CS205 C/C++ Programming Lab Assignment 3

Name: 钟兆玮 (Zhaowei Zhong)
SID: 11611722

Part 1 - Analysis

You will in this lab build on what was done in Lab 1 (computation of distances) and write a program that is easier to use. This program isn't very simple if written well and you'll have several weeks to write it.
As entering latitude and longitude is boring, you are provided with a text file that contains this information for many cities in the world (You can add your hometown if you wish). This file is named world_cities.csv and contains one line per city with the following information:

- City name
- Province or state (may be absent)
- Country
- Latitude
- Longitude

These values are separated by commas (.csv means "Comma Separated Values", it's a commonly used format for exchanging data). When a piece of data is missing (province or state), you get two commas in succession, for instance (first row in the file):

```
Aarhus,,Denmark,56.150,10.217
```

The cities are in alphabetical order.

Part 1

The cities are in alphabetical order.
We aren't in this lab interested in the "province or state" information. You are asked to define a structure to hold city and country name, latitude and longitude.
You must create an array containing structures as you have defined them, read the file, and load data into the array.
1. Initially set the maximum length for names (city and country name) to 25, and the array size to 800. Your program should issue a warning when data is truncated or not loaded, but it mustn't crash.
2. Set the maximum length for names to 35, and the array size to 1000. Check that your program runs without any problem.
3. Rename the file to world_cities.tmp. Run your program. It mustn't crash and should display a warning about the missing file.
4. Rename the file to world_cities.csv and continue with part 2.

Part 2

Once all data is loaded, your program should enter a loop from which it will exit when the user types "bye" (case-insensitive). For each iteration, it will prompt for the names of two cities.
The program must search the cities in the table by their names and retrieve the latitudes and longitudes. If the city isn't found or if the length of the name is shorter than three letters, a message must be displayed and the user must be prompted for another name. The city names should be case-insensitive.
Please note that in the file New York appears as "New York City". If people type "New York", then "New York City" must be retrieved. However, if users only type "New" (minimum acceptable length), it can match several cities. The list of the matched cities must be displayed, and the user prompted for the right one.
Finally, the names of the cities (as stored in the memory) and the distance between them must be displayed.
As before, your program should ignore all whitespaces on both ends of user inputs.

Part 2 - Code

```
#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <math.h>
#include <algorithm>

#define MAX_NAME_LENGTH 35
#define ARRAY_SIZE 800

struct City {
    std::string name;
    std::string country;
    float latitude;
    float longitude;
};

std::vector<std::string> split(std::string str,std::string pattern);
double get_distance(City city1, City city2);

int main() {
    City data[ARRAY_SIZE];
    // Part 1: Load Data
    std::ifstream infile;
    std::string line;
    infile.exceptions( std::ifstream::failbit | std::ifstream::badbit );
    try {
        infile.open("world_cities.csv");
        int i = 0;
        while (!infile.eof()) {
```

```
            getline(infile, line);
            if (i >= ARRAY_SIZE) {
                std::cout << "[Warning] Data is too large and will be truncated." <
< std::endl;
                break;
            }
            std::cout << line << std::endl;
            std::vector<std::string> result = split(line, ",");
            data[i] = City {
                result[0],
                result[2],
                stof(result[3]),
                stof(result[4])
            };
            i++;
        }
    }
    catch (const std::ifstream::failure& e) {
        std::cout << "Data file is missing." << std::endl;
        exit(-1);
    }
    // Part 2: Calculate Distance
    while (true) {
        std::cout << "Please enter two city names (\"bye\" to exit)." << std::endl;
        // City #1
        City city1;
        std::cout << "City #1: ";
        std::string city1_input;
        getline(std::cin, city1_input);
        city1_input.erase(0, city1_input.find_first_not_of(" "));
        city1_input.erase(city1_input.find_last_not_of(" ") + 1);
        std::transform(city1_input.begin(), city1_input.end(), city1_input.begin(),
 ::tolower);
        if (city1_input == "bye") {
            std::cout << "Bye!" << std::endl;
            exit(0);
        }
        int city1_result[ARRAY_SIZE];
        int city1_total = 0;
        for (int i = 0; i < ARRAY_SIZE; i++) {
            std::string city_name = data[i].name;
            std::transform(city_name.begin(), city_name.end(), city_name.begin(), :
:tolower);
            std::string::size_type idx = city_name.find(city1_input);
            if (idx != std::string::npos ) {
                city1_result[city1_total] = i;
                city1_total++;
            }
        }
```

```
        if (city1_total == 0) {
            std::cout << "No city matches your input." << std::endl;
            continue;
        } else if (city1_total == 1) {
            std::cout << "Matched: " << data[city1_result[0]].name << std::endl;
            city1 = data[city1_result[0]];
        } else {
            std::cout << "There's more than one cities matched your input: " << std
::endl;
            for (int i = 0; i < city1_total; i++) {
                std::cout << i + 1 << ". " << data[city1_result[i]].name << std::en
dl;
            }
            std::cout << "Please select one: ";
            int selection;
            while (true) {
                std::cin >> selection;
                if (selection < 1 || selection > city1_total) {
                    std::cout << "Wrong input, please input again: ";
                } else {
                    break;
                }
            }
            city1 = data[city1_result[selection - 1]];
        }
        std::cin.sync();
        std::cin.clear();
        // City #2
        City city2;
        std::cout << "City #2: ";
        std::string city2_input;
        getline(std::cin, city2_input);
        city2_input.erase(0, city2_input.find_first_not_of(" "));
        city2_input.erase(city2_input.find_last_not_of(" ") + 1);
        std::transform(city2_input.begin(), city2_input.end(), city2_input.begin(),
 ::tolower);
        if (city2_input == "bye") {
            std::cout << "Bye!" << std::endl;
            exit(0);
        }
        int city2_result[ARRAY_SIZE];
        int city2_total = 0;
        for (int i = 0; i < ARRAY_SIZE; i++) {
            std::string city_name = data[i].name;
            std::transform(city_name.begin(), city_name.end(), city_name.begin(), :
:tolower);
            std::string::size_type idx = city_name.find(city2_input);
            if (idx != std::string::npos ) {
                city2_result[city2_total] = i;
```

```
                city2_total++;
            }
        }
        if (city2_total == 0) {
            std::cout << "No city matches your input." << std::endl;
            continue;
        } else if (city2_total == 1) {
            std::cout << "Matched: " << data[city2_result[0]].name << std::endl;
            city2 = data[city2_result[0]];
        } else {
            std::cout << "There's more than one cities matched your input: " << std
::endl;
            for (int i = 0; i < city2_total; i++) {
                std::cout << i + 1 << ". " << data[city2_result[i]].name << std::en
dl;
            }
            std::cout << "Please select one: ";
            int selection;
            while (true) {
                std::cin >> selection;
                if (selection < 1 || selection > city2_total) {
                    std::cout << "Wrong input, please input again: ";
                } else {
                    break;
                }
            }
            city2 = data[city2_result[selection - 1]];
        }
        // Calculation
        std::cout << "The distance between " << city1.name << " and " << city2.name
 << " is " << get_distance(city1, city2) << "km." << std::endl << std::endl;
        std::cin.sync();
        std::cin.clear();
    }
}

std::vector<std::string> split(std::string str, std::string pattern) {
    std::string::size_type pos;
    std::vector<std::string> result;
    str += pattern;
    int size = str.size();
    for (int i = 0; i < size; i++) {
        pos = str.find(pattern, i);
        if (pos < size) {
            std::string s = str.substr(i, pos - i);
            result.push_back(s);
            i = pos + pattern.size() - 1;
        }
    }
```

```
    return result;
}

double get_distance(City city1, City city2) {
    double radLat1 = city1.latitude * 3.1415926 / 180.0;
    double radLat2 = city2.latitude * 3.1415926 / 180.0;
    double a = radLat1 - radLat2;
    double b = city1.longitude * 3.1415926 / 180.0 - city2.longitude * 3.1415926 /
180.0;
    double dst = 2 * asin((sqrt(pow(sin(a / 2), 2) + cos(radLat1) * cos(radLat2) *
pow(sin(b / 2), 2) )));
    dst = dst * 6378.137;
    dst = round(dst * 10000) / 10000;
    return dst;
}
```

