John Jones
ID: 1001639122
CSE 4382-001
11.24.21

Assignment 11

Description of How the Code Works

My code utilizes SQLite version 3 and is built in C++. Inputs are passed into the program from the command line arguments. These arguments are converted to strings in the code and then checked against regular expression statements in a Boolean returned function to validate the input. If the input is correct, it is sent into the database with SQL statements that are programmed in the code.

Once data is written to/deleted from the database, **a log entry is created** which shows the **user id who is running the program, their username, and the values that were written or deleted from the database.** The log file is **only viewable by the root account**, once the program is compiled as **./compileSETUID.sh**, as seen below. The config file contains the database filename to use when running the program and this file is only able to be read from or written to by the root account as well.

Requirements that are met:

i. ADD, DEL, LIST functionality is present
ii. Program runs as a SetUID program to elevate privileges
iii. Audit log functionality is present
iv. Program connects to and writes to a database backend using SQLite ver 3
v. Code and other files are attached along with submission

Installation, Setup Instructions

SQLite Installation Instructions: Run the following commands in Bash Shell as user seed:

(Note: results may vary depending on repositories currently available on VM, consult the web for more detailed instructions on successfully installing)

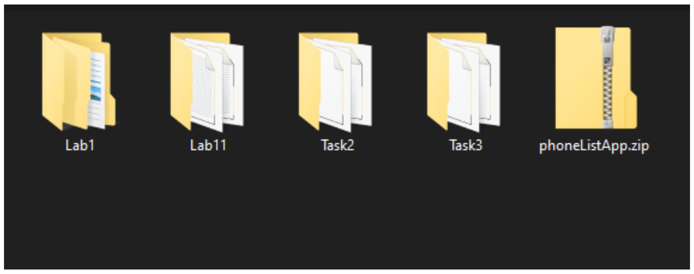
1. **sudo apt install sqlite3**
2. **sudo apt install sqlitebrowser**
3. **sudo apt-get install libsqlite3-dev**
4. **sudo apt-get install cl-sql-sqlite3 sqlitebrowser**
5. **sudo apt-get install sqlite3 libsqlite3-dev**

If the above commands do not work, **sudo apt update** and **sudo apt upgrade** should be used before to update repositories.

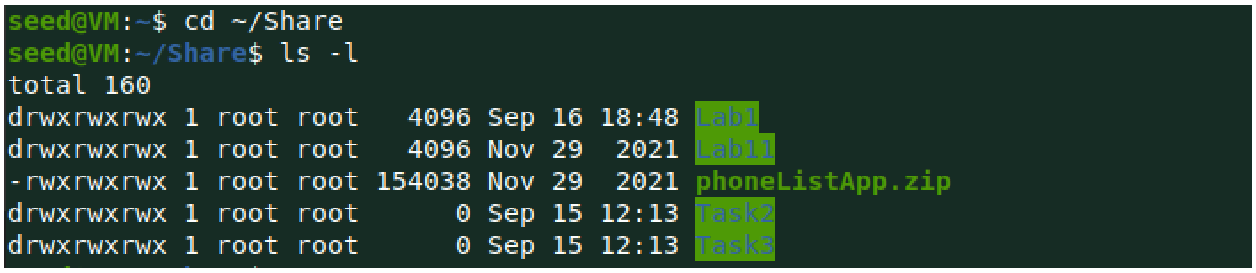
Program Installation Instructions:

(It is assumed that installation users are running Seed-Ubuntu 20.04 VM. Please see other assumptions down below.)

