CSC 2430 Spring 2019

LAB 3 – Call Analysis II

Due Tuesday, May 21 *5pm*

Goal: In this assignment you will rework Lab 2 using C++ classes.

First, You will define two classes in your program. This time, you **SHOULD NOT** add your custom variables/functions to the classes, and the following class declaration is strictly enforced.

CALL: This class will:

1) Include the following **public** methods. All of these should be declared **const**, meaning that the function won't make any changes on the class' member variables (see zyBooks 11.4 for more details). It's fair to say so, because these are data 'accessor's. Make sure to keep these const keywords – points will be taken off otherwise.

```
// Formatted Getters. Returning formatted strings, not raw strings.
string getFormattedStart() const;
string getCountryCode() const;
string getLocalPhone() const;
string getFormattedDuration() const;
// Phone Number Prefix Checker. Return true if call's phone# starts with the parameter string.
bool phoneNumStartsWith(string prefix) const;
```

2) Also include **public** "setter" methods for setting the start, phone number, and duration as strings. Your setters should check if the data being set is **valid** before setting the value, and **return true if** the data being set is valid (use your validation functions from Lab 2), **false** otherwise, without setting the value.

```
bool setStart(string in);
bool setPhoneNum(string in);
bool setDuration(string in);
```

3) Include the following **private** member variables.

```
string start;
string phonenum;
string duration;
```

CALLDB: This class will:

1) Include the following **private** member variables

```
static const int MAXCALLS = 15;  // maximum number of calls
CALL callLog[MAXCALLS];         // Stores calls in database
unsigned int numCalls;          // Number of calls stored
```

2) Have a constructor taking no parameters. Should initialize the # of records to 0.

3) Include the following **public** methods

```
// Load records from file into database. Return # of records that can't be added
unsigned int load(istream& fin);

// Get count of call records
unsigned int getCountCalls() const;

// Retrieve call record
bool getCall(unsigned int index, CALL& call) const;

// Return first index >= of call w/ matching E164 prefix, or -1
int findByPrefix(unsigned int startIndex, string prefix) const;
```

Your job is to

1) Write “calldb2.h” that defines these two classes
2) Implement these classes in “calldb2.cpp.”
3) **[Milestone 1]** Check your “calldb2.h” and “calldb2.cpp” files with the provided “lab3unittest.cpp”. Again, do NOT use zyBooks autograder (part 1) to debug. You should pass the unit test before submitting.
4) **Submit** your finished and unit tested “calldb2.h” and “calldb2.cpp” to the Autograder Part 1.
5) **[Milestone 2]** Write a main program called “callanalysis2.cpp” that has the same functionality as the main program in Lab 2. HINT: start with the previous lab’s “callanalysis.cpp” and make changes.
6) **Submit** your all three files “calldb2.h”, “calldb2.cpp”, and “callanalysis2.cpp” to the Autograder Part 2.
    a. **Note: As your program is expected to behave exactly the same to the previous lab’s, you will be able to get points by just renaming and submitting the last lab’s codes. However, you will get ZERO point if you do so. Your code will be reviewed if it is following the above requirements.**
7) Make sure your last submission to the Autograder Part 2 follow the appropriate style.

Program Style
Please refer to previous Lab assignment documents for appropriate style.

Grading
Correctness is essential. Make sure your solution builds as described above, correctly processes supplied input files, and passes the unit tests. We will test on other input file(s) as well.

Even if your solution operates correctly, points will be taken off for:
- Not following the design described above
- Not adhering to style guidelines described above
- Using techniques not presented in class
- Programming error not caught by other testing

Academic Integrity
This programming assignment is to be done on an individual basis. At the same time, it is understood that learning from your peers is valid and you are encouraged to talk among yourselves about programming in general and current assignments in particular. Keep in mind, however, that each individual student must do the work in order to learn. Hence, the following guidelines are established:
- Feel free to discuss any and all programming assignments but do not allow other students to look at or copy your code. Do not give any student an electronic or printed copy of any program you write for this class.
- Gaining the ability to properly analyze common programming errors is an important experience. Do not deprive a fellow student of his/her opportunity to practice problem solving: control the urge to show them what to do by writing the code for them.
- If you’ve given the assignment a fair effort and still need help, see the instructor or a lab assistant.
- **If there is any evidence that a program or other written assignment was copied from another student, neither student will receive any credit for it. This rule will be enforced.**
- Protect yourself: Handle throw-away program listings carefully.