**Homework 4**

**John Ingram**

**Thomas Bunney**

**Daniel Balding**

**Problem 1:**

**Using the original AL Weather Station Data file find all records that have a bad data flag (-9999) for either the PRCP, TMAX or TMIN fields. Produce a new data file that omits those records with bad data flags.**

**Code:**

/\* John Ingram

   Thomas Bunney

   Daniel Balding\*/

#include <iostream>

#include <string>

#include <fstream>

#include <iomanip>

#include <algorithm>

#include <cctype>

#include <cstdlib>

#include <cmath>

using namespace std;

bool filegood(fstream &handle, string fname);

void find\_max\_min(double tmax[], double tmin[], int elements, double &max\_val, double &min\_val);

float total\_precip\_by\_station(float prcp[], int elements, string answer, string dataline, string station[]);

float total\_precip\_day(float prcp[], unsigned int days[], int elements, int DOM);

float total\_precip\_range(float prcp[], unsigned int days[], int elements, int DOM, int start\_day, int end\_day);

int main()

{

    fstream infile, outfile;

    string infilename = "C:/Users/owner/Documents/AL\_Weather\_Station.txt";

    string outfilename = "C:/Users/owner/Documents/Filtered\_AL\_Weather\_Station.txt";

    string firstline = "";

    string dataline = "";

    string s\_tmax, s\_tmin, s\_prcp;

    double tmax[9000], tmin[9000];

    double max = 0.0, min = 0.0;

    int records = 0;

    int pos\_tmax, pos\_tmin, pos\_prcp, pos\_mdpr, pos\_dapr, pos\_snwd, pos\_snow, pos\_awnd;

    cout << "WEATHER STATION DATA" << endl << endl;

    cout << "Open the weather data file." << endl << endl;

    infile.open("C:/Users/owner/Documents/AL\_Weather\_Station.txt");

    if (!filegood(infile, infilename))

        return 2;

    cout << "Read the first line from the file. " << endl;

    getline(infile, firstline);

    cout << firstline << endl;

    outfile.open("C:/Users/owner/Documents/Filtered\_AL\_Weather\_Station.txt");

    pos\_mdpr = firstline.find("MDPR");

    if (pos\_mdpr != string::npos)

    {

        cout << "The position of column MDPR is " << pos\_mdpr << endl;

    }

    else

    {

        cout << "Error: MDPR not found." << endl;

        system("pause");

        return 5;

    }

    pos\_dapr = firstline.find("DAPR");

    if (pos\_dapr != string::npos)

    {

        cout << "The position of column DAPR is " << pos\_dapr << endl;

    }

    else

    {

        cout << "Error: DAPR not found." << endl;

        system("pause");

        return 6;

    }

    pos\_snwd = firstline.find("SNWD");

    if (pos\_snwd != string::npos)

    {

        cout << "The position of column SNWD is " << pos\_snwd << endl;

    }

    else

    {

        cout << "Error: SNWD not found." << endl;

        system("pause");

        return 7;

    }

    pos\_snow = firstline.find("SNOW");

    if (pos\_snow != string::npos)

    {

        cout << "The position of column SNOW is " << pos\_snow << endl;

    }

    else

    {

        cout << "Error: SNOW not found." << endl;

        system("pause");

        return 8;

    }

    pos\_awnd = firstline.find("AWND");

    if (pos\_awnd != string::npos)

    {

        cout << "The position of column AWND is " << pos\_awnd << endl;

    }

    else

    {

        cout << "Error: AWND not found." << endl;

        system("pause");

        return 9;

    }

    pos\_tmax = firstline.find("TMAX");

    if (pos\_tmax != string::npos)

    {

        cout << "The position of column TMAX is " << pos\_tmax << endl;

    }

    else

    {

        cout << "Error: TMAX not found." << endl;

        system("pause");

        return 3;

    }

    pos\_tmin = firstline.find("TMIN");

    if (pos\_tmin != string::npos)

    {

        cout << "The position of column TMIN is " << pos\_tmin << endl;

    }

    else

    {

        cout << "Error: TMIN not found." << endl;

        system("pause");

        return 4;

    }

    pos\_prcp = firstline.find("PRCP");

    if (pos\_prcp != string::npos)

    {

        cout << "The position of column PRCP is " << pos\_prcp << endl;

    }

    else

    {

        cout << "Error: PRCP not found." << endl;

        system("pause");

        return 1;

    }

    outfile << firstline << endl;

    system ("pause");

    cout << "Read the second line from the file. " << endl;

    getline(infile, dataline);

    cout << dataline << endl;

    system ("pause");

    cout << "Now get first line of data." << endl << endl;

    getline(infile, dataline);

    cout << dataline << endl;

        while (!infile.eof())

