Lab 5: Data Files

Your name:	Zach Evans
Your email address:	zqevans@gmail.com
Your student ID number:	W01006113

Lab Objectives

To gain experience with

- reading and writing files
- the command line
- the concepts of sequential access

For this lab, use the two ziped sets of data files on the Moodle site.

P1. Reading Text Files (20 points)

You already know how to read data from a file and write data to a file, by using the system variables `cin` and `cout` together with command-line redirection (`myprog <input.txt >output.txt`).

However, this approach is limited. You need to learn to use files directly in your program in the following situations:

- to read input from more than one file
- to read and update a single file
- to prompt the user for the name of a file

In these situations, you need files. C++ supports three file types, `fstream`, `ifstream` and `ofstream`. The latter two are used to open files for input or output only. Since `fstream` can be used for both reading and writing, we will not cover those other file classes in this course.

To open a file, first define an object of type `fstream` and then call the `open` member function.

```
ofstream my_data;
my_data.open("output.txt");
```

Next, you should check that the file was opened successfully:

```
if (my_data.fail())
   cout << "Unable to open output.txt";
```

Then you can read and write with the same commands that you used with `cin` and `cout`, such as `<< >> getline` etc.

Write a program which compares the contents of two files. Have your program display the line number and text of the first pair of lines that differ.

First, prompt the user for the names of the two files (2 points). Then open each file (2 points). Remember to check for failure (2 points). Keep reading a line from each of them. Increment a line number counter (2 points). If both inputs fail, print a message indicating that both files are identical (2 points). If one of the inputs fails and the other does not, print a message indicating which file is shorter, then exit (2 points). If the two input lines are different, print the line number counter and both input lines, then exit (2 points). If the two input lines are identical, keep on reading (2 points). Plus 4 points for overall structure and style.

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

using namespace std;

int main()
{
    fstream fileOne;
    fstream fileTwo;
    string fileOneName;
    string fileTwoName;
    string fileOneLine;
    string fileTwoLine;

    cout << "Please enter file one: ";
    cin >> fileOneName;

    cout << "Please enter file two: ";
    cin >> fileTwoName;

    fileOne.open(fileOneName.c_str());
    fileTwo.open(fileTwoName.c_str());

    if (fileOne.fail())
    {
        cout << "Unable to open" << fileOneName << "\n";
    }

    if (fileTwo.fail())
    {
        cout << "Unable to open" << fileTwoName << "\n";
    }

    int lineNumber = 1;
    while (!fileOne.fail() && !fileTwo.fail())
    {

        getline(fileOne, fileOneLine);
        getline(fileTwo, fileTwoLine);

        if (fileOne.fail() && !fileTwo.fail())
        {
            cout << "File 1 is shorter.\n";
            break;
        }

        if (fileTwo.fail() && !fileOne.fail())
        {
            cout << "File 2 is shorter.\n";
            break;
        }

        if (fileOne.fail() && fileTwo.fail())
        {
            cout << "Files are identical.\n";
        }

        if (fileOneLine != fileTwoLine)
        {
            cout << "Line number: " << lineNumber << "\n";
            cout << "File 1: " << fileOneLine << "\n";
            cout << "File 2: " << fileTwoLine << "\n";
            break;
        }
        lineNumber++;
    }
    return 0;
}
```

P4. Command Line Arguments (20 points)

Extend your file comparison program in two ways. First, make it possible to specify the file names on the command line (5 points). Only ask the user for the file names when they are not specified (5 points). And support a command line flag to control the number of differences identified (5 points). Plus 5 points for overall structure and style. Usage should look like this:

```
differ -a file1.txt file2.txt
   displays all differences
differ -n  file1.txt file2.txt
   displays the first n differences
   And, for example
differ -20
   displays the first 20 differences and prompts the user for the file names
```

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

using namespace std;

int string_to_int(string s)
 {
        istringstream instr(s);
        int n;
        instr >> n;
        return n;
 }



int main(int argc, const char * argv[])
{
    fstream fileOne;
    fstream fileTwo;
    string fileOneName;
    string fileTwoName;
    string fileOneLine;
    string fileTwoLine;
    int numDifs;
    int difCount = 0;
    bool allDifs = false;



    for (int i = 1; i < argc; i++)
    {
        cout << argv[i] << "\n";

    }

    if (argc >= 1)
    {
        string arg = string(argv[1]);
        if (arg.length() >= 2 && arg[0] == '-')
        {
            if (arg[1] == 'a')
            {
                allDifs = true;
            }
            else
            {
                numDifs = string_to_int(arg.substr(1, arg.length() - 1));
            }
        }

        if (argc >= 3) //Assuming three arguments: the flag, and two file names
        {
            fileOneName = string(argv[2]);
            fileTwoName = string(argv[3]);
        }

    }

    if (argc == 2) //Only argument is the flag
    {
    cout << "Please enter file one: ";
    cin >> fileOneName;

    cout << "Please enter file two: ";
    cin >> fileTwoName;
    }

    fileOne.open(fileOneName.c_str());
    fileTwo.open(fileTwoName.c_str());

    if (fileOne.fail())
    {
        cout << "Unable to open" << fileOneName << "\n";
```

Don't forget to upload your lab when you're finished.