**OOP lab1**

**Name:** Muhammad Raza Mustafa

**Roll number:** 24k-1017 **Section:** BCS-1H

**Lab Task1**#include <iostream>

#include <string>

using namespace std;

class manageEvent {

private:

    string event1Participant[5] = {""}, event2Participant[5] = {""};

    int event1ParticipantNum = 0, event2ParticipantNum = 0;

public:

    //Task 1: Register event Participants

    void registerEventParticipant(int eventNumber) {

        //Event validator

        if (eventNumber < 1 || eventNumber > 2) {

            cout << "Invalid event number. Please enter 1 or 2." << endl;

            cin >> eventNumber;

            cin.ignore();

            registerEventParticipant(eventNumber);

            return;

        }

        string name;

        //Event 1 Registration

        if (eventNumber == 1) {

            cout << "Enter participant name for Event 1" << endl;

            for(int i = 0; i < 5; i++) {

                cout << "Participant " << i + 1 << " name: ";

                getline(cin,name);

                if(name == "") break;

                event1Participant[i] = name;

                event1ParticipantNum++;

            }

        }

        //Event 2 Registration

        if (eventNumber == 2) {

            cout << "\nEnter participant name for Event 2" << endl;

            for(int i = 0; i < 5; i++) {

                cout << "Participant " << i + 1 << " name: ";

                getline(cin,name);

                if(name == "") break;

                event2Participant[i] = name;

                event2ParticipantNum++;

            }

        }

    }

    //Task 2: Attendance Checker

    void attendanceChecker() {

        string name;

        cout<<"\nAttendance checker\nEnter name: ";

        cin>> name;

        cin.ignore();

        int found = 0;

        // Checking in Event 1

        for(int i = 0; i < event1ParticipantNum; i++) {

            if(name == event1Participant[i]) {

                cout << "Participant " << name << " is attending Event 1." << endl;

                found = 1;

                break;

            }

        }

        // Checking in Event 2

        for(int i = 0; i < event2ParticipantNum; i++) {

            if(name == event2Participant[i]) {

                cout << "Participant " << name << " is attending Event 2." << endl;

                found = 1;

                break;

            }

        }

        if(!found) {

            cout<<"Participant "<<name<<" is not attending any event."<<endl;

        }

        return;

    }

    //Task 3: Calculate Donations

    void calculateDonations() {

        cout << "\nTotal donations = " << 10 \* (event1ParticipantNum + event2ParticipantNum) << endl;

        return;

    }

    //Task 4: Display Event Details

    void displayEventDetails() {

        // Event 1: details

        cout << "\nEvent 1 Details:" << endl;

        cout << "Number of participants: " << event1ParticipantNum << endl;

        cout << "Participants: ";

        if (event1ParticipantNum == 0) cout << "None";

        for(int i = event1ParticipantNum - 1; i >= 0; i--) {

            cout << event1Participant[i] << " ";

        }

        cout << endl;

        // Event 2: details

        cout << "\nEvent 2 Details:" << endl;

        cout << "Number of participants: " << event2ParticipantNum << endl;

        cout << "Participants: ";

        if (event2ParticipantNum == 0) cout << "None";

        for(int i = event2ParticipantNum - 1; i >= 0; i--) {

            cout << event2Participant[i] << " ";

        }

        cout << endl;

    }

    //Task 5: Event popularity bar chart

    void eventPopularityBarChart() {

        cout<<"Event Popularity bar chart\n" << endl;

        cout<<"Event 1: ";

        for(int i = 1; i <= event1ParticipantNum; i++) {

            cout<< "\* ";

        }

        cout<<"\nEvent 2: ";

        for(int i = 1; i <= event2ParticipantNum; i++) {

            cout<< "\* ";

        }

        cout<< endl;

    }

};

int main() {

    manageEvent Event;

    //Event.registerEventParticipant(0) //for testing wrong input

    Event.registerEventParticipant(1);

    Event.registerEventParticipant(2);

