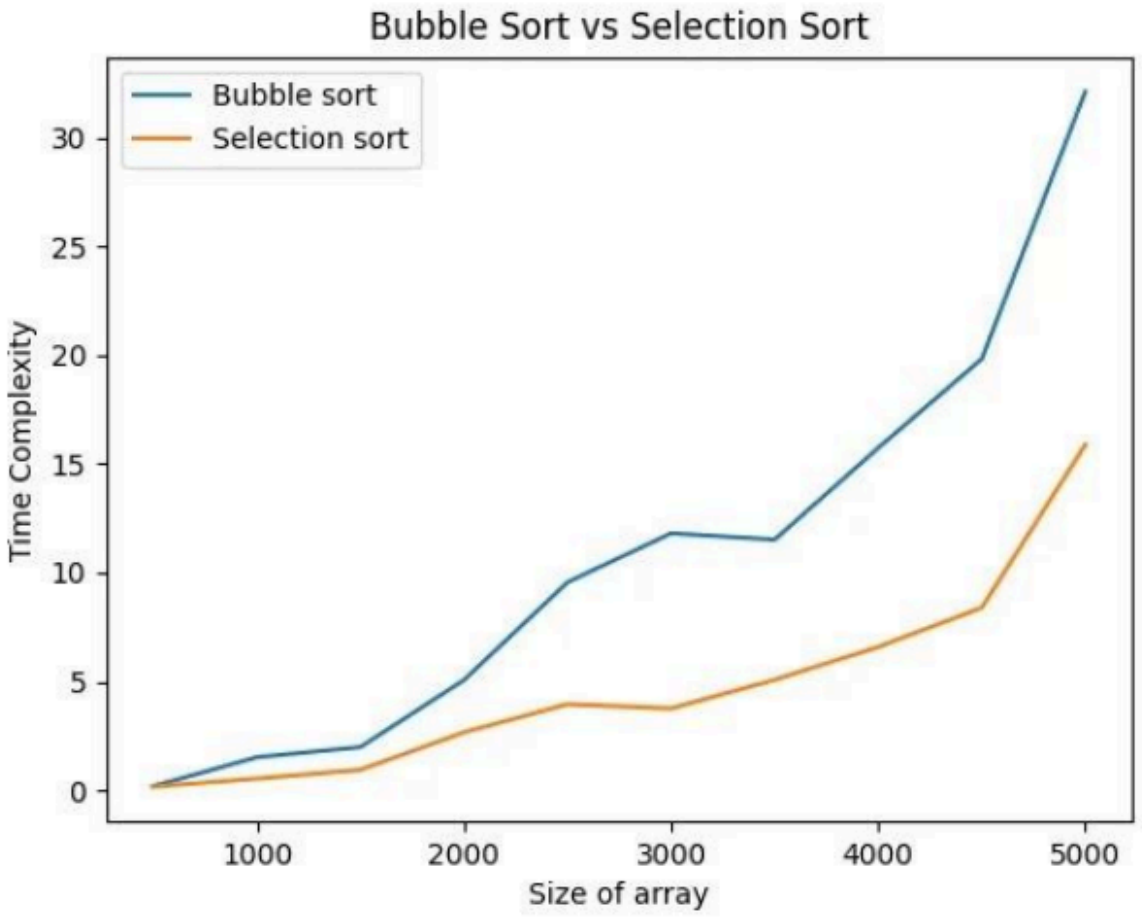
```
    clock_t startTime, endTime;
    double totalTime;

    vector <int> arr;
    ifstream readfile("input.txt");

    int value;
    while(readfile >> value){
        arr.push_back(value);
    }
    readfile.close();

    // Bubble Sort
    startTime = clock();
    bubbleSort(arr, arr.size());
    endTime = clock();

    cout<<num;
    totalTime = ((double)(endTime - startTime)) / CLOCKS_PER_SEC;
    cout<<"        "<<totalTime;

    // Selection Sort
    startTime = clock();
    selectionSort(arr, arr.size());
    endTime = clock();

    totalTime = ((double)(endTime - startTime)) / CLOCKS_PER_SEC;
    cout<<"             "<<fixed << setprecision(6) << totalTime<<endl;
}

// Main function
int main(){
    cout<<"         Bubble Sort         Selection Sort"<<endl;

    // Running sorting algorithms for different input sizes
    runner(1000);
    runner(1500);
    runner(2000);
    runner(2500);
    runner(3000);
    runner(3500);
    runner(4000);
    runner(4500);
    runner(5000);

    return 0;
}
```

