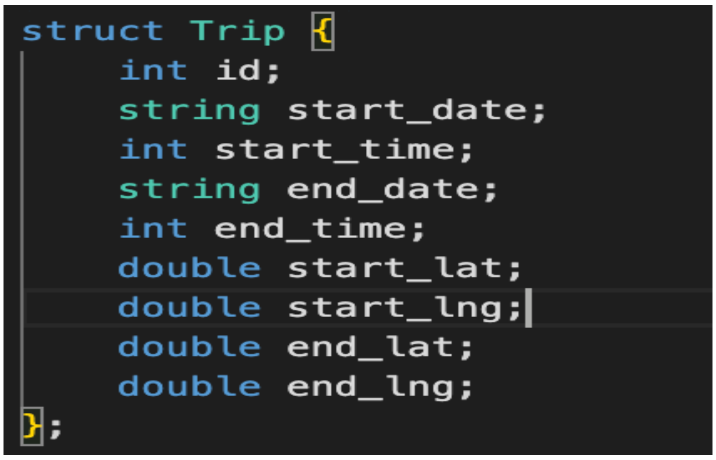
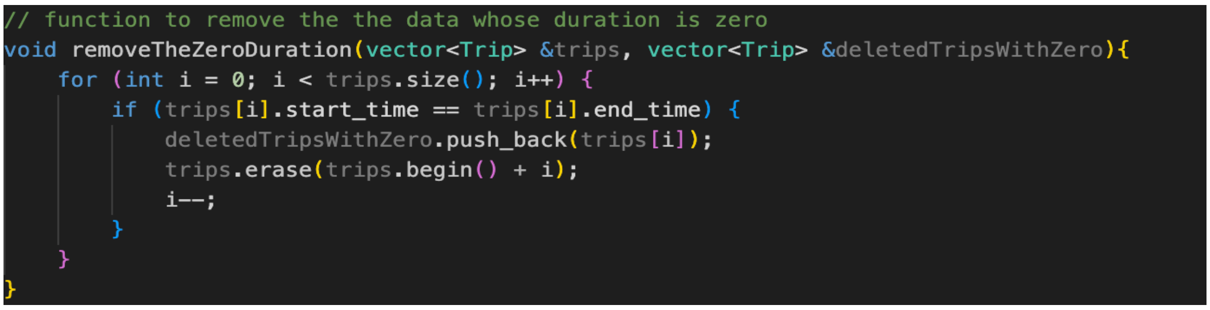
Question 1

- Firstly, I have made a data struct called **Trip** which contains the different data types as mentioned below.



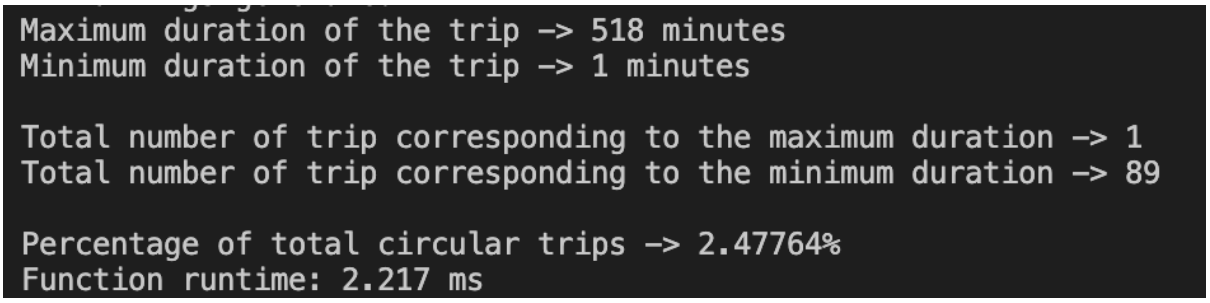
```
struct Trip {
    int id;
    string start_date;
    int start_time;
    string end_date;
    int end_time;
    double start_lat;
    double start_lng;
    double end_lat;
    double end_lng;
};
```

- Then I have created array of the above data structure type
- Then I stored the data in that array.
- Now according to question, I have created the below function to remove the data which have zero trip duration (here I am using array of name **deletedTripsZero** which store the details of the data which have zero trip duration).