    {

        // Extract the TMAX and TMIN values

        s\_tmax = dataline.substr(pos\_tmax, 5);

        s\_tmin = dataline.substr(pos\_tmin, 5);

        s\_prcp = dataline.substr(pos\_prcp, 5);

        // Check for bad data

        if (s\_tmax != "-9999" && s\_tmin != "-9999" && s\_prcp != "-9999") // Good record

        {

            cout << dataline << endl;

            outfile << dataline << endl;

            tmax[records] = stod(s\_tmax) / 10.0;

            tmin[records] = stod(s\_tmin) / 10.0;

            records++;

        }

        getline(infile, dataline);

    }

    cout << "There are " << records << " good data records for tmax and tmin" << endl;

    infile.close();

    find\_max\_min(tmax, tmin, records, max, min);

    cout << fixed << setprecision(2);

    cout << "The max temperature is " << max << endl;

    cout << "The min temperature is " << min << endl;

    outfile.close();

    cout << "\n\n";

    system("pause");

    return 0;

    return 0;

}

void find\_max\_min(double tmax[], double tmin[], int elements, double &max\_val, double &min\_val)

{

    max\_val = tmax[0];

    min\_val = tmin[0];

    for (int k = 1; k < elements; k++)

    {

        if (tmax[k] > max\_val)

            max\_val = tmax[k];

        if (tmin[k] < min\_val)

            min\_val = tmax[k];

    }

    return;

}

//

// Function fileopen

//

bool filegood(fstream &handle, string fname)

{

    if (!handle)

    {

        cout << "Unable to open the file " << fname << ". Check path and permissions. " << endl;

        system("pause");

        return false;

    }

    else

    {

        cout << "File " << fname << " was opened successfully." << endl;

        return true;

    }

}

float total\_precip\_day(float prcp[], unsigned int days[], int elements, int DOM)

{

    float total\_prcp = 0.0;

    int k = 0;

    for (k = 0; k < elements; k++)

    {

        if (days[k] == DOM)

        {

            total\_prcp += prcp[k];

        }

    }

    return total\_prcp;

}

float total\_precip\_range(float prcp[], unsigned int days[], int elements, int DOM, int start\_day, int end\_day)

{

    float total\_prcp = 0.0;

    int k = 0;

    for (k = 0; k < elements; k++)

    {

        if ((days[k] >= start\_day) && (days[k] <= end\_day))

        {

            total\_prcp += prcp[k];

        }

    }

    return total\_prcp;

}

//Add function for part 3

float total\_precip\_by\_station(float prcp[], int elements, string answer, string dataline, string station[])

{

    string s\_station;

    float total\_prcp = 0.0;

    for(int i = 0; i < elements; i++)

    {

        if(station[i].find(answer) != string::npos)

        {

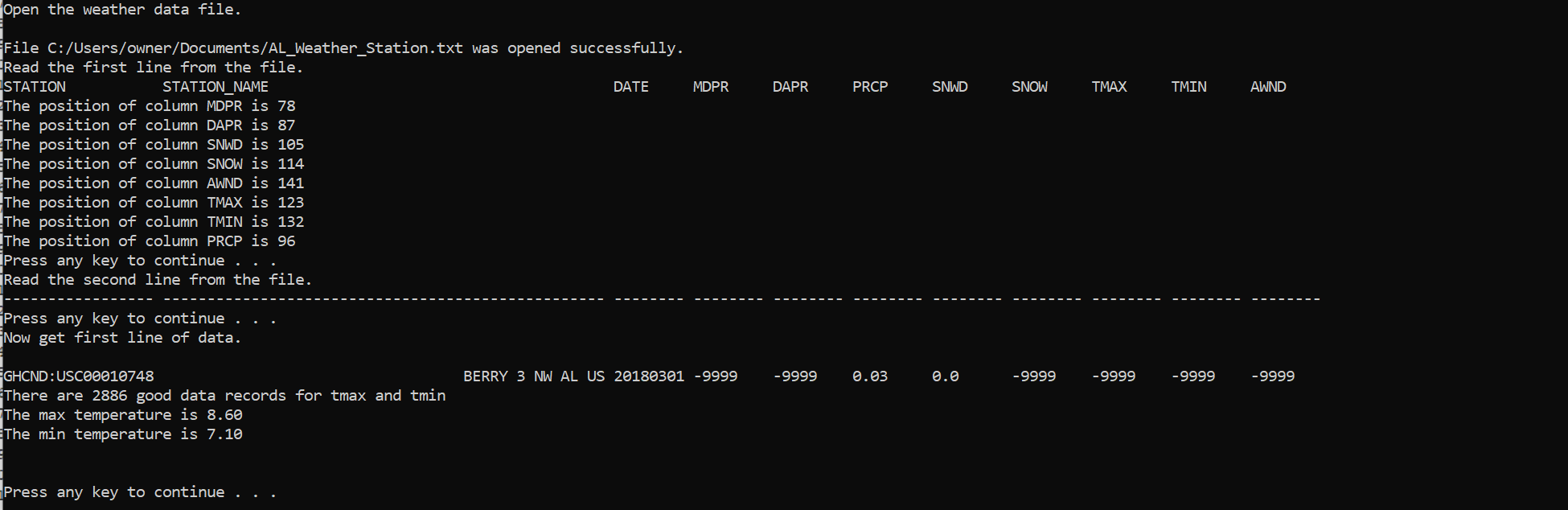
            total\_prcp += prcp[i];