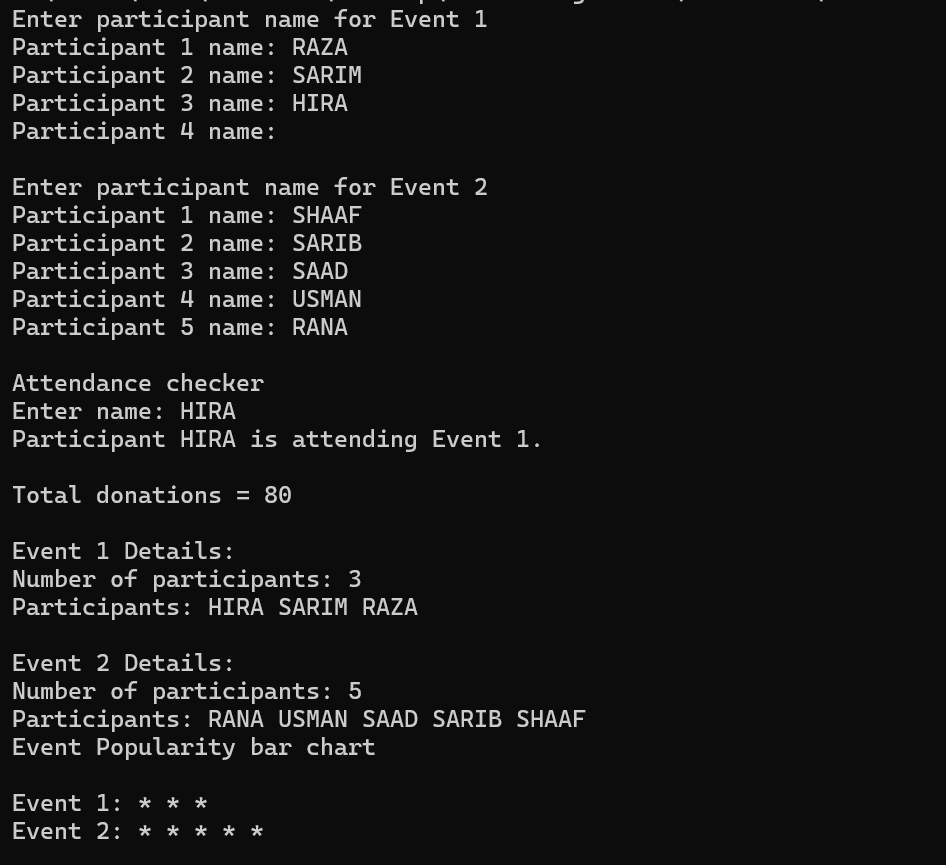
    Event.attendanceChecker();

    Event.calculateDonations();

    Event.displayEventDetails();

    Event.eventPopularityBarChart();

}

**  
Output:**

**HOME TASK:**

#include <iostream>

#include <string>

#include <iomanip>

#include <cstdlib>

#include <ctime>

using namespace std;

class AQI {

private:

    float citesAQIData[4][7];

    float maxAQIDay[4] , minAQIDay[4];

public:

    // Data Entry Function

    void dataEntry(bool useRandomData = false) {

        srand(time(0)); // Seed random number generator

        for (int i = 0; i < 4; i++) {

            cout << "City "<< i + 1;

            if(useRandomData) cout <<" Random number inputing..";

            for (int j = 0; j < 7; j++) {

                if (useRandomData) {

                    citesAQIData[i][j] = rand() % 301; // Generate random AQI between 0 and 300

                } else {

                    cout << "\nEnter AQI for Day " << j + 1 << ": ";

                    cin >> citesAQIData[i][j];

                }

            }

            cout << endl;

        }

        calculateWeeklyAQI();

    }

    float averageAQI[4];

    //Task 1: Weekly AQI tracker

    void calculateWeeklyAQI() {

        for (int city = 0; city < 4; city++) {

            float sum = 0, maxAQI = 0 , minAQI = 99999;

            for (int day = 0; day < 7; day++) {

                sum += citesAQIData[city][day];

                if (citesAQIData[city][day] > maxAQI) {

                    maxAQI = citesAQIData[city][day];

                }

                if(citesAQIData[city][day] < minAQI) {

                    minAQI = citesAQIData[city][day];

                }

            }

            maxAQIDay[city] = maxAQI;

            minAQIDay[city] = minAQI;

            averageAQI[city] = sum / 7;