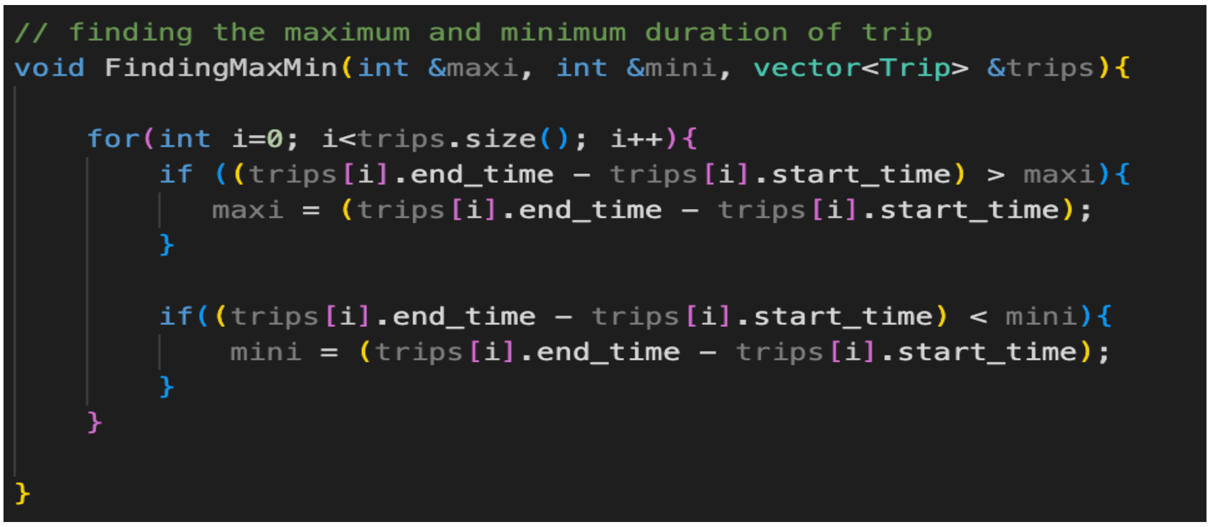
```
// function to remove the the data whose duration is zero
void removeTheZeroDuration(vector<Trip> &trips, vector<Trip> &deletedTripsWithZero){
    for (int i = 0; i < trips.size(); i++) {
        if (trips[i].start_time == trips[i].end_time) {
            deletedTripsWithZero.push_back(trips[i]);
            trips.erase(trips.begin() + i);
            i--;
        }
    }
}
```

Here are the report of the result for the 1st part of question 1.



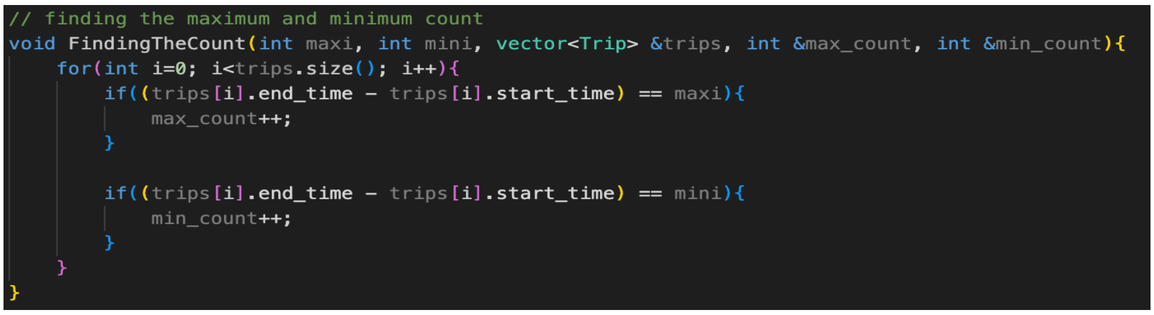
```
Maximum duration of the trip -> 518 minutes
Minimum duration of the trip -> 1 minutes

Total number of trip corresponding to the maximum duration -> 1
Total number of trip corresponding to the minimum duration -> 89

Percentage of total circular trips -> 2.47764%
Function runtime: 2.217 ms
```

- After coming to the next part of the question, here I have created the function **FindingMaxMin** which finds the maximum and minimum of the trip duration.



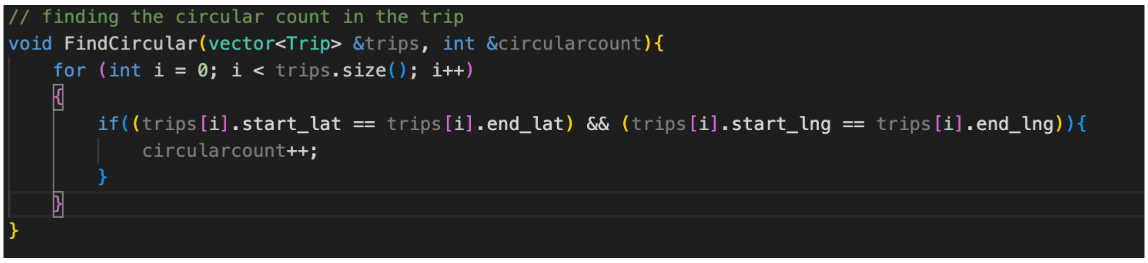
```
// finding the maximum and minimum duration of trip
void FindingMaxMin(int &maxi, int &mini, vector<Trip> &trips){

    for(int i=0; i<trips.size(); i++){
        if ((trips[i].end_time - trips[i].start_time) > maxi){
            maxi = (trips[i].end_time - trips[i].start_time);
        }

        if((trips[i].end_time - trips[i].start_time) < mini){
            mini = (trips[i].end_time - trips[i].start_time);
        }
    }

}
```