        }

    }

    return total\_prcp;

}

**Output:**

**Problem 2:**

**Using the filtered data file from problem 1, create another file (weather\_station\_five\_column.txt) with only the following five columns of information:**

**STATION NAME           DATE               PRCP          TMAX              TMIN**

**Add spaces to separate the three parts of the date field and convert the temperatures from Celsius to Fahrenheit**

**Code:**

/\* John Ingram

   Thomas Bunney

   Daniel Balding\*/

#include <iostream>

#include <string>

#include <fstream>

#include <iomanip>

#include <algorithm>

#include <cctype>

#include <cstdlib>

#include <cmath>

using namespace std;

int main()

{

    fstream infile, outfile;

    infile.open("C:/Users/owner/Documents/Filtered\_AL\_Weather\_Station.txt");

    outfile.open("C:/Users/owner/Documents/weather\_station\_five\_column.txt");

    string newline = "";

    string s\_tmax, s\_tmin, s\_date;

    float tmax, tmin;

    int pos\_tmax, pos\_tmin, pos\_prcp, pos\_mdpr, pos\_dapr, pos\_snwd, pos\_snow, pos\_awnd, counter = 0, pos\_date;

    getline(infile, newline);

    newline.erase(0, 18);

    pos\_mdpr = newline.find("MDPR");

    newline.erase(pos\_mdpr, 8);

    pos\_dapr = newline.find("DAPR");

    newline.erase(pos\_dapr, 8);

    pos\_snwd = newline.find("SNWD");

    newline.erase(pos\_snwd, 8);

    pos\_snow = newline.find("SNOW");

    newline.erase(pos\_snow, 8);

    pos\_awnd = newline.find("AWND");

    newline.erase(pos\_awnd, 8);

    pos\_tmax = newline.find("TMAX");

    pos\_tmin = newline.find("TMIN");

    pos\_date = newline.find("DATE");

    outfile << newline << endl;

    getline(infile, newline);

    while(!infile.eof())

    {

            newline.erase(0, 18);

            newline.erase(pos\_mdpr, 8);

            newline.erase(pos\_dapr, 8);

            newline.erase(pos\_snwd, 8);

            newline.erase(pos\_snow, 8);

            newline.erase(pos\_awnd, 8);

            s\_tmax = newline.substr(pos\_tmax, 2);

            s\_tmin = newline.substr(pos\_tmin, 2);

            tmax = stof(s\_tmax) / 10.0;

            tmin = stof(s\_tmin) / 10.0;

            tmax = (tmax \* (9 / 5)) + 32;

            tmin = (tmin \* (9 / 5)) + 32;

            s\_tmax = to\_string(tmax);

            s\_tmin = to\_string(tmin);

            s\_date = newline.substr(pos\_date, 8);

            s\_date.insert(4," ",1);

            s\_date.insert(7," ",1);

            newline.replace(pos\_date, s\_date.length(), s\_date);

            newline.replace(pos\_tmax, s\_tmax.length(), s\_tmax);

            newline.replace(pos\_tmin, s\_tmin.length(), s\_tmin);

            outfile.precision(2);

            outfile << setprecision(2) << newline << endl;

            counter++;

            getline(infile, newline);

    }

    cout << counter << endl;

    infile.close();

    outfile.close();

    return 0;

}

**Output:**

**See file.**

**Problem 3:**

**Using the weather\_station\_five\_column.txt file produced by problem two above:**

**Determine the highest and lowest temperatures in the state for the entire month of March 2018.**

**Print out (to the screen) the station names with highest and lowest temperatures along with the high and low values. (Multiple stations may record the same values for high and low.)**

/\* John Ingram

   Thomas Bunney

   Daniel Balding\*/

#include <iostream>

#include <fstream>

#include <iomanip>

#include <string>

#include <vector>

#include <cstring>

using namespace std;

void getInfo(string inFileName, vector<string> &names, vector<string> &times, vector<double> &maxTemps, vector<double> &minTemps);

int main(void)