1. Copy the included zip file **phoneListApp.zip** to the **Share** directory used in the seed VM from withing your main OS *(I used Windows 10 GUI here)*

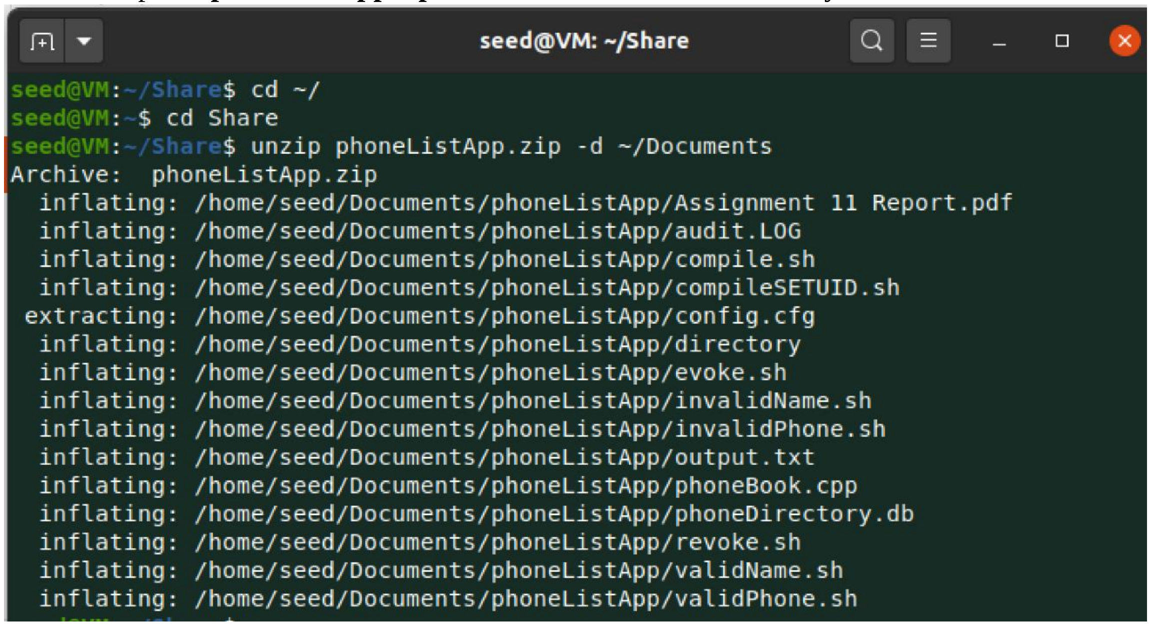


2. Launch your VM and run the **cd ~/Share** command followed by the **ls -l** command to verify that the zip file is now copied successfully to the Share folder.



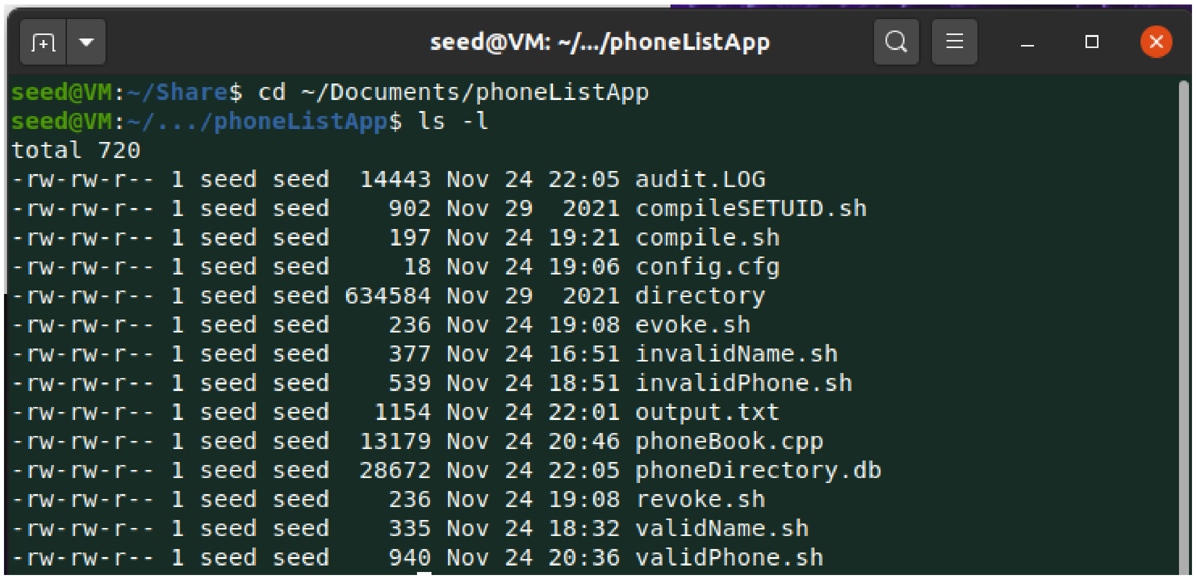
```
seed@VM:~$ cd ~/Share
seed@VM:~/Share$ ls -l
total 160
drwxrwxrwx 1 root root   4096 Sep 16 18:48 Lab1
drwxrwxrwx 1 root root   4096 Nov 29  2021 Lab11
-rwxrwxrwx 1 root root 154038 Nov 29  2021 phoneListApp.zip
drwxrwxrwx 1 root root      0 Sep 15 12:13 Task2
drwxrwxrwx 1 root root      0 Sep 15 12:13 Task3
```

