PIMPRI CHINCHWAD EDUCATION TRUST's.

PIMPRI CHINCHWAD COLLEGE OF ENGINEERING

(An Autonomous Institute)

Class : SY BTech **Acad. Yr. 2025-26** **Semester :** I

Name of the student: Samruddhi Ramswami Bansode. **PRN :** 124B1B019

Department: Computer Engineering **Division :** A

Course Name : Data Structure Lab

Course Code: BCE23PC02

Completion Date :

Assignment No. 3

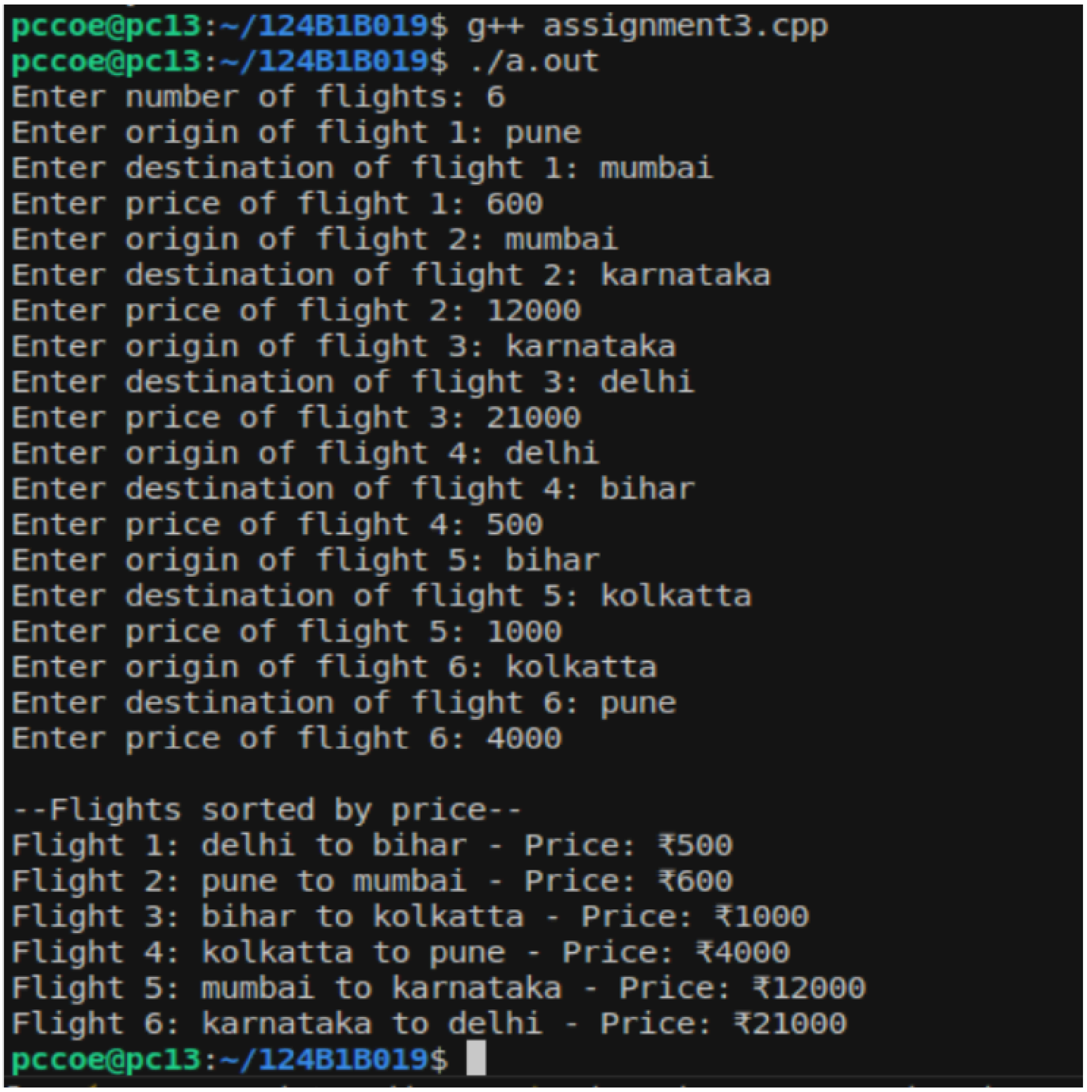
Problem Statement:

A travel booking website shows available flights sorted by ticket price so users can choose the cheapest options first. Write a program for the above scenario.
Hint:
Given an array of flight prices, implement Merge Sort to sort the prices in ascending order. Handle large data sets with many duplicate prices and analyze Merge Sort's performance on such inputs.