        }

    }

    void printWeeklyAQI() {

        float worstAverage = 0;

        int worstCity = -1;

        cout << "\n+--------------------------- AVERAGE AQI ------------------------+" << endl;

        for (int i = 0; i < 4; i++) {

            cout << "|\t\t  City " << i + 1 << " AQI average: " << averageAQI[i] <<"\t\t\t |" << endl;

            if (averageAQI[i] > worstAverage) {

                worstAverage = averageAQI[i];

                worstCity = i;

            }

        }

        cout << "|                                                                |" << endl;

        cout << "| The worst air quality is City " << worstCity + 1

            << " with an average AQI of " << worstAverage << fixed << setprecision(3)<<" |" <<endl;

        cout << "+----------------------------------------------------------------+" << endl;

    }

    // Task2: Critical Pollution Day

    void criticalPollutionDay() {

        cout << "\n|  +---------- Critical Pollution Days Check ------------+  |" << endl;

        cout<< "|  |                                                     |  |" << endl;

        for (int i = 0; i < 4; i++) {

            bool critical = false;

            cout << "|  | City " << i + 1 << " Critical Pollution Days:                     |  |"<< endl;

            for (int j = 0; j < 7; j++) {

                if (citesAQIData[i][j] > 150) {

                    cout << "|  |   Day " << j + 1 << " AQI: " << citesAQIData[i][j] << fixed << setprecision(3)

                         << " is a critical pollution day.   |  |\n";

                    critical = true;

                }

            }

            if (!critical) {

                cout << "|  |           No critical pollution day in City            |     |" << endl;

            }

            cout<< "|  |                                                     |  |" << endl;

        }

        cout << "|  +-----------------------------------------------------+  |" << endl;

    }

    // Task3: Data Visualization

    void printWeeklyAQIChart() {

        cout << "\nWeekly AQI Chart:\n" << endl;

        for (int i = 0; i < 4; i++) {

            cout << "   AQI |" << "\tCity " << i + 1 << "\n";

            cout << "--------------------------" << endl;

            int startingPoint = ((maxAQIDay[i] / 25) + 1) \* 25;

            for (int j = startingPoint; j >= 0; j -= 25) {

                cout << "   " << right << setw(3) << j << " | ";

                for (int k = 0; k < 7; k++) {

                    if (citesAQIData[i][k] >= j) {

                        cout << "\* ";

                    } else {

                        cout << "  ";

                    }

                }

                cout << endl;

            }

            cout << "---------------------------" << endl;

            cout << "   Day | 1 2 3 4 5 6 7" << endl;

            cout << endl;

        }

    }

};

class month {

    private:

        float monthAQIAverage[4];

        AQI weeks[4];

        float maxAQI[4] , minAQI[4];

    public:

    void monthDataEntry(bool useRandomData = false) {

        for (int i = 0; i < 4; i++) {

            cout << "Week " << i +1 << endl;

            weeks[i].dataEntry(useRandomData);

            cout << endl;

        }

    }

    //Task 4: Monthly AQI Compare

    void calculateCitiesMonthlyAverageAQI() {

        //Cites

        for(int city = 0; city < 4; city++) {

            float sumAverage = 0;

            for(int week = 0; week < 4; week++) {

                sumAverage += weeks[week].averageAQI[city];

            }

            monthAQIAverage[city] = sumAverage / 4.0;

        }

    }

    void DisplayCitiesMonthlyAverageAQI() {

    float mostImprovedAQI = 0;

    int mostImprovedAQICity = -1;

    calculateCitiesMonthlyAverageAQI();

        cout <<"-------MONTHLY AVERAGE AQI --------"<<endl;

        for (int city = 0; city < 4; city++) {

            cout << "City " << city + 1 << " Monthly Average AQI: " << monthAQIAverage[city] << endl;

            // Improvement between the first and last weeks

            float improvement = weeks[3].averageAQI[city] - weeks[0].averageAQI[city];

            if (improvement > mostImprovedAQI) {

                mostImprovedAQI = improvement;

                mostImprovedAQICity = city;

            }

        }

        if (mostImprovedAQICity != -1) {

            cout << "\nThe city with the most improved AQI is City " << mostImprovedAQICity + 1