3. In your VM, run the following commands in the image below to navigate to the share directory and unzip the **phoneListApp.zip** file in the **~/Documents** directory on the VM.



```
seed@VM:~/Share$ cd ~/
seed@VM:~$ cd Share
seed@VM:~/Share$ unzip phoneListApp.zip -d ~/Documents
Archive:  phoneListApp.zip
  inflating: /home/seed/Documents/phoneListApp/Assignment 11 Report.pdf
  inflating: /home/seed/Documents/phoneListApp/audit.LOG
  inflating: /home/seed/Documents/phoneListApp/compile.sh
  inflating: /home/seed/Documents/phoneListApp/compileSETUID.sh
 extracting: /home/seed/Documents/phoneListApp/config.cfg
  inflating: /home/seed/Documents/phoneListApp/directory
  inflating: /home/seed/Documents/phoneListApp/evoke.sh
  inflating: /home/seed/Documents/phoneListApp/invalidName.sh
  inflating: /home/seed/Documents/phoneListApp/invalidPhone.sh
  inflating: /home/seed/Documents/phoneListApp/output.txt
  inflating: /home/seed/Documents/phoneListApp/phoneBook.cpp
  inflating: /home/seed/Documents/phoneListApp/phoneDirectory.db
  inflating: /home/seed/Documents/phoneListApp/revoke.sh
  inflating: /home/seed/Documents/phoneListApp/validName.sh
  inflating: /home/seed/Documents/phoneListApp/validPhone.sh
```