Name- Sahil Tiwaskar
Roll no- 50B



	Bubble Sort	Selection Sort
1000	0.008288	0.005022
1500	0.019455	0.007643
2000	0.034069	0.010189
2500	0.049624	0.016990
3000	0.073951	0.029945
3500	0.095547	0.036158
4000	0.122078	0.044582
4500	0.155861	0.057012
5000	0.185592	0.059436

```
import matplotlib.pyplot as plt
import numpy as np
num_a = np.array([500,1000,1500,2000,2500,3000,3500,4000,4500,5000])
tc_a =
np.array([0.194258,1.52175,1.99094,5.07002,9.56315,11.8094,11.5187,15.73
88,19.8269,32.1124])
num_b = np.array([500,1000,1500,2000,2500,3000,3500,4000,4500,5000])
tc_b =
np.array([0.167507,0.541027,0.943503,2.66785,3.95831,3.76273,5.07976,6.5
8913,8.3958,15.8889])
plt.plot(num_a, tc_a, label='Bubble sort')
plt.plot(num_b, tc_b, label='Selection sort')
plt.xlabel('Size of array')
plt.ylabel('Time Complexity')
```



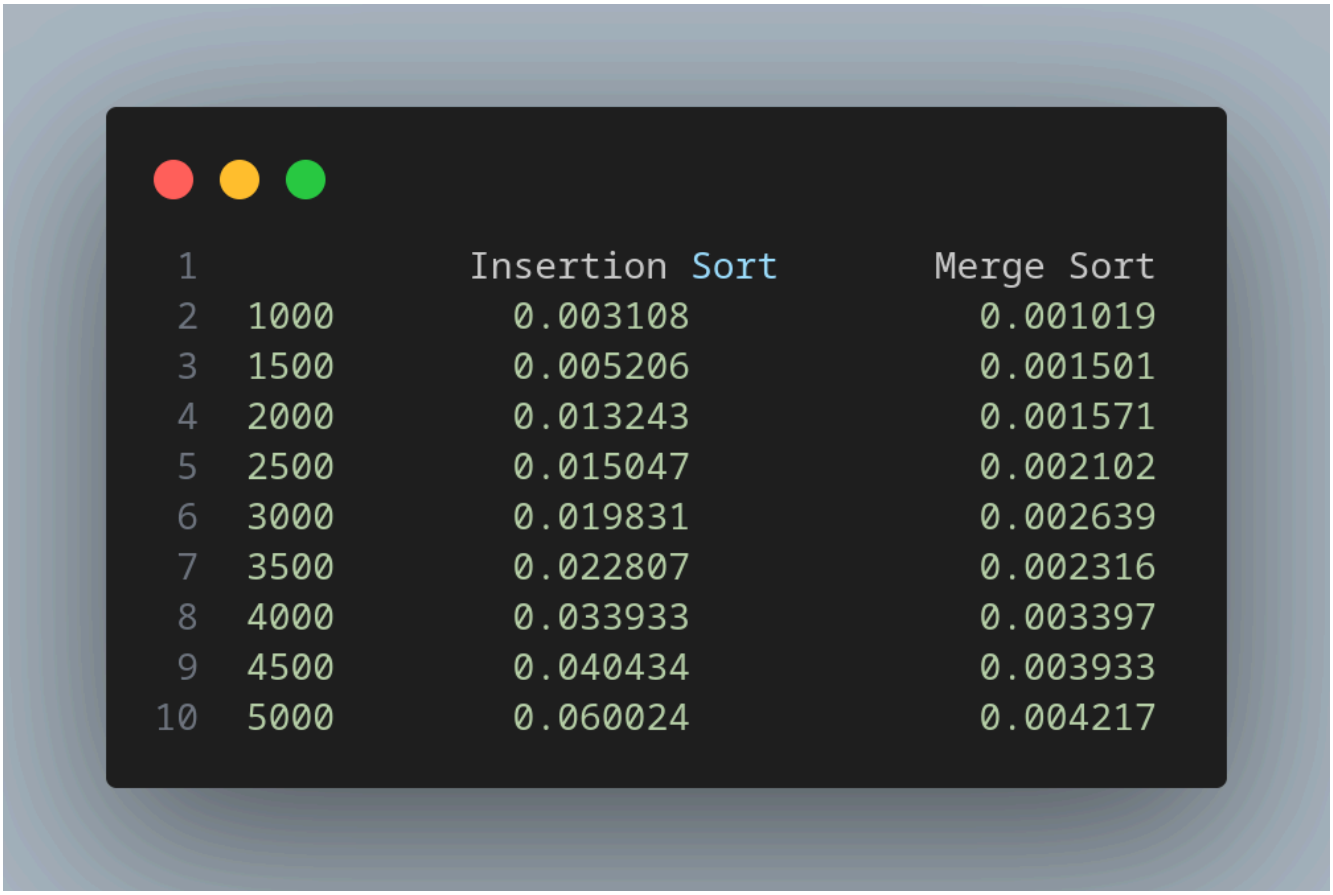
Practical no. 2

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

// Function to print array elements to a file
void printArr(vector <int> arr, int n, string filename){
    ofstream outputfile(filename);
    for(int i=0; i<n; i++){
        outputfile << arr[i]<< " ";
    }
    outputfile.close();
}