Source Code :

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

// Class to store flight details
class Flight {
        public:
        string origin;
        string destination;
        int price;
};

// Merge function for sorting based on price
void merge(Flight arr[], int f, int mid, int l) {
        int size1 = mid - f + 1;
        int size2 = l - mid;
        Flight left[size1], right[size2];
        for (int i = 0; i < size1; i++){
                left[i] = arr[f + i];
        }
        for (int i = 0; i < size2; i++){
                right[i] = arr[mid + 1 + i];
        }
        int i = 0, j = 0, k = f;
        while (i < size1 && j < size2) {
                if (left[i].price < right[j].price){
                        arr[k++] = left[i++];
                } else{
                        arr[k++] = right[j++];
                }
        }
        while (i < size1){
                arr[k++] = left[i++];
        }
        while (j < size2){
                arr[k++] = right[j++];
        }
}

// Merge Sort function
void mergeSort(Flight arr[], int f, int l) {
        if (f < l) {
                int mid = (f + l) / 2;
                mergeSort(arr, f, mid);
                mergeSort(arr, mid + 1, l);
                merge(arr, f, mid, l);
        }
}

int main() {
        int n;
        cout << "Enter number of flights: ";
        cin >> n;
        Flight flights[n];
        for (int i = 0; i < n; i++) {
                cout << "Enter origin of flight " << i + 1 << ": ";
                cin >> flights[i].origin;
                cout << "Enter destination of flight " << i + 1 << ": ";
                cin >> flights[i].destination;
                cout << "Enter price of flight " << i + 1 << ": ";
                cin >> flights[i].price;
        }
        // Sort flights by price
        mergeSort(flights, 0, n - 1);

        // Display sorted flights
        cout << "\n--Flights sorted by price--\n";
        for (int i = 0; i < n; i++) {
                cout << "Flight " << i + 1 << ": " << flights[i].origin
                << " to " << flights[i].destination
                << " - Price: " << flights[i].price << endl; ₹
        }
        return 0;
}
```

Screen Shot of Output :



```
pccoe@pc13:~/124B1B019$ g++ assignment3.cpp
pccoe@pc13:~/124B1B019$ ./a.out
Enter number of flights: 6
Enter origin of flight 1: pune
Enter destination of flight 1: mumbai
Enter price of flight 1: 600
Enter origin of flight 2: mumbai
Enter destination of flight 2: karnataka
Enter price of flight 2: 12000
Enter origin of flight 3: karnataka
Enter destination of flight 3: delhi
Enter price of flight 3: 21000
Enter origin of flight 4: delhi
Enter destination of flight 4: bihar
Enter price of flight 4: 500
Enter origin of flight 5: bihar
Enter destination of flight 5: kolkatta
Enter price of flight 5: 1000
Enter origin of flight 6: kolkatta
Enter destination of flight 6: pune
Enter price of flight 6: 4000

--Flights sorted by price--
Flight 1: delhi to bihar - Price: ₹500
Flight 2: pune to mumbai - Price: ₹600
Flight 3: bihar to kolkatta - Price: ₹1000
Flight 4: kolkatta to pune - Price: ₹4000
Flight 5: mumbai to karnataka - Price: ₹12000
Flight 6: karnataka to delhi - Price: ₹21000
pccoe@pc13:~/124B1B019$
```

Conclusion:

The program efficiently sorts flight details by price using the Merge Sort algorithm, ensuring `O(n log n)` performance even for large datasets. Its stability preserves the order of flights with identical prices, making it reliable for real-world travel booking applications.