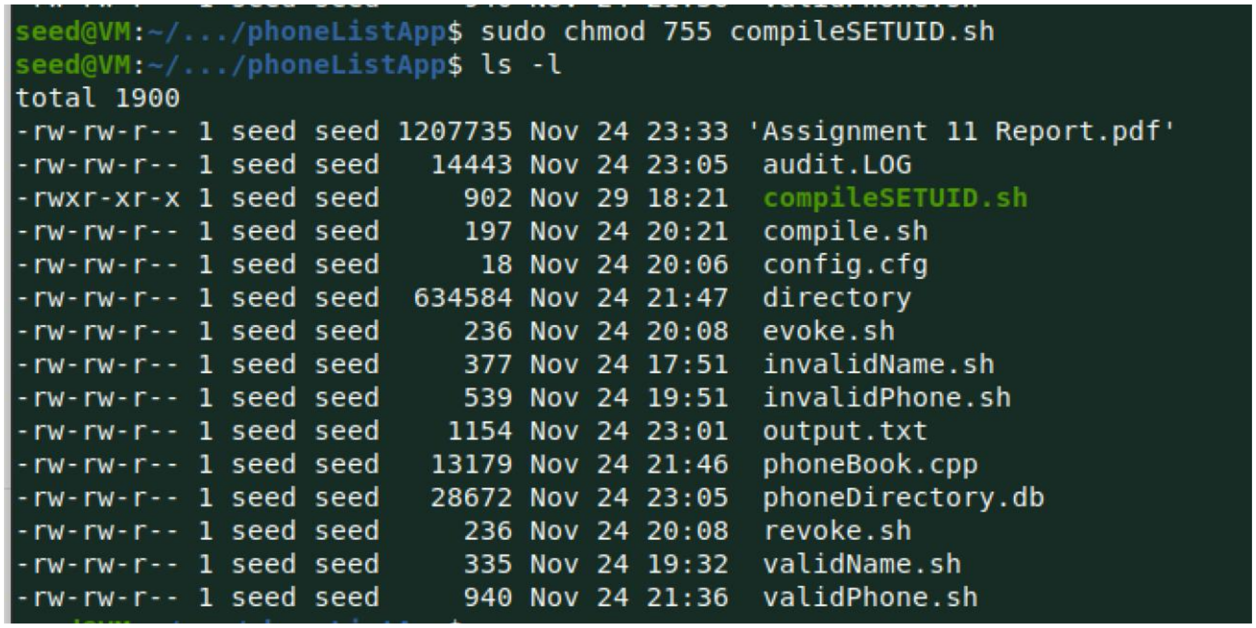
4. Now, run the following **cd ~/Documents/phoneListApp** command to navigate to the newly created directory. I used the **ls -l** command here to verify that all of the files were correctly extracted.



```
seed@VM:~/Share$ cd ~/Documents/phoneListApp
seed@VM:~/.../phoneListApp$ ls -l
total 720
-rw-rw-r-- 1 seed seed  14443 Nov 24 22:05 audit.LOG
-rw-rw-r-- 1 seed seed    902 Nov 29  2021 compileSETUID.sh
-rw-rw-r-- 1 seed seed    197 Nov 24 19:21 compile.sh
-rw-rw-r-- 1 seed seed     18 Nov 24 19:06 config.cfg
-rw-rw-r-- 1 seed seed 634584 Nov 29  2021 directory
-rw-rw-r-- 1 seed seed    236 Nov 24 19:08 evoke.sh
-rw-rw-r-- 1 seed seed    377 Nov 24 16:51 invalidName.sh
-rw-rw-r-- 1 seed seed    539 Nov 24 18:51 invalidPhone.sh
-rw-rw-r-- 1 seed seed   1154 Nov 24 22:01 output.txt
-rw-rw-r-- 1 seed seed  13179 Nov 24 20:46 phoneBook.cpp
-rw-rw-r-- 1 seed seed  28672 Nov 24 22:05 phoneDirectory.db
-rw-rw-r-- 1 seed seed    236 Nov 24 19:08 revoke.sh
-rw-rw-r-- 1 seed seed    335 Nov 24 18:32 validName.sh
-rw-rw-r-- 1 seed seed    940 Nov 24 20:36 validPhone.sh
```