// Function to perform Insertion Sort and print sorted array to a file
void insertionSort(vector <int> arr, int n){
    for(int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i - 1;
        while(j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = key;
    }
    printArr(arr, n, "insertionSort.txt");
}

// Function to perform Merge Sort
void merge(vector<int>& arr, int l, int m, int r) {
    int n1 = m - l + 1;
    int n2 = r - m;

    vector<int> L(n1), R(n2);

    for (int i = 0; i < n1; i++)
        L[i] = arr[l + i];
    for (int j = 0; j < n2; j++)
        R[j] = arr[m + 1 + j];

    int i = 0, j = 0, k = l;
    while (i < n1 && j < n2) {
        if (L[i] <= R[j]) {
            arr[k] = L[i];
            i++;
        } else {
            arr[k] = R[j];
            j++;
        }
```

```
        k++;
    }

    while (i < n1) {
        arr[k] = L[i];
        i++;
        k++;
    }

    while (j < n2) {
        arr[k] = R[j];
        j++;
        k++;
    }
}

void mergeSort(vector<int>& arr, int l, int r) {
    if (l >= r) return;
    int m = l + (r - l) / 2;
    mergeSort(arr, l, m);
    mergeSort(arr, m + 1, r);
    merge(arr, l, m, r);
}

// Function to generate random numbers and store them in a file
void randomNums(int n){
    vector <int> arr;
    ofstream outputfile("input.txt");
    for(int i=0; i<n; i++){
        outputfile << rand() % (n+1)<< " ";
    }
    outputfile.close();
}

// Function to run sorting algorithms and measure their execution time
void runner(int num){
    randomNums(num); // put number of numbers to generate

    clock_t startTime, endTime;
    double totalTime;

    vector <int> arr;
    ifstream readfile("input.txt");

    int value;
    while(readfile >> value){
        arr.push_back(value);
    }
    readfile.close();
```

```
    // Insertion Sort
    startTime = clock();
    insertionSort(arr, arr.size());
    endTime = clock();

    cout<<num;
    totalTime = ((double)(endTime - startTime)) / CLOCKS_PER_SEC;
    cout<<"        "<<totalTime;

    // Merge Sort
    startTime = clock();
    mergeSort(arr, 0, arr.size() - 1);
    printArr(arr, arr.size(), "mergeSort.txt");
    endTime = clock();

    totalTime = ((double)(endTime - startTime)) / CLOCKS_PER_SEC;
    cout<<"             "<<fixed << setprecision(6) << totalTime<<endl;
}

// Main function
int main(){
    cout<<"          Insertion Sort       Merge Sort"<<endl;

    // Running sorting algorithms for different input sizes
    runner(1000);
    runner(1500);
    runner(2000);
    runner(2500);
    runner(3000);
    runner(3500);
    runner(4000);
    runner(4500);
    runner(5000);

    return 0;
}
```

Name - Sahil Tiwaskar
Roll no. - 50



	Insertion Sort	Merge Sort
1000	0.003108	0.001019
1500	0.005206	0.001501
2000	0.013243	0.001571
2500	0.015047	0.002102
3000	0.019831	0.002639
3500	0.022807	0.002316
4000	0.033933	0.003397
4500	0.040434	0.003933
5000	0.060024	0.004217

```
import matplotlib.pyplot as plt
```

```
import numpy as np
num_a = np.array([500,1000,1500,2000,2500,3000,3500,4000,4500,5000
])
tc_a = np.array([0.003818,0.01499,
0.013212,0.016528,0.022382,0.029368,0.030546,0.035046,0.041716,0.04303
2])
num_b = np.array([500,1000,1500,2000,2500,3000,3500,4000,4500,5000
])
tc_b = np.array([0.083997,
0.350364,0.680607,1.16463,1.69413,3.90823,4.14539,5.9282,7.56915,8.3915
4])
plt.plot(num_a, tc_a, label='Merge sort')
plt.plot(num_b, tc_b, label='Insertion sort')
plt.xlabel('Size of array')
plt.ylabel('Time Complexity')
plt.legend()
plt.title('Merg
```