{

    string inFileName = ".\\weather\_station\_five\_column.txt";

    double max = -999999999.9, min = 99999999.9;

    vector<string> names, minNames , maxNames, dates, minTimes, maxTimes;

    vector<double>  maxTemps, minTemps;

    getInfo(inFileName, names, dates, maxTemps, minTemps);

    for(int i = 0; i < names.size(); i++)

    {

        if(maxTemps[i] > max)

        {

            max = maxTemps[i];

            maxNames.clear();

            maxTimes.clear();

            maxNames.push\_back(names[i]);

            maxTimes.push\_back(dates[i]);

        } else if(maxTemps[i] == max) {

            maxNames.push\_back(names[i]);

            maxTimes.push\_back(dates[i]);

        }

        if(minTemps[i] < min)

        {

            min = minTemps[i];

            minNames.clear();

            minTimes.clear();

            minNames.push\_back(names[i]);

            minTimes.push\_back(dates[i]);

        } else if(minTemps[i] == min) {

            minNames.push\_back(names[i]);

            minTimes.push\_back(dates[i]);

        }

    }

    cout << "The highest temperature in the state was " << max <<" It was measuread at the following stations: " << endl;

    for(int i = 0; i < maxNames.size(); i++) cout << maxNames[i] << " on "<< maxTimes[i] << endl;

    cout << endl;

    cout << "The lowest temperature in the state was "<< min <<" It was measuread at the following stations: " << endl;

    for(int i = 0; i < minNames.size(); i++) cout << minNames[i] << " on "<< minTimes[i] << endl;

    cout << endl;

    system("pause");

    return 0;

}

void getInfo(string inFileName, vector<string> &names, vector<string> &times, vector<double> &maxTemps, vector<double> &minTemps)

{

    ifstream file;

    string identifiers [3] = {"DATE", "TMAX", "TMIN"};

    string months [12] = {"January", "February", "March", "April", "May", "June", "July",

                         "August", "September", "October", "November", "December"};

    int positions [3] = {}; // Note that the first position is the end, while the others are beginnings

    int month;

    string line, rawDate, date;

    file.open(inFileName);

    if (!file)

    {

        cout << "Error. Not able to open file.";

        system("pause");

        exit(1);

    }

    // Set initial positions

    getline(file, line);

    positions[0] = line.find(identifiers[0]) - 7;

    positions[1] = line.find(identifiers[1]);

    positions[2] = line.find(identifiers[2]);

    // Put data into vectors

    while(!file.eof())

    {

        getline(file, line);

        if (line.length() > positions[1])

        {

            date = "";

            names.push\_back(line.substr(0,positions[0])); //Add the station name to the vector

            rawDate = line.substr(positions[0]+13, 5);

            month = (int)atof(rawDate.substr(0,2).c\_str());

            date.append(months[month-1]);

            date.append(" ");

            date.append(rawDate.substr(2,3));

            times.push\_back(date); // Add the date to the vector

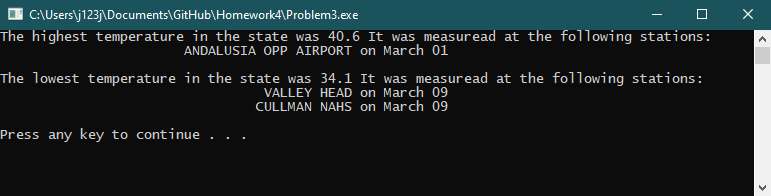
            maxTemps.push\_back((double)atof((line.substr(positions[1], positions[2]-positions[1])).c\_str())); // add max to  the vector

            minTemps.push\_back((double)atof((line.substr(positions[2], string::npos)).c\_str())); // add min to  the vector

        }

    }

}

**Output:**

**Problem 4:**

**Using the five-column data file produced from problem 2 write a program that gives the user five options as described by a-e.**

**The program should provide a menu showing each of the five options. The user then selects one of the five options. Depending on the option the user will be prompted to enter the corresponding parameters. Then those parameters will be passed to a function to do the work.**

**Code:**

#include <iostream>

#include <string>

#include <fstream>

#include <iomanip>

#include <algorithm>

#include <cctype>

#include <cstdlib>

#include <cmath>

/\* John Ingram

   Thomas Bunney

   Daniel Balding\*/

using namespace std;

float total\_precip\_day(float prcp[], unsigned int days[], int elements, int DOM);

float total\_precip\_range(float prcp[], unsigned int days[], int elements, int DOM, int start\_day, int end\_day);