- For the next part of the question, I have created a function **FindingTheCount** which counts the number of maximum trip duration count and minimum trip duration count.



```
// finding the maximum and minimum count
void FindingTheCount(int maxi, int mini, vector<Trip> &trips, int &max_count, int &min_count){
    for(int i=0; i<trips.size(); i++){
        if((trips[i].end_time - trips[i].start_time) == maxi){
            max_count++;
        }

        if((trips[i].end_time - trips[i].start_time) == mini){
            min_count++;
        }
    }
}
```

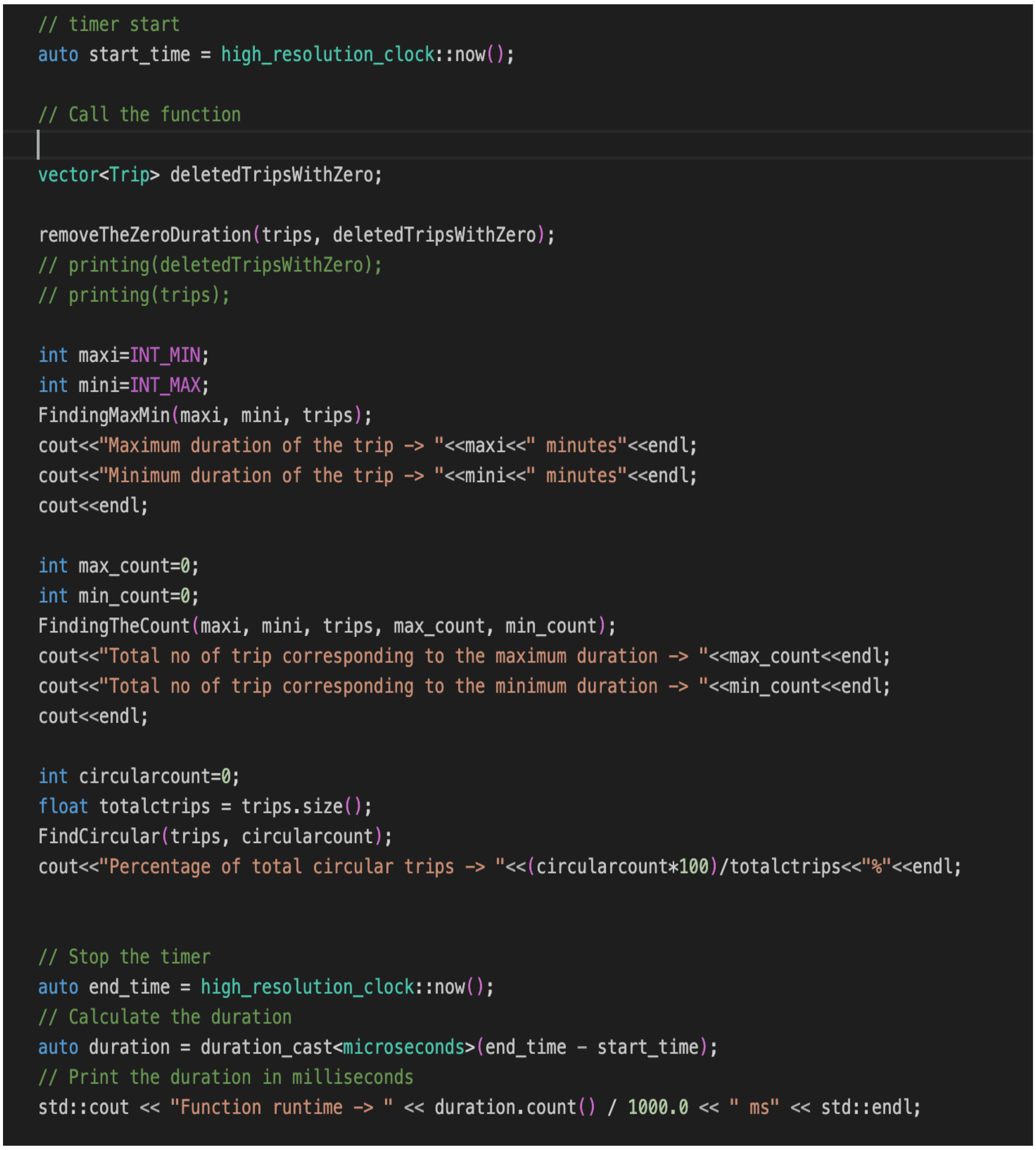
- For the next part of the question, I have created a function **FIndCircular** which finds the number of circular trips is there in the dataset.
- After that I found the percentage.



```
// finding the circular count in the trip
void FindCircular(vector<Trip> &trips, int &circularcount){
    for (int i = 0; i < trips.size(); i++)
    {
        if((trips[i].start_lat == trips[i].end_lat) && (trips[i].start_lng == trips[i].end_lng)){
            circularcount++;
        }
    }
}
```

Coming to the final part of the question 1 part 1

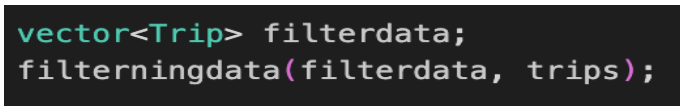
- Here I have used **chrono library** to find the total runtime for the function.



