Student Lab Activity

A. Lab # CIS CIS170C-A7

B. Lab 7 of 7: Sequential Files

C. Lab Overview - Scenario/Summary

You will code, build, and execute a program that requires sequential files to create an address database.

Learning Outcomes

1. Continue using a menu system with console applications
2. Be able to write a console application
3. Demonstrate entering, appending, storing, and retrieving records
4. Be able to write lines of output to a text file in order to create a report

D. Deliverables

Section	Deliverable	Points
Step	Program Listing and Output	**45**

E. Lab Steps

Preparation:

If you are using the Citrix remote lab, follow the login instructions located in the iLab tab in Course Home.

Locate the Visual Studio 2010 icon and launch the application.

Lab:

Step 1: Requirements: An Address Database
Create a C++ console application that will store and retrieve names and addresses in a text file. The program should do the following. 1. It should accept a series of names and addresses from the console. 2. The user's input should be written to a text file in the CSV format described in the lecture, but do not include the field names in the first row of the file.

3. Read the records from the text file, and display them in a user-friendly format.
4. Provide a menu to allow the user to append records to the file, display the records, or exit the application.

Build upon the code below to complete the assignment.

```
//Specification: Append and display records in a address database

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

using namespace std;

void menu(void);
void writeData(void);
void readData(void);
string * split(string, char);

const char FileName[] = "TestAddress.txt";

int main () {
	menu();
	return 0;
} //end main

void menu(void) {
//allow user to choose to append records, display records or exit the program

}//end menu

void writeData(void){
//Write the Address Info to a file

}//end write data

void readData(void){
//read data from a file
//use the split function to break a
//deliminated line of text into fields

}//end read data

string * split(string theLine, char theDeliminator){
	//Break theline into fields and save the fields to an array.
	//Each field will occupy one element in a character array.
	//theLine is a string with fields separated with theDeliminator character.
	//Assumes the last field in the string is terminated with a newline.
	//Useage: string *theFields = split(lineBuffer, ',');

	//determine how many splits there will be so we can size our array
	int splitCount = 0;
	for(int i = 0; i < theLine.size(); i++){
		if (theLine[i] == theDeliminator)
			splitCount++;
	}
	splitCount++; //add one more to the count because there is not an
ending comma

	//create an array to hold the fields
	string* theFieldArray;
	theFieldArray = new string[splitCount];

	//split the string into seperate fields
	string theField = "";
	int commaCount = 0;

	for(int i = 0; i < theLine.size(); i++){ //read each character and look for
the deliminator
		if (theLine[i] != theDeliminator) {
			theField += theLine[i]; //build the field
		}
		else { //the deliminator was hit so save to the field to the array
			theFieldArray[commaCount] = theField; //save the field to the
array
			theField = "";
			commaCount++;
		}
	}
	theFieldArray[commaCount] = theField; //the last field is not marked
with a comma...

	return theFieldArray;
} //end split
```

Step 2: Processing Logic

Using the pseudocode below, write the code that will meet the requirements.

The pseudocode for the writeData function is shown below.

```
Start

   open the text file to append

   start do while loop

         Allow user to enter name

         store name (using getline method)

         Allow user to enter city
```

```
                store city (using getline method)

                .

                .

                write name, city, etc. to the file

     end loop

     close the file

End
```

The program input should appear similar to this.

```
Append Records

Name..........John Smith
Street.........902 Union Ave
City............Any Town
State...........TX
Zip Code......78552

"Enter another Record? (Y/N) "
```

The file structure should look like this.

```
John Smith, 902 Union Ave, Any Town, TX,
79552
Eric Jones, 345 State Way, Fresno, CA,
93432
...
```

The file output should appear similar to the following.

```
Show Records
________________________________________
Record #1
Name...........John Smith
Street..........902 Union Ave
City.............Any Town
State...........TX
Zip Code......78552
________________________________________
Record #2
Name...........Eric Jones
Street..........345 State Way
City.............Fresno
State...........CA
Zip Code.......93432
```

```
________________________________________________

(A)ppend Records, (S)how Records, (E)xit
```

Step 3: Create a New Project

Create a new project and name it LAB7. Write your code using the processing logic in Step 2. Make sure you save your program.

Step 4: Compile and Execute

a) Compile your program. Eliminate all the syntax errors.

b) Build your program and verify the results of the program. Make corrections to the program logic, if necessary, until the results of the program execution are what you expect.

Step 5: Print Screenshots and Program

1. Capture a screen print of your output. (Do a print screen and paste into an MS Word document.)
2. Copy your code and paste it into the same MS Word document that contains the screen print of your output.
3. Save the Word document as Lab07_LastName_FirstInitial.

END OF LAB