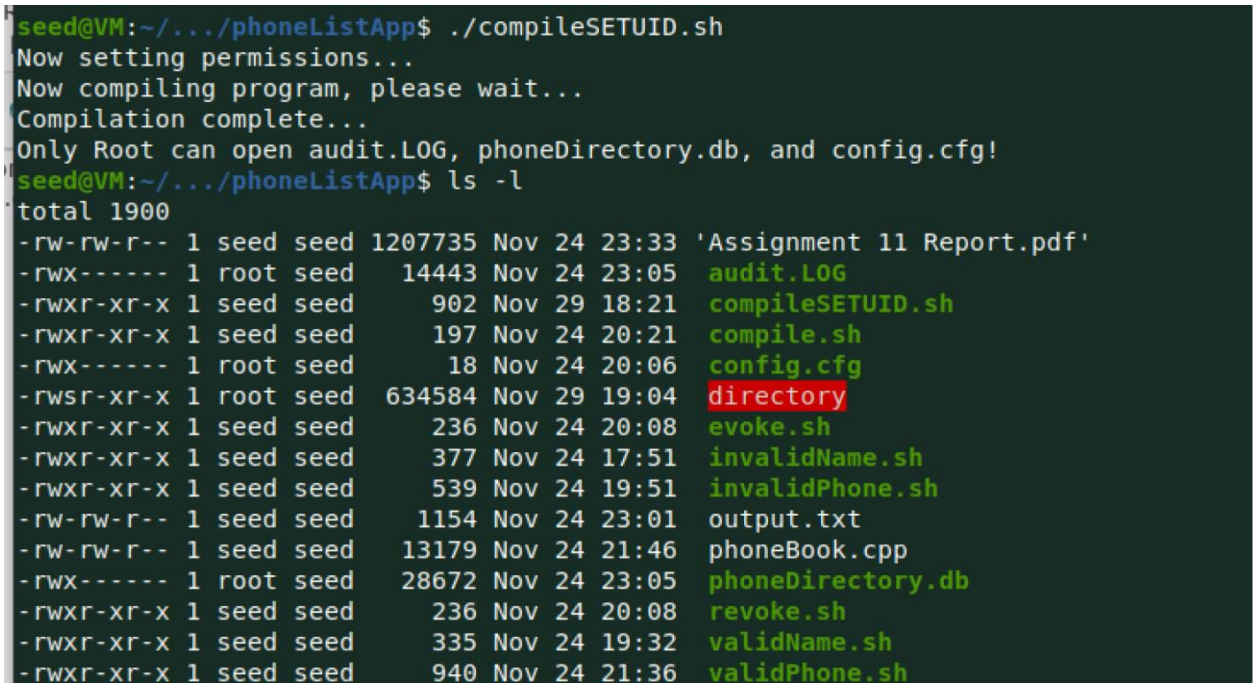
5. Once here, run the **sudo chmod 755 compileSETUID.sh** script to change the permissions of this file to execute. I then run the ls -l command here to make sure this has happened.

   (Note: It is important to run this chmod command here in order to be able to run shell scripts.From my experience this is not necessary if running directly from the Share folder!)



```
seed@VM:~/.../phoneListApp$ sudo chmod 755 compileSETUID.sh
seed@VM:~/.../phoneListApp$ ls -l
total 1900
-rw-rw-r-- 1 seed seed 1207735 Nov 24 23:33 'Assignment 11 Report.pdf'
-rw-rw-r-- 1 seed seed   14443 Nov 24 23:05  audit.LOG
-rwxr-xr-x 1 seed seed     902 Nov 29 18:21  compileSETUID.sh
-rw-rw-r-- 1 seed seed     197 Nov 24 20:21  compile.sh
-rw-rw-r-- 1 seed seed      18 Nov 24 20:06  config.cfg
-rw-rw-r-- 1 seed seed  634584 Nov 24 21:47  directory
-rw-rw-r-- 1 seed seed     236 Nov 24 20:08  evoke.sh
-rw-rw-r-- 1 seed seed     377 Nov 24 17:51  invalidName.sh
-rw-rw-r-- 1 seed seed     539 Nov 24 19:51  invalidPhone.sh
-rw-rw-r-- 1 seed seed    1154 Nov 24 23:01  output.txt
-rw-rw-r-- 1 seed seed   13179 Nov 24 21:46  phoneBook.cpp
-rw-rw-r-- 1 seed seed   28672 Nov 24 23:05  phoneDirectory.db
-rw-rw-r-- 1 seed seed     236 Nov 24 20:08  revoke.sh
-rw-rw-r-- 1 seed seed     335 Nov 24 19:32  validName.sh
-rw-rw-r-- 1 seed seed     940 Nov 24 21:36  validPhone.sh
```

6. From here we can run the **compileSETUID.sh** script in order to execute the commands needed to setup the remaining files