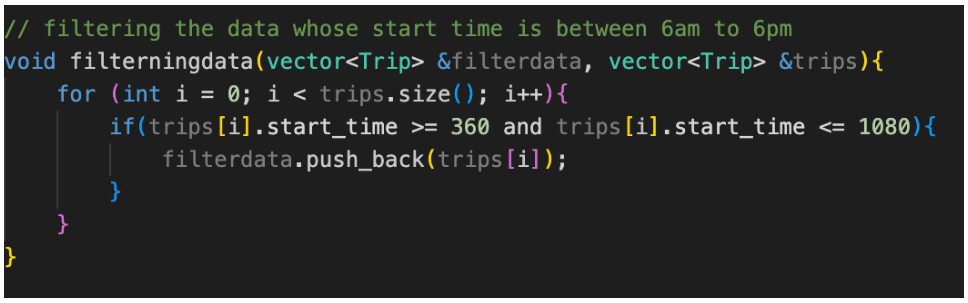
```cpp
// timer start
auto start_time = high_resolution_clock::now();

// Call the function

vector<Trip> deletedTripsWithZero;

removeTheZeroDuration(trips, deletedTripsWithZero);
// printing(deletedTripsWithZero);
// printing(trips);

int maxi=INT_MIN;
int mini=INT_MAX;
FindingMaxMin(maxi, mini, trips);
cout<<"Maximum duration of the trip -> "<<maxi<<" minutes"<<endl;
cout<<"Minimum duration of the trip -> "<<mini<<" minutes"<<endl;
cout<<endl;

int max_count=0;
int min_count=0;
FindingTheCount(maxi, mini, trips, max_count, min_count);
cout<<"Total no of trip corresponding to the maximum duration -> "<<max_count<<endl;
cout<<"Total no of trip corresponding to the minimum duration -> "<<min_count<<endl;
cout<<endl;

int circularcount=0;
float totalctrips = trips.size();
FindCircular(trips, circularcount);
cout<<"Percentage of total circular trips -> "<<(circularcount*100)/totalctrips<<"%"<<endl;


// Stop the timer
auto end_time = high_resolution_clock::now();
// Calculate the duration
auto duration = duration_cast<microseconds>(end_time - start_time);
// Print the duration in milliseconds
std::cout << "Function runtime -> " << duration.count() / 1000.0 << " ms" << std::endl;
```

Coming to the 2nd part of question 1.

- For this part I have created the array of Trip type data structures of name **filterdata** which I used to store of the filter data according to the question.
- Then I created a function **filterningdata** which filters the data according to the question condition.

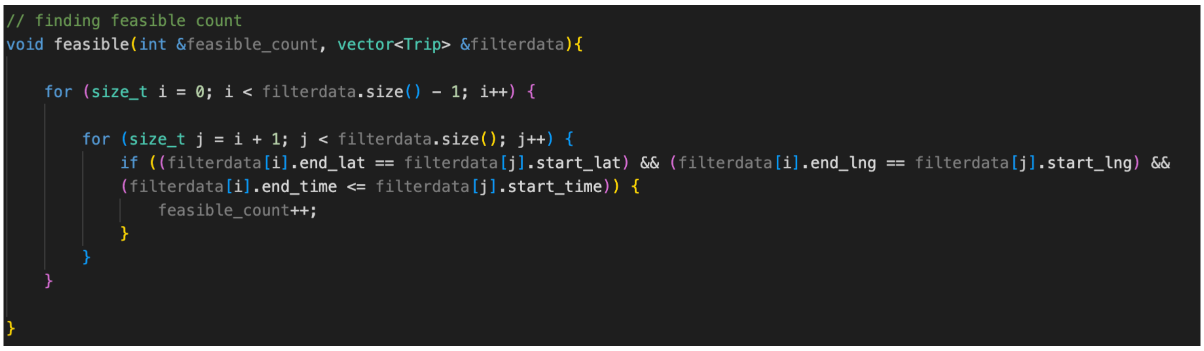


```
vector<Trip> filterdata;
filterningdata(filterdata, trips);
```



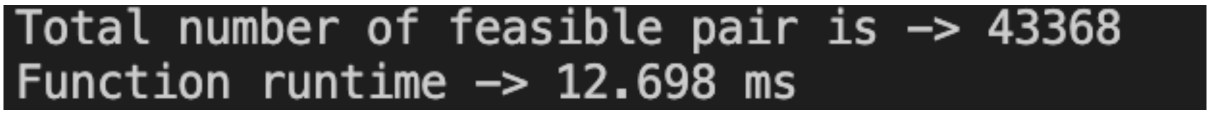
```
// filtering the data whose start time is between 6am to 6pm
void filterningdata(vector<Trip> &filterdata, vector<Trip> &trips){
    for (int i = 0; i < trips.size(); i++){
        if(trips[i].start_time >= 360 and trips[i].start_time <= 1080){
            filterdata.push_back(trips[i]);
        }
    }
}
```

- Now going forward according to the question I have created the function name feasible which counts the number of feasible pair is formed.