                << " with an improvement of " << mostImprovedAQI << " AQI points."  << endl;

        } else {

            cout << "\nNo significant improvement in AQI for any city." << endl;

        }

    }

    //Task 5: Generate a report

    void generateReport() {

        // Display Average

        cout << "+---------------------------REPORT--------------------------+" << endl;

        cout << "|                                                           |" << endl;

        cout << "+-----------------------Weekly Average----------------------+" << endl;

        cout << "|                                                           |" << endl;

        cout << "| Weeks    |  City 1   |  City 2   |  City 3   |  City 4    |" << endl;

        for (int week = 0; week < 4; week++) {

            cout << "| Week " << week + 1 << "   ";

            for (int city = 0; city < 4; city++) {

                cout << "| " << right << setw(8) << fixed << setprecision(3) << weeks[week].averageAQI[city] << "  ";

            }

            cout << " |" << endl;

        }

        cout << "+-----------------------------------------------------------+" << endl;

        cout << "|                                                           |" << endl;

        //Critical Pollution Day

        cout <<"+------------------Critical Pollution Day-------------------+" << endl;

        for(int week = 0; week < 4; week++) {

            cout << "|                                                           |" << endl;

            cout <<"|                           "<<"WEEK " << week + 1 <<"                          |" << endl;

            cout << "|                                                           |";

            weeks[week].criticalPollutionDay();

        }

        cout << "+-----------------------------------------------------------+" << endl;

        //Comparision

        float maxAQI[4] = {0}, minAQI[4] = {0};

        for (int week = 0; week < 4; week++) {

            for (int city = 0; city < 4; city++) {

                if (weeks[week].averageAQI[city] > maxAQI[city]) {

                    maxAQI[city] = weeks[week].averageAQI[city];

                }

                if (weeks[week].averageAQI[city] < minAQI[city]) {

                    minAQI[city] = weeks[week].averageAQI[city];

                }

            }

        }

        cout << "|                                                           |" << endl;

        cout << "+------------------------COMPARISION------------------------+" << endl;

        cout << "|                                                           |" << endl;

        cout << "|          |  City 1   |  City 2   |  City 3   |  City 4    |" << endl;

        cout << "| Highest  ";

        for(int i = 0; i < 4; i++) {

            cout << "| " << right << setw(8) << fixed << setprecision(3) << maxAQI[i] << "  ";

        }

        cout << " |" << endl;

        cout << "| Lowest   ";

        for(int i = 0; i < 4; i++) {

            cout << "| " << right << setw(8) << fixed << setprecision(3) << minAQI[i] << "  ";

        }

        cout << " |" << endl;

        cout << "| Differece";

        for(int i = 0; i < 4; i++) {

            cout << "| " << right << setw(8) << fixed << setprecision(3) << maxAQI[i] - minAQI[i] << "  ";

        }

        cout << " |" << endl;

        cout << "+-----------------------------------------------------------+" << endl;

    }

};

int main() {

    AQI aqi;

    month monthData;

    char choice;

    cout << "Do you want to use random data for AQI tracking? (y/n): ";

    cin >> choice;

    if (choice == 'y' || choice == 'Y') {

        aqi.dataEntry(true);

    } else {

        aqi.dataEntry();

    }

    aqi.printWeeklyAQI();

    aqi.criticalPollutionDay();

    aqi.printWeeklyAQIChart();

    cout << "Do you want to use random data for AQI tracking? (y/n): ";

    cin >> choice;

    system("cls");

    if (choice == 'y' || choice == 'Y') {

        monthData.monthDataEntry(true);

    } else {

        monthData.monthDataEntry();

    }

    monthData.DisplayCitiesMonthlyAverageAQI();

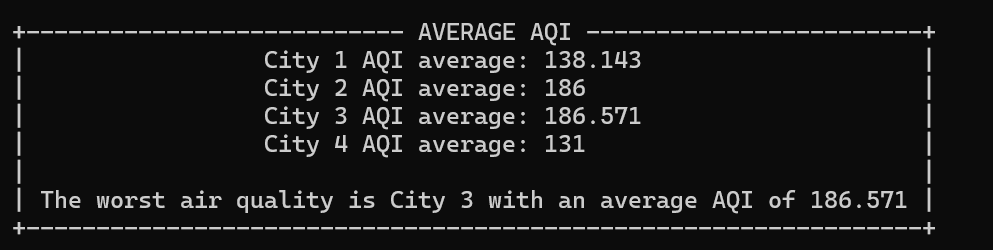
    cout<<"\n"<<endl;