//Declare function 3

float total\_precip\_by\_station(float prcp[], int elements, string answer, string dataline, string station[]);

//Declare function 4

float maxTemp(float temps[], int elements, string stations[], string station);

float minTemp(float temps[], int elements, string stations[], string station);

float sumOfTemp(float temps[], int elements, string stations[], string station);

//Declare function 5

float minTemp(float temps[], int elements, string stations[], string station, unsigned int startDay, unsigned int endDay, unsigned int days[]);

float maxTemp(float temps[], int elements, string stations[], string station, unsigned int startDay, unsigned int endDay, unsigned int days[]);

float avgOfTemp(float temps[], int elements, string stations[], string station, unsigned int startDay, unsigned int endDay, unsigned int days[]);

int main(void)

{

    string dataline = "";

    string station[3000];

    string s\_year, s\_month, s\_day;

    string s\_tmax, s\_tmin, s\_prcp, station\_name;

    string menu[6] = { "1. Total precipition from all stations for a given day",

                      "2. Total precipitation over a date range. ",

                      "3. Total precipitation by station for March. ",

                      "4. Temperature max, min and average by station for March.",

 "5. Temperature max, min and average by station for date range.",

 "6. Quit." };

    unsigned int j = 0, k = 0;

    unsigned int pos\_station\_name = 0;

    unsigned int pos\_date = 0;

    unsigned int pos\_prcp = 0;

    unsigned int pos\_tmax = 0;

    unsigned int pos\_tmin = 0;

    unsigned int option = 0;

    unsigned int dayofmonth = 0, daysofmarch[3000];

    unsigned int start\_day = 0, end\_day = 0;

    float tmax = 0, tmin = 0, tavg\_hi = 0, tavg\_low = 0, prcp[2900] = { 0 };

    float tmaxF[2900] = { 0 }, tminF[2900] = { 0 };

    float max = 0, min = 0;

    ifstream infile;

    cout << "SUBSET of WEATHER STATION DATA" << endl << endl;

    cout << "Open the five column weather data file." << endl << endl;

    infile.open("weather\_station\_five\_column.txt");

    if (!infile)

    {

        cout << "Unable to open the input file. " << endl;

        system("pause");

        return(1);

    }

    else

    {

        cout << "Data file opened." << endl;

    }

    cout << "Use the first line of the file to find the column positions. " << endl;

    getline(infile, dataline);

    pos\_station\_name = dataline.find("STATION\_NAME");

    pos\_date = dataline.find("DATE");

    pos\_prcp = dataline.find("PRCP");

    pos\_tmax = dataline.find("TMAX");

    pos\_tmin = dataline.find("TMIN");

    cout << "Now get the data records." << endl << endl;

    j = 0;

    getline(infile, dataline);

    while (!infile.eof())

    {

        station[j] = dataline.substr(0, 44);

        s\_year = dataline.substr(pos\_date, 4);

        s\_month = dataline.substr(pos\_date + 5, 2);

        s\_day = dataline.substr(pos\_date + 8, 2);

        daysofmarch[j] = stoi(s\_day);

        s\_prcp = dataline.substr(pos\_prcp, 5);

        s\_tmax = dataline.substr(pos\_tmax, 5);

        s\_tmin = dataline.substr(pos\_tmin, 5);

        prcp[j] = stof(s\_prcp);

        tmaxF[j] = stof(s\_tmax);

        tminF[j] = stof(s\_tmin);

        j++;

        getline(infile, dataline);

    }

    cout << "There are " << j << " records in the new file." << endl;

    infile.close();

    cout << endl << endl;

    cout << fixed << setprecision(2);

    while (option != 6)

    {

        cout << "Select an option from the following menu." << endl << endl;

        for (k = 0; k < 6; k++)

            cout << menu[k] << endl;

        cin >> option;

        cin.ignore(100, '\n');

        cout << "You selected option " << menu[option - 1] << endl;

        switch (option)

        {

        case 1: {

            cout << "Enter the day 1 - 31." << endl;

            cin >> dayofmonth;

            if (dayofmonth >= 1 && dayofmonth <= 31)

            {

                cout << "The total precipitation for the state of Alabama " << endl

                    << "on March " << dayofmonth << " was "

                    << total\_precip\_day(prcp, daysofmarch, j, dayofmonth)

                    << " inches. " << endl;

            }

            else

            {

                cout << "Error. Day must be between 1 and 31. " << endl;

            }

            system("pause");

            break;

        }

        case 2: {

            cout << "Enter the beginning day 1 - 30." << endl;

            cin >> start\_day;

            if (start\_day >= 1 && start\_day <= 31)

            {

                cout << "Enter the ending day 2 - 30." << endl;

                cin >> end\_day;

                if (end\_day >= 1 && end\_day <= 31)