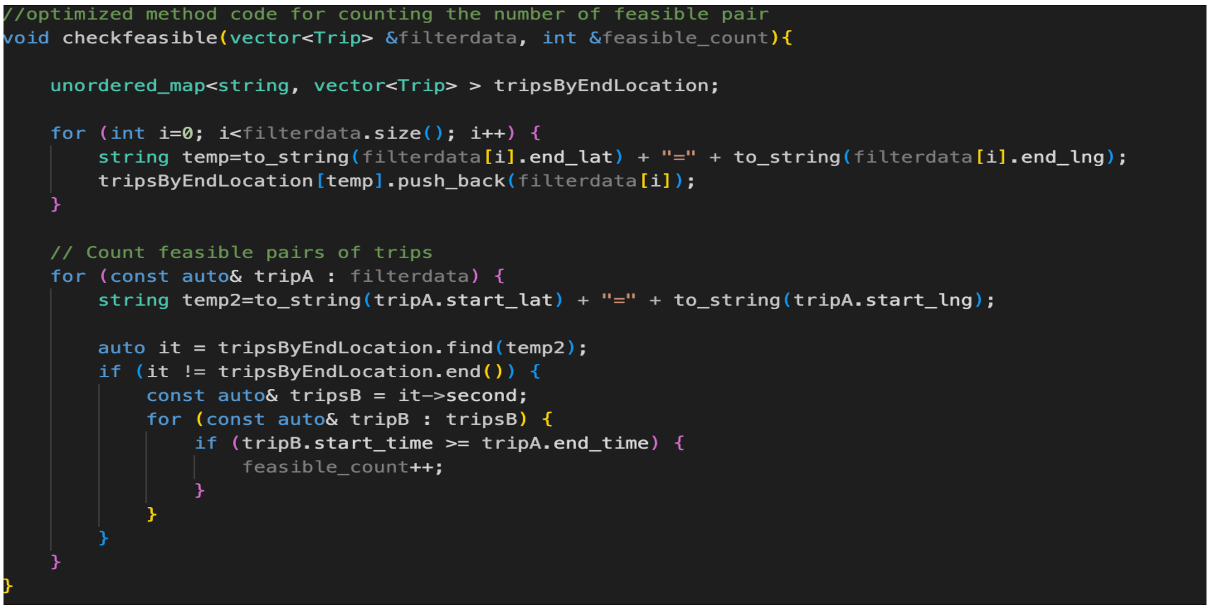
```
// finding feasible count
void feasible(int &feasible_count, vector<Trip> &filterdata){

    for (size_t i = 0; i < filterdata.size() - 1; i++) {

        for (size_t j = i + 1; j < filterdata.size(); j++) {
            if ((filterdata[i].end_lat == filterdata[j].start_lat) && (filterdata[i].end_lng == filterdata[j].start_lng) &&
            (filterdata[i].end_time <= filterdata[j].start_time)) {
                feasible_count++;
            }
        }
    }

}
```

Here are the report of the result for the 2nd part of question 1.



```
Total number of feasible pair is -> 43368
Function runtime -> 12.698 ms
```

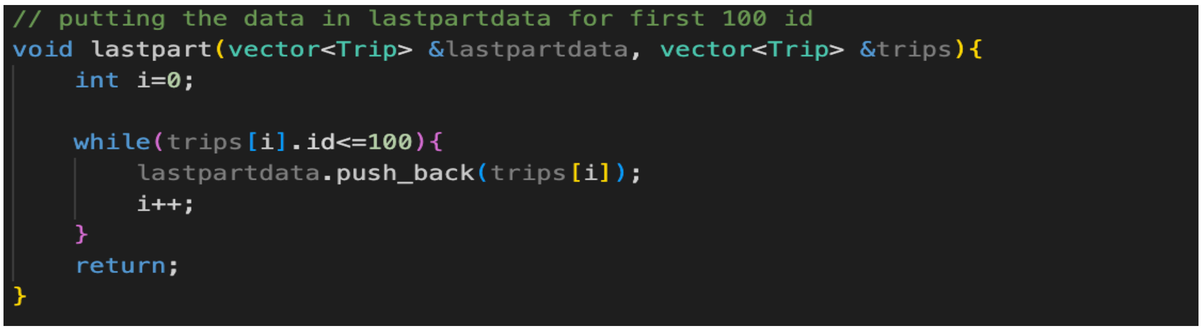
- But the above code has time complexity of **O(n²)** then I optimize the code which is the below one has time complexity of **O(n)**.



```cpp
//optimized method code for counting the number of feasible pair
void checkfeasible(vector<Trip> &filterdata, int &feasible_count){

    unordered_map<string, vector<Trip> > tripsByEndLocation;

    for (int i=0; i<filterdata.size(); i++) {
        string temp=to_string(filterdata[i].end_lat) + "=" + to_string(filterdata[i].end_lng);
        tripsByEndLocation[temp].push_back(filterdata[i]);
    }

    // Count feasible pairs of trips
    for (const auto& tripA : filterdata) {
        string temp2=to_string(tripA.start_lat) + "=" + to_string(tripA.start_lng);

        auto it = tripsByEndLocation.find(temp2);
        if (it != tripsByEndLocation.end()) {
            const auto& tripsB = it->second;
            for (const auto& tripB : tripsB) {
                if (tripB.start_time >= tripA.end_time) {
                    feasible_count++;
                }
            }
        }
    }
}
```

