Assignment:

The National Weather Service maintains a database of temperatures for major cities. You will find a fictional copy of part of this database for one day in the file: *temperatures.txt* which is part of this assignment. Each line in the file contains a city, the low temperature for the day, and the high temperature for the day.

Step (1)

Your first task is to write a program which reads this file into three parallel arrays in memory. One array contains the cities, and the other two arrays contain the high and low temperatures. The arrays are ‘parallel’ in the sense that the *n-th* element of the cities array contains the city associated with the *n-th* element of the temperature arrays. Although the data in the test file describes only a few cities you should dimension your arrays for an arbitrary number of cities in the live data file (which may be different from the test file). The size of the arrays should be defined in your program by the declaration *const int ARRAY_SIZE = 1000*.

For the purpose of this assignment (so as not to make it too complicated) you should declare your three arrays at *file scope* (i.e. outside any function and before the function main()). The declarations should be as follows:

```
string city[ARRAY_SIZE];

double lowTemp[ARRAY_SIZE];

double highTemp[ARRAY_SIZE];
```

Write a function whose signature is as follows:

```
int LoadData ()
```

The function *LoadData* should prompt the user for the file path to the data file. If it cannot open the file for some reason, it should return a value of -1 as an error result. Otherwise it should read the data in the file into the three parallel arrays, and return the number of city records that it read.

Now write a function whose signature is as follows:

```
void ShowAll (int count)
```

The parameter count provides the number of cities actually stored in the database (i.e the value returned by the function *LoadData()*). *ShowAll()* should read the designated number of records in the parallel arrays and display it on the screen in the order it appears in the database, in single column format. In this display format the low and high temperatures appear in parentheses following the city name, and there is one city record per line of output.

Sample file:

Katmandu, -34, 28

Perth, 92, 105

Chicago, 22, 45

St. Petersburg, 19, 37

Miami, 68, 84

Cincinnati, 21, 39

Bern, 45, 55

Johannesburg, 87, 102

Seattle, 34, 42

New York, 24, 39

St. Louis, 12, 23

Beijing, 45, 66

Munich, 31, 38

Zurich, 33, 41

Buenos Aires, 69, 84

For example, using the data in the sample file, the output would look like this:

Katmandu (-34, 28)

Perth (92, 105)

Chicago (22, 45)

St. Petersburg (19, 37)

…

…

Now test your work so far, by writing a *main()* function to prompt the user for the path to the data file, open and read the file to the parallel arrays using the function *LoadData()* (providing error messages and recovery as appropriate if the file doesn't exist). Then display the contents of the file using *ShowAll()*.

Step (2)

Extend your program into a menu driven application to query the database. When the program starts it should ask the user for the path to the data file, and read the data into the parallel arrays, as in step 1.

Then it should enter a loop, in each iteration of which it prompts the user for what they want to do. The actions are as follows depending on how the user responds to the prompt. See the sample dialog below to see how this works in practice.

- Q or q: Quit. The program terminates.
- S or s: The program displays all the records in the database using the function *ShowAll* you have already developed in step (1).
- F or f: Find the specified city(s). The program displays all records which contain the designated text **somewhere** in the city field, in a **case insensitive** manner (upper or lower case should both work). You should use a function with the following signature for this purpose: *int ShowCities(int count, string name).* This function should display all records whose city field contains the string passed in the parameter called *name*. It returns the number of records displayed. *count* specifies the number of records in the database.
  Hint: Making the search case insensitive adds some complexity to this assignment. Consider converting a copy of each of the strings to either uppercase or lowercase before doing the search. The string.find() function can be used.

Sample Dialog

Note: In the example below, **the user's input is in red BOLD.** The program output is not.

```
Welcome to John's Library Database.

(L)oad File, (S)how All, (F)ind City, (Q)uit: L

Please enter the name of the temperatures file: C:\temperatures.txt

15 record(s) found.

(L)oad File, (S)how All, (F)ind City, (Q)uit: F

City: C

    Chicago (22, 45)

    Cincinnati (21, 39)

    Munich (31, 38)

    Zurich (33, 41)

4 record(s) found.

(L)oad File, (S)how All, (F)ind City, (Q)uit: F

City: Katman

    Katmandu (-34, 28)

1 record(s) found.

(L)oad File, (S)how All, (F)ind City, (Q)uit: Q
```

Design Considerations

1) Please display your name and not mine in the welcome message.