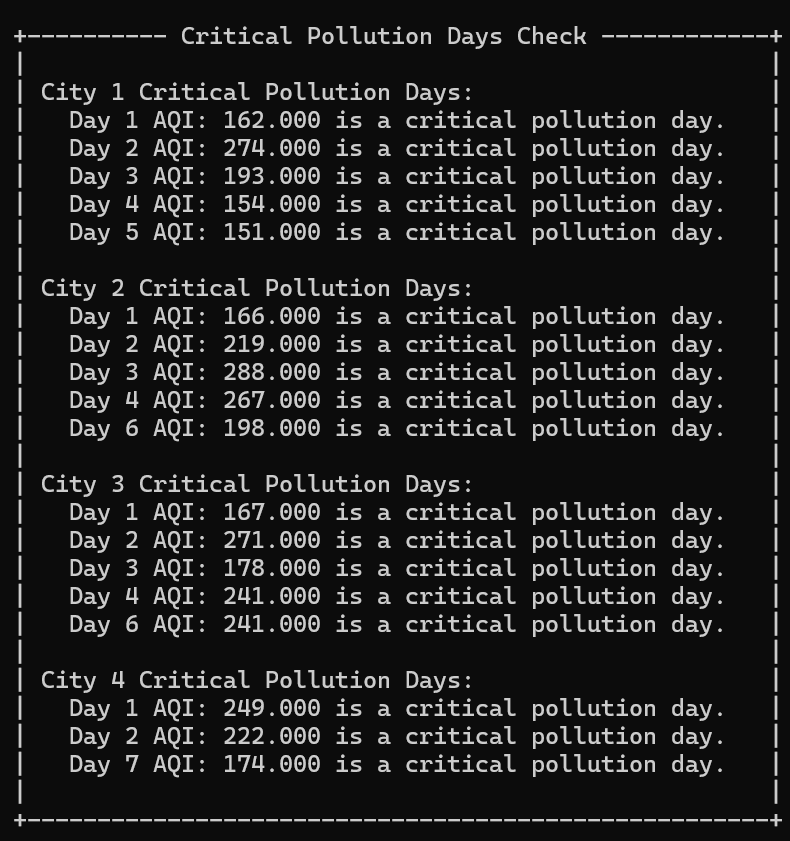
    monthData.generateReport();

    return 0;

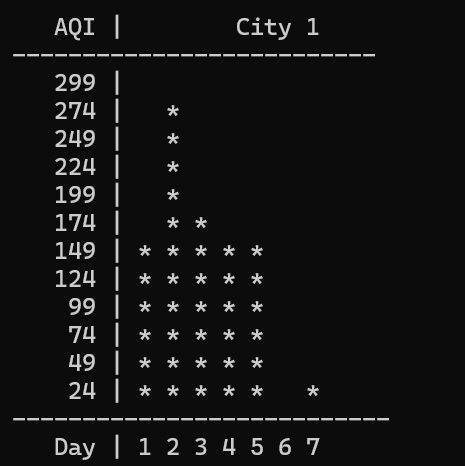
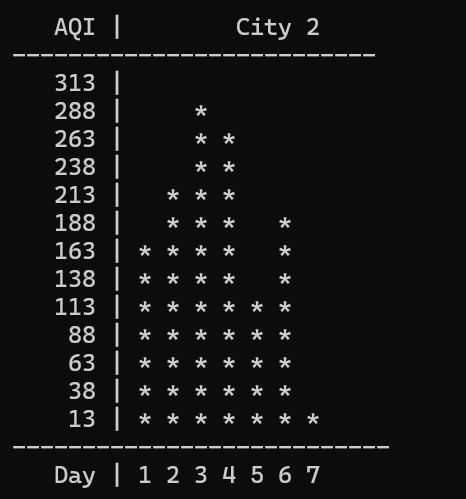
}