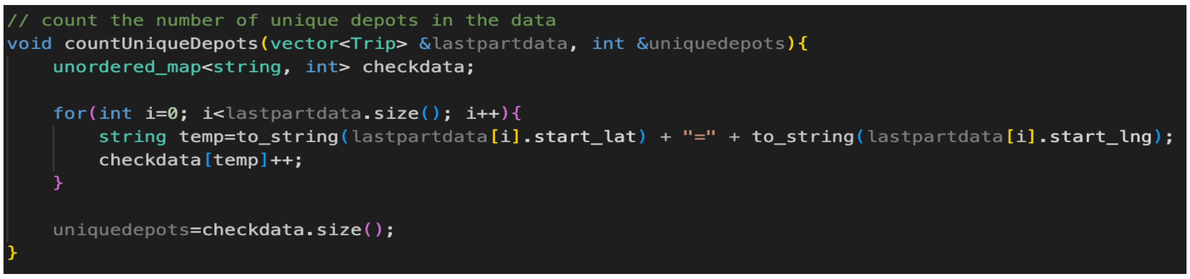
Coming to last part of the question 1

- Here question is asking to include only the first 100 trips so for that I have created the function name **lastpart** in which I am storing the first 100 data in **lastpartdata** array.



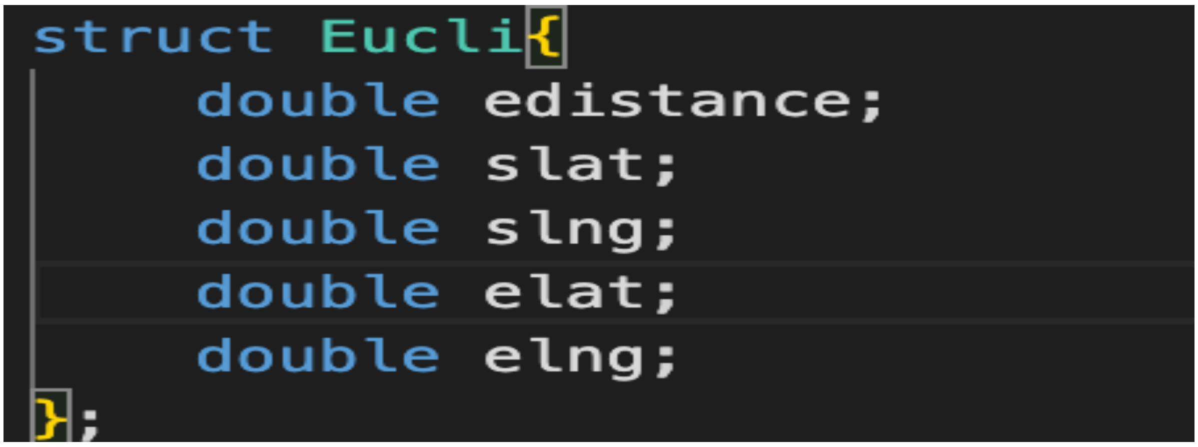
```cpp
// putting the data in lastpartdata for first 100 id
void lastpart(vector<Trip> &lastpartdata, vector<Trip> &trips){
    int i=0;

    while(trips[i].id<=100){
        lastpartdata.push_back(trips[i]);
        i++;
    }
    return;
}
```

- Now for the next part I have created the function **countUniqueDepots** where I count the number of unique depots in the dataset for the first 100 trip data.