2) The following two lines seem to be required to but the file back to a state where it can be opened again it (otherwise, some compilers, like mine, will not allow the file to be reopened):
   inFile.close();

inFile.clear(std::ios_base::goodbit);

3) It is an essential part of your grade that you design your application according to the considerations described above. For example, you must use functions that have the specified signatures, and arrays that have the specified declarations. They are as follows:

// These declarations should be at file scope

const int ARRAY_SIZE = 1000;

string city[ARRAY_SIZE];

double lowTemp[ARRAY_SIZE];

double highTemp[ARRAY_SIZE];

// Function prototypes

int LoadData();

void ShowAll(int count);

int ShowCities(int count, string name);

Solution:

```
#include <iostream>
#include <fstream>
#include <string>

using namespace std;

const int ARRAY_SIZE = 1000;
const int STR_BUFFER_SIZE = 100;
string city[ARRAY_SIZE];
double lowTemp[ARRAY_SIZE];
double highTemp[ARRAY_SIZE];

// Function prototypes
int LoadData();
void ShowAll(int count);
int ShowCities(int count, string name);

int main()
{
    int numOfRecords = 0;
    char selector = 'q', yesNoAnswer = 'n';
    string name = "";

    cout<<"Welcome to Abdul’s Library Database.\n";

    do // menu loop
    {
        cout<<"\n(L)oad File,\t(S)how All,\t(F)ind City,\t(Q)uit: ";
        cin>>selector;
        selector = toupper(selector);
        switch(selector)
        {
        case 'L':
            if (numOfRecords > 0)
            {
                cout<<"Are you sure you want to delete previous data? (Y/N): ";
                cin>>yesNoAnswer;
                if (yesNoAnswer == 'Y' || yesNoAnswer == 'y')
                    numOfRecords = 0;
                else
                    break;
            }
            numOfRecords = LoadData();
            if (numOfRecords == -1)
                cout<<"Cannot open the input file!\n";
            else
                cout<<numOfRecords<<" record(s) found.\n";
            break;
        case 'S':
            if (numOfRecords <= 0)
                cout<<"No records found!\n";
            else
                ShowAll(numOfRecords);
            break;
        case 'F':
            cout<<"City: ";
            cin>>name;
            if (numOfRecords <= 0)
                cout<<"No records found!\n";
            else
                cout<<ShowCities(numOfRecords, name)<<" record(s) found.\n";
            break;
        }
    }
    while (selector != 'q' && selector != 'Q');

    return 0;
}

int LoadData()
{
    int numOfRecords = 0;
    string filename = "";
    char c, tmp[STR_BUFFER_SIZE];
    ifstream inFile;

    cout<<"Please enter the name of the temperatures file: ";
    cin>>filename;
    inFile.open(filename.c_str());
    if (!inFile) // cannot open the file
        return -1;
    while (!inFile.eof()) // reading data from file
    {
        inFile.getline(tmp, STR_BUFFER_SIZE, ',');
        if (numOfRecords == 0)
            city[numOfRecords].assign(tmp);
        else
            city[numOfRecords].assign(&tmp[1]); // the first character is
newline symbol
        inFile>>lowTemp[numOfRecords];
        inFile>>c; // read coma after the low temperature
        inFile>>highTemp[numOfRecords];
        ++numOfRecords;
    }
    --numOfRecords; // the last line is empty
    inFile.close();
    inFile.clear(std::ios_base::goodbit);

    return numOfRecords;
}

void ShowAll(int count)
{
    int i;

    for (i = 0; i < count; ++i)
        cout<<city[i]<<" ("<<lowTemp[i]<<", "<<highTemp[i]<<")\n";
}

int ShowCities(int count, string name)
{
    int numOfRecords = 0, i;
    unsigned int j;
    string upperString; // stores a lowercase copy of the string

    for (i = 0; i < count; ++i)
    {
        upperString = "";
        for (j = 0; j < city[i].length(); ++j)
            upperString += tolower(city[i][j]);
        if (upperString.find(name) != string::npos) // the name is a substring
of the upperString
        {
            cout<<city[i]<<" ("<<lowTemp[i]<<", "<<highTemp[i]<<")\n";
            ++numOfRecords;
        }
    }

    return numOfRecords;
}
```