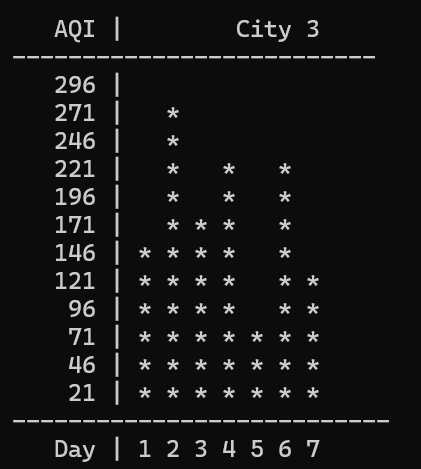
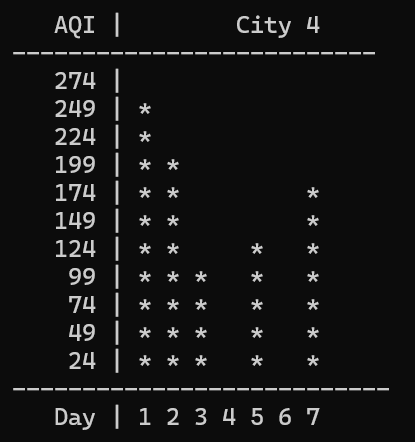
**OUTPUT**

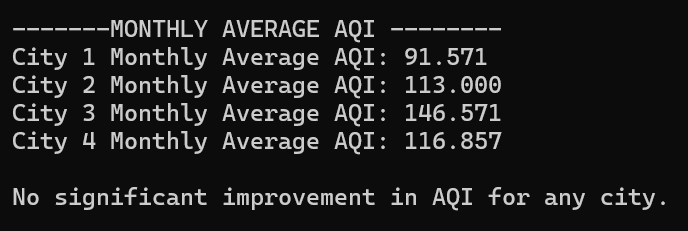
**Task 1**

**Task 2**

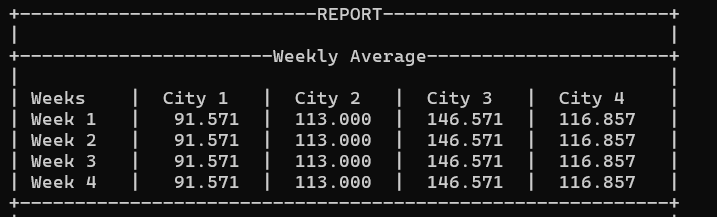
**Task 3  
  
 **

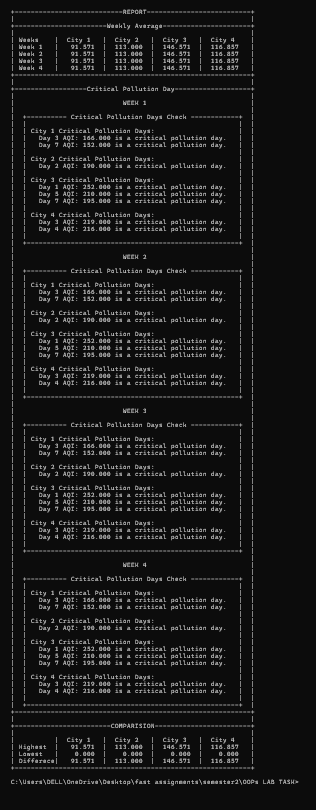
**** ****

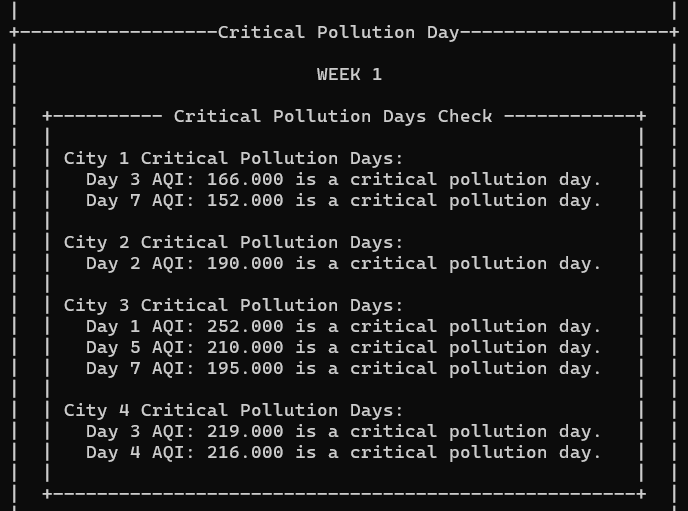
**** ****

Task 4:  
  
