Introduction to Systems Programming (System I)

Homework #4

Max Points: 35

Due: Friday, October 30, 2020 before 11:59 PM

Email-based help Cutoff: 5:00 PM on Friday, October 30, 2020

You should save/rename this document using the naming convention MUid.docx (example: `vendomcg.docx`).

Objective: The objective of this exercise is to gain experience with writing an application that interfaces with a database using C++

Fill in answers to all of the questions. You may discuss the questions with your neighbor or instructor. **Yes, you may verify your solution with reference solution available on Canvas.**

Name:

Note: This assignment builds upon Lab #7. In this assignment, you will need to implement more features to improve the functionality of the program.

Programming in C++ with SQL

[35 points]

Background: As we discussed in class, we can create a program just like the mysql command-line program that we used in the previous lab. In this lab, we will be making a light-weight interactive program to interface with a MySQL RDMS.

The program will need to do the following:

1. Take an argument from the command-line to specify interactive mode (-I), load (-L) a file into the database, write (-W) the results of a query/queries to a file, drop (-D) a table, create (-C) a table based on some file. **[1 point]**
    a. Interactive mode will operate like the mysql program where it reads from standard input to get queries from a user.
    b. Loading a file takes an additional argument of the file (so it will be: -L filename) to parse and load into the database.
    c. Writing a file takes two additional arguments of the input and output file (so it will be: -W inputFile outputFile)
    d. Dropping a table takes an additional argument of the table name.
    e. Creating a table takes an additional argument of the file specifying the table that the program will create.

2. The program will need to use the argument to determine its execution path. **[1 point]**
    a. The program will need to connect to the database. **(INFO ON NEXT PAGE)** [**1 point]**
3. Interactive mode (-I): **[5 points]**
    a. Continue to run unless the user enters "quit" or reaches logical end-of-file
    b. Take a user input and make it a query
    c. Execute the query
    d. Access and format the results
    e. Display the results to the user **(EXAMPLE ON NEXT PAGE)**
4. Load a file (-L): **[4 points]**
    a. Parse the supplied file.
        i. Note: it can be safely assumed any file give will be structured the same, but values can be different.
        ii. Note: You should change the first column (muid_tmp) to include your actual muid and not "muid"
    b. Generate a query to insert the file's contents into the database
    c. Execute the query **(EXAMPLE ON NEXT PAGE)**
5. Create a table (-C): **[6 points]**
    a. Parse the supplied file
        i. Note: it can be safely assumed any file give will be structured the same, but values can be different (first column is table then columns formatted as attribute:value for some arbitrary number of columns).
    b. Generate a query to insert the file's contents into the database
    c. Execute the query (Note: It should print "Table muid_myTable Created" after execution and muid should be your actual muid and not "muid")
6. Drop a table (-D): **[6 points]**
    a. Generate a query to drop a table specified based on the command-line argument after -D (e.g., **./hw5 -D tmpTable**)
    b. Execute the query (Note: It should print "Table muid_myTable Dropped" after execution and muid should be your actual muid and not "muid")
7. Write data to file (-W): **[10 points]**
    a. Read each line from the supplied file (list of queries)
        i. Note: this file will include several queries and should execute ***all*** queries
    b. Build a query from the line
    c. Execute the query
    d. Write the results of the query to an output file (format should resemble #3.e)
        i. Do **NOT** write empty results
8. Exception handling: **[1 point]**
    a. Use a similar approach as seen in the lab sample code to appropriate capture the parse exception

EXTRA CREDIT: Create a manual page for your application based on the condensed version of the ls manual page (on the last page of this assignment). **[3 points]**
NOTE: Compilation is the same as Lab8
NOTE: consider how many arguments you will need for *each* feature as they vary.

Database Information (same as Lab 8):

Virtual Machine
Server: 127.0.0.1
User: cse278
Password: S3rul3z
Database Name: cse278

Example Query Result:

```
select * from Product;
-----Query Result-----
| 1 | Product1 | 50 | 1 | Canon |
| 2 | Product2 | 150 | 2 | GizmoWorks |
| 3 | Product3 | 150 | 2 | Hitachi |
------End Result------
```

Example File Load Result:

```
cvendome@Dagobah:solutions$./lab7 -L test_input.csv
Data Line 1 Loaded
Data Line 2 Loaded
Data Line 3 Loaded
```

Condensed ls manual page:

```
LS(1)                    BSD General Commands Manual                    LS(1)

NAME
     ls -- list directory contents

SYNOPSIS
     ls [-ABCFGHLOPRSTUW@abcdefghiklmnopqrstuwx1] [file ...]

DESCRIPTION
     For each operand that names a file of a type other than directory, ls
     displays its name as well as any requested, associated information.  For
     each operand that names a file of type directory, ls displays the names
     of files contained within that directory, as well as any requested, asso-
     ciated information.

     If no operands are given, the contents of the current directory are dis-
     played.  If more than one operand is given, non-directory operands are
     displayed first; directory and non-directory operands are sorted sepa-
     rately and in lexicographical order.

     The following options are available:

     -@      Display extended attribute keys and sizes in long (-l) output.

     -1      (The numeric digit ``one''.)  Force output to be one entry per
             line.  This is the default when output is not to a terminal.

     -A      List all entries except for . and ...  Always set for the super-
             user.

     -a      Include directory entries whose names begin with a dot (.).

     -B      Force printing of non-printable characters (as defined by
             ctype(3) and current locale settings) in file names as \xxx,
             where xxx is the numeric value of the character in octal.

     -b      As -B, but use C escape codes whenever possible.

     -C      Force multi-column output; this is the default when output is to
             a terminal.

     -c      Use time when file status was last changed for sorting (-t) or
             long printing (-l).

     -d      Directories are listed as plain files (not searched recursively).

EXAMPLES
     The following is how to do an ls listing sorted by increasing size

           ls -lrS
```

Submit to Canvas (and *NOT* CODE)

Once you successfully completed the aforementioned exercises upload the following files to Canvas.

i. The C++ source file of the program developed as part of this lab.
ii. A text file for the manual page if you do the extra credit.

Ensure you actually **submit** the files after uploading them to Canvas and the files are **NOT** empty/template files.