                {

                    cout << "The total precipitation for the state of Alabama " << endl

                        << "from March " << start\_day << " and " << end\_day << " was "

                        << total\_precip\_range(prcp, daysofmarch, j, dayofmonth, start\_day, end\_day)

                        << " inches. " << endl;

                }

                else

                {

                    cout << "Error. Day must be between 2 and 31. " << endl;

                }

            }

            else

            {

                cout << "Error. Day must be between 1 and 31. " << endl;

            }

            system("pause");

            break;

        }

        case 3: {

            cout << "Enter the station name." << endl;

            getline(cin, station\_name);

            cout << "This is the total Precipitation by station name" << total\_precip\_by\_station(prcp, j, station\_name, dataline, station) << endl;

            // Add code here

            system("pause");

            break;

        }

        case 4: {

            cout << "Enter the station name: ";

            getline(cin, station\_name);

            cout << "The Minimum Temperature is: " << minTemp(tminF, j, station, station\_name) << endl;

            cout << "The Maximum Temperature is: " << maxTemp(tmaxF, j, station, station\_name) << endl;

            cout << "The Average Maximum Temerature is: " << sumOfTemp(tmaxF, j, station, station\_name)/31.0 << endl;

            cout << "The Average Minimum Temerature is: " << sumOfTemp(tminF, j, station, station\_name)/31.0 << endl;

            system("pause");

            break;

        }

        case 5: {

            cout << "Enter the station name." << endl;

            getline(cin, station\_name);

            for (k = 0; k < station\_name.length(); k++)       // Make sure it is uppercase

                station\_name[k] = toupper(station\_name[k]);

            cout << "Enter the beginning day 1 - 30." << endl;

            cin >> start\_day;

            cout << "Enter the ending day 2 - 30." << endl;

            cin >> end\_day;

            cout << "The Minimum Temperature is: " << minTemp(tminF, j, station, station\_name, start\_day, end\_day, daysofmarch) << endl;

            cout << "The Maximum Temperature is: " << maxTemp(tmaxF, j, station, station\_name, start\_day, end\_day, daysofmarch) << endl;

            cout << "The Average Maximum Temerature is: " << avgOfTemp(tmaxF, j, station, station\_name, start\_day, end\_day, daysofmarch)<< endl;

            cout << "The Average Minimum Temerature is: " << avgOfTemp(tminF, j, station, station\_name, start\_day, end\_day, daysofmarch) << endl;

            system("pause");

            break;

        }

        }

        cout << endl << endl;

    }

    cout << endl << endl;

    return 0;

}

float total\_precip\_day(float prcp[], unsigned int days[], int elements, int DOM)

{

    float total\_prcp = 0.0;

    int k = 0;

    for (k = 0; k < elements; k++)

    {

        if (days[k] == DOM)

        {

            total\_prcp += prcp[k];

        }

    }

    return total\_prcp;

}

float total\_precip\_range(float prcp[], unsigned int days[], int elements, int DOM, int start\_day, int end\_day)

{

    float total\_prcp = 0.0;

    int k = 0;

    for (k = 0; k < elements; k++)

    {

        if ((days[k] >= start\_day) && (days[k] <= end\_day))

        {

            total\_prcp += prcp[k];

        }

    }

    return total\_prcp;

}

//Add function for part 3

float total\_precip\_by\_station(float prcp[], int elements, string answer, string dataline, string station[])

{

    string s\_station;

    float total\_prcp = 0.0;

    for(int i = 0; i < elements; i++)

    {

        if(station[i].find(answer) != string::npos)

        {

            total\_prcp += prcp[i];

        }

    }

    return total\_prcp;

}

//Add function for part 4

float minTemp(float temps[], int elements, string stations[], string station)

{

    float min = 9999.0;

    string realName;

    for(int i = 0; i < elements; i++)

    {

        if(stations[i].find(station) != string::npos)

        {

            if(temps[i]< min) min = temps[i];

        }

    }

    return min;

}

float maxTemp(float temps[], int elements, string stations[], string station)

{

    float max = -9999.0;

    for(int i = 0; i < elements; i++)

    {

        if(stations[i].find(station) != string::npos)

        {

            if(temps[i] > max) max = temps[i];

        }

    }

    return max;

}

float sumOfTemp(float temps[], int elements, string stations[], string station)

{

    float sum = 0;

    for(int i = 0; i < elements; i++)

    {

        if(stations[i].find(station) != string::npos)

        {

            sum += temps[i];

        }

    }

    return sum;

}

//Add function for part 5

float minTemp(float temps[], int elements, string stations[], string station, unsigned int startDay, unsigned int endDay, unsigned int days[])