Part 3 - Result & Verification

Test Case #1: Initially set the maximum length for names (city and country name) to 25, and the array size to 800.
Input: shenzhen / beijing

```
~/Courses/CPP/assignment3  master  ./cities
[Warning] Data is too large and will be truncated.
Please enter two city names ("bye" to exit).
City #1: shenzhen
Matched: Shenzhen
City #2: beijing
Matched: Beijing
The distance between Shenzhen and Beijing is 1943.56km.

Please enter two city names ("bye" to exit).
City #1:
```

Test Case #2: Set the maximum length for names to 35, and the array size to 1000.
Input: shenzhen / beijing

```
~/Courses/CPP/assignment3  master  g++ cities.cpp -o cities
~/Courses/CPP/assignment3  master  ./cities
Please enter two city names ("bye" to exit).
City #1: shenzhen
Matched: Shenzhen
City #2: beijing
Matched: Beijing
The distance between Shenzhen and Beijing is 1943.56km.

Please enter two city names ("bye" to exit).
City #1:
```

Test Case #3: Rename the file to world_cities.tmp. Run your program.

```
~/Courses/CPP/assignment3  master  ./cities
Data file is missing.
~/Courses/CPP/assignment3  master
```

Test Case #4: Normal Case

Input: New York / New

```
~/Courses/CPP/assignment3  master  ./cities
Please enter two city names ("bye" to exit).
City #1: New York
Matched: New York City
City #2: New
There's more than one cities matched your input:
1. New Delhi
2. New Orleans
3. New York City
4. Newcastle upon Tyne
5. Newcastle
Please select one: 3
The distance between New York City and New York City is 0km.

Please enter two city names ("bye" to exit).
City #1:
```

Test Case #4: Normal Case

Input: new york / shenzhen

```
~/Courses/CPP/assignment3  master  ./cities
Please enter two city names ("bye" to exit).
City #1: new york
Matched: New York City
City #2: shenzhen
Matched: Shenzhen
The distance between New York City and Shenzhen is 12951.2km.

Please enter two city names ("bye" to exit).
City #1:
```

Test Case #4: Exit

Input: bye

```
~/Courses/CPP/assignment3  master  ./cities
Please enter two city names ("bye" to exit).
City #1: bye
Bye!
~/Courses/CPP/assignment3  master
```

Test Case #4: Exit

Input: bYe

```
~/Courses/CPP/assignment3  master  ./cities
Please enter two city names ("bye" to exit).
City #1: bYe
Bye!
~/Courses/CPP/assignment3  master
```

Test Case #4: Exit

Input: BYE

```
~/Courses/CPP/assignment3  master  ./cities
Please enter two city names ("bye" to exit).
City #1: BYE
Bye!
~/Courses/CPP/assignment3  master
```

Part 4 - Difficulties & Solutions

If the input file was missing, it would cause an error. I use `try-catch` statements to handle and process the exceptions.