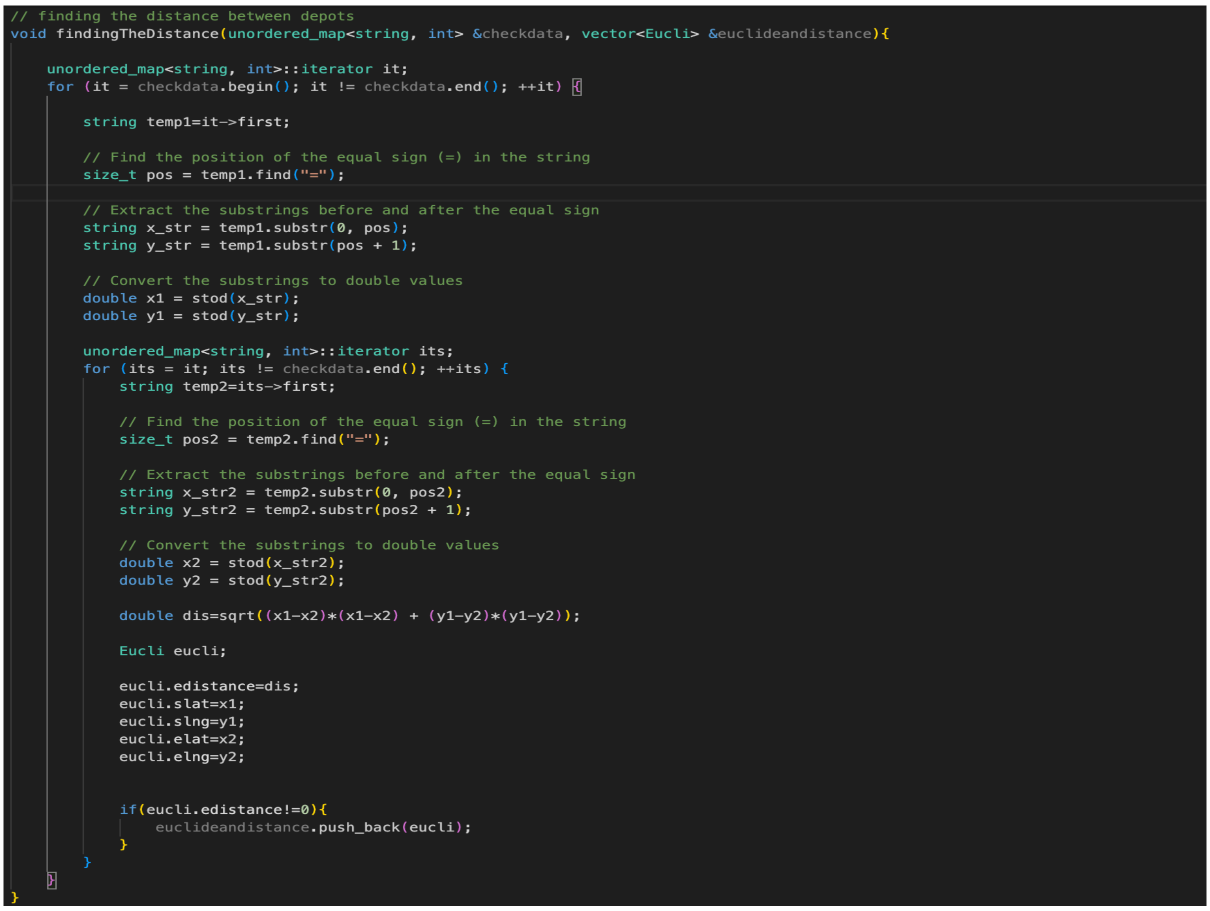
```cpp
// count the number of unique depots in the data
void countUniqueDepots(vector<Trip> &lastpartdata, int &uniquedepots){
    unordered_map<string, int> checkdata;

    for(int i=0; i<lastpartdata.size(); i++){
        string temp=to_string(lastpartdata[i].start_lat) + "=" + to_string(lastpartdata[i].start_lng);
        checkdata[temp]++;
    }

    uniquedepots=checkdata.size();
}
```

- Now coming to the next part, I have created new data structures of the name **Eucli** to store the distance, latitude, and longitude of the depots.



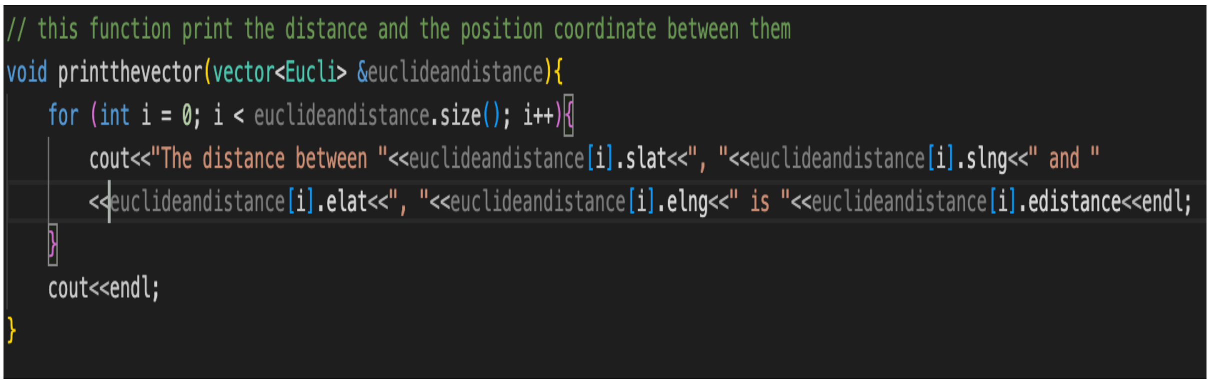
```
struct Eucli{
        double edistance;
        double slat;
        double slng;
        double elat;
        double elng;
};
```

- Then I have created a function name **findingTheDistance** which find the distance between depots and then store it in the array name of **euclideandistance**.



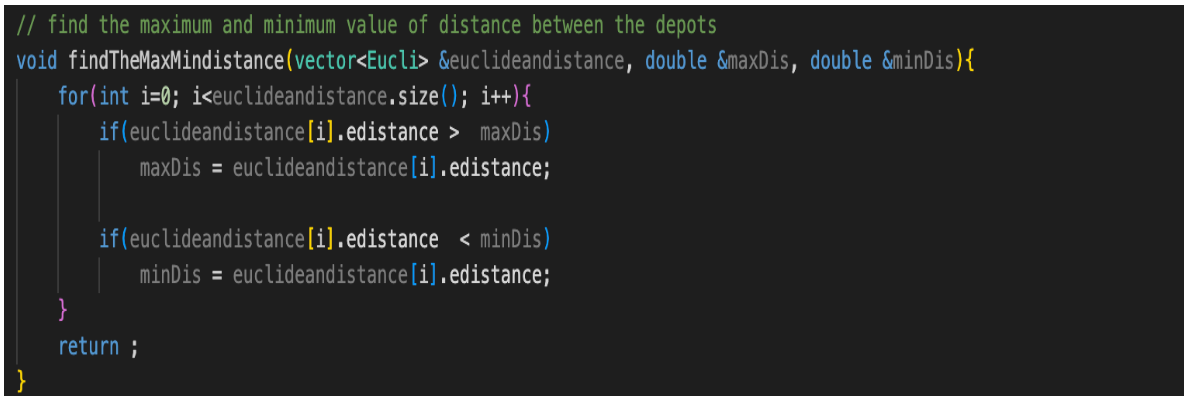
```
// finding the distance between depots
void findingTheDistance(unordered_map<string, int> &checkdata, vector<Eucli> &euclideandistance){

    unordered_map<string, int>::iterator it;
    for (it = checkdata.begin(); it != checkdata.end(); ++it) {

        string temp1=it->first;

        // Find the position of the equal sign (=) in the string
        size_t pos = temp1.find("=");

        // Extract the substrings before and after the equal sign
        string x_str = temp1.substr(0, pos);
        string y_str = temp1.substr(pos + 1);

        // Convert the substrings to double values
        double x1 = stod(x_str);
        double y1 = stod(y_str);

        unordered_map<string, int>::iterator its;
        for (its = it; its != checkdata.end(); ++its) {
            string temp2=its->first;

            // Find the position of the equal sign (=) in the string
            size_t pos2 = temp2.find("=");

            // Extract the substrings before and after the equal sign
            string x_str2 = temp2.substr(0, pos2);
            string y_str2 = temp2.substr(pos2 + 1);

            // Convert the substrings to double values
            double x2 = stod(x_str2);
            double y2 = stod(y_str2);

            double dis=sqrt((x1-x2)*(x1-x2) + (y1-y2)*(y1-y2));

            Eucli eucli;

            eucli.edistance=dis;
            eucli.slat=x1;
            eucli.slng=y1;
            eucli.elat=x2;
            eucli.elng=y2;


            if(eucli.edistance!=0){
                euclideandistance.push_back(eucli);
            }
        }
    }
}
```

- After that I have created a function for printing the data of **euclideandistance** whose name is **printthevector.**



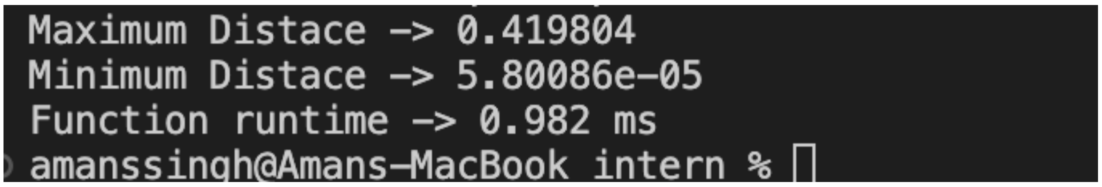
```
// this function print the distance and the position coordinate between them
void printthevector(vector<Eucli> &euclideandistance){
    for (int i = 0; i < euclideandistance.size(); i++){
        cout<<"The distance between "<<euclideandistance[i].slat<<", "<<euclideandistance[i].slng<<" and "
        <<euclideandistance[i].elat<<", "<<euclideandistance[i].elng<<" is "<<euclideandistance[i].edistance<<endl;
    }
    cout<<endl;
}
```

- After that I created a function named **findTheMaxMindistance** which finds the maximum and minimum distance between depots.



```
// find the maximum and minimum value of distance between the depots
void findTheMaxMindistance(vector<Eucli> &euclideandistance, double &maxDis, double &minDis){
    for(int i=0; i<euclideandistance.size(); i++){
        if(euclideandistance[i].edistance >  maxDis)
            maxDis = euclideandistance[i].edistance;

        if(euclideandistance[i].edistance  < minDis)
            minDis = euclideandistance[i].edistance;
    }
    return ;
}
```

Here are the report of the result for the 3rd part of question 1



```
Maximum Distace -> 0.419804
Minimum Distace -> 5.80086e-05
Function runtime -> 0.982 ms
amanssingh@Amans-MacBook intern % 
```