{

    float min = 9999.0;

    string realName;

    for(int i = 0; i < elements; i++)

    {

        if(stations[i].find(station) != string::npos && days[i] >= startDay && days[i] <= endDay)

        {

            if(temps[i]< min) min = temps[i];

        }

    }

    return min;

}

float maxTemp(float temps[], int elements, string stations[], string station, unsigned int startDay, unsigned int endDay, unsigned int days[])

{

    float max = -9999.0;

    for(int i = 0; i < elements; i++)

    {

        if(stations[i].find(station) != string::npos && days[i] >= startDay && days[i] <= endDay)

        {

            if(temps[i] > max) max = temps[i];

        }

    }

    return max;

}

float  avgOfTemp(float temps[], int elements, string stations[], string station, unsigned int startDay, unsigned int endDay, unsigned int days[])

{

    float sum = 0;

    int count = 0;

    for(int i = 0; i < elements; i++)

    {

        if(stations[i].find(station) != string::npos && days[i] >= startDay && days[i] <= endDay)

        {

            sum += temps[i];

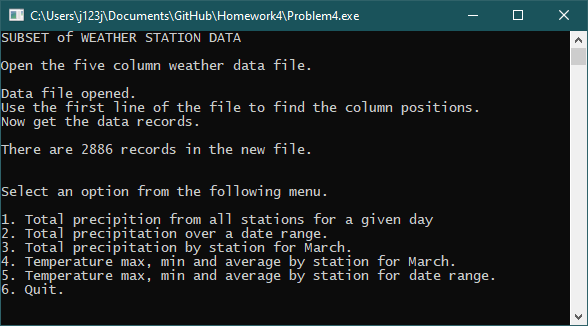
            count++;