 ****

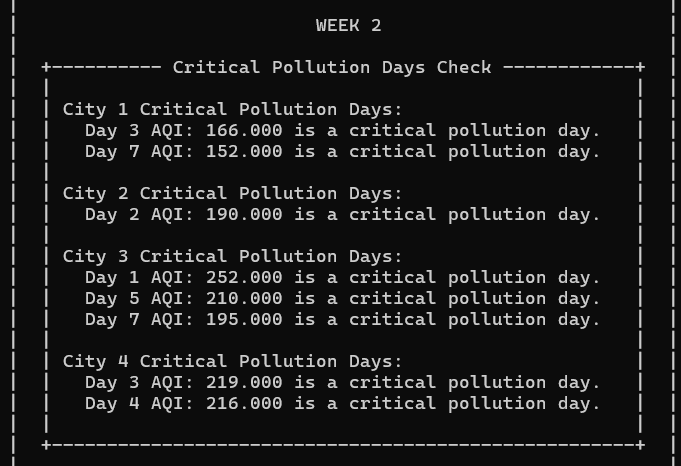
Task 5:

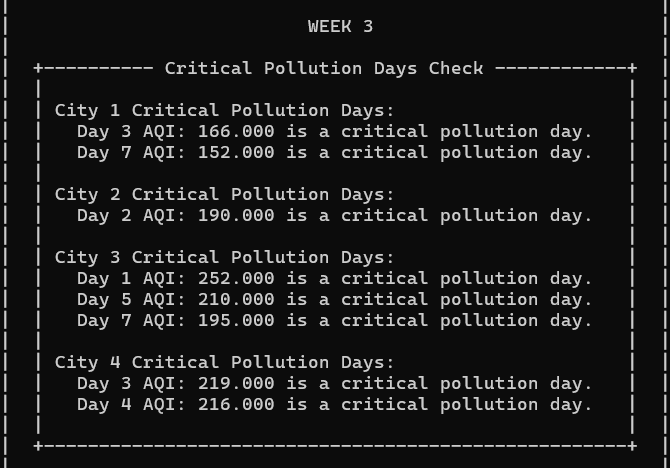
****

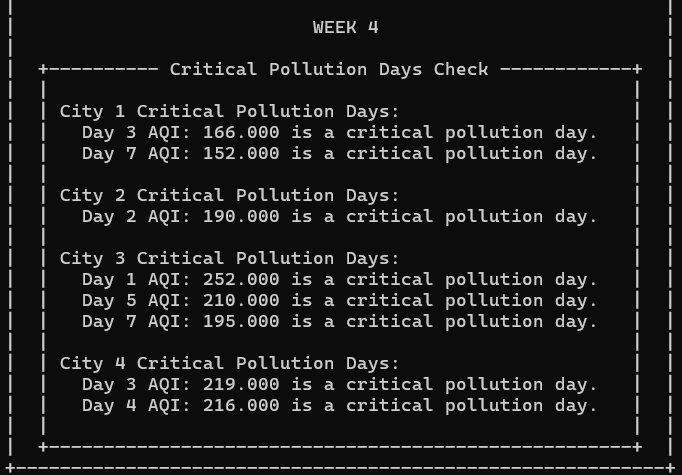
****

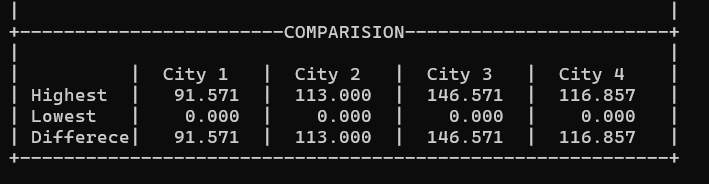
**** complete Report look

like this

****

****

****

****