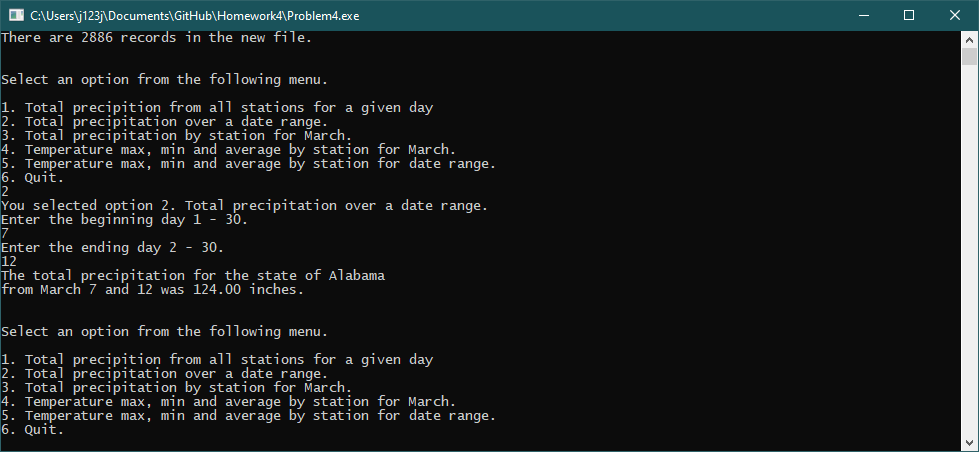
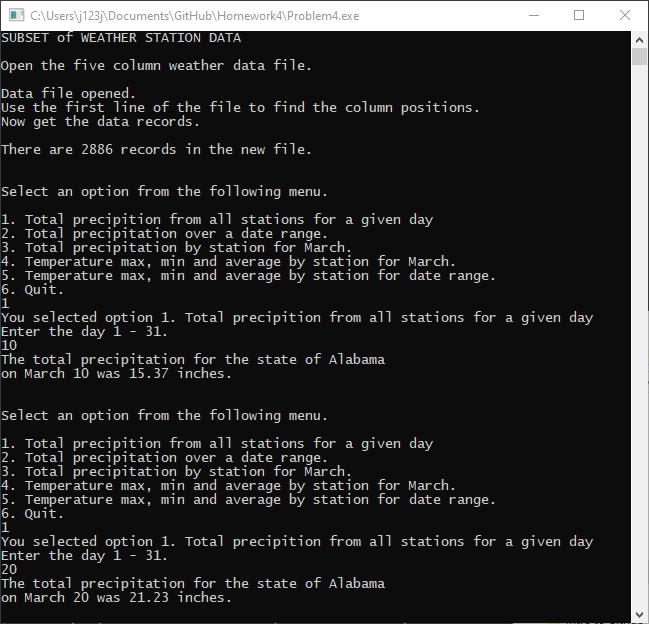
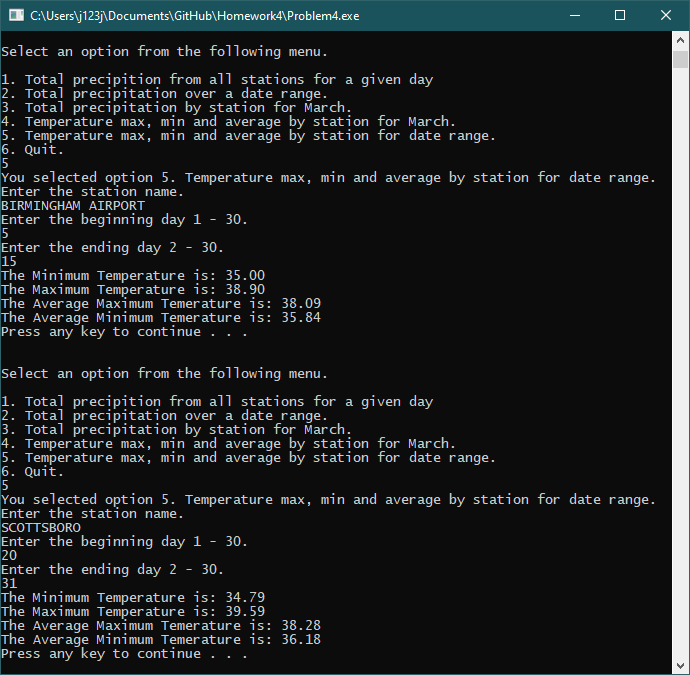
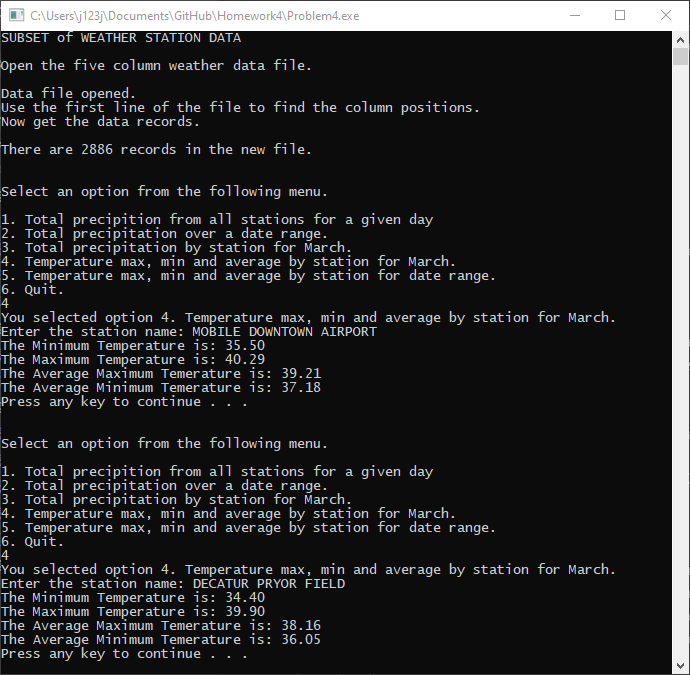
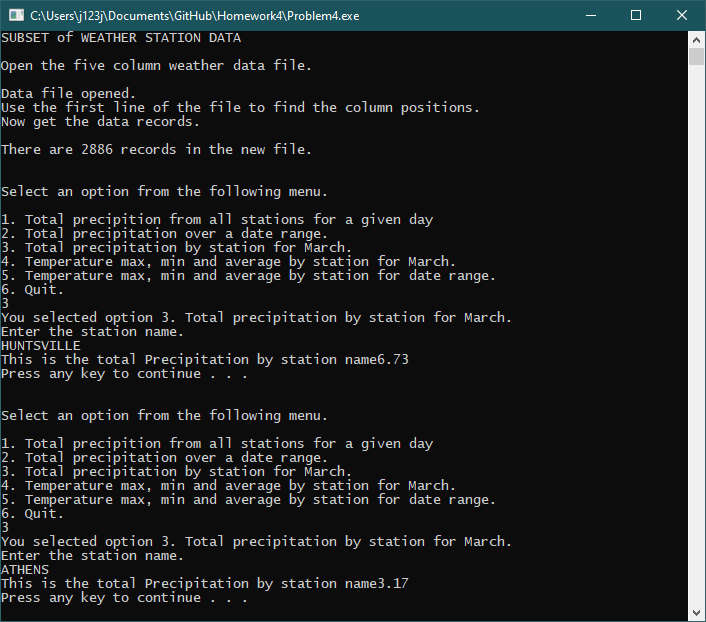
        }

    }

    return sum/count;

}

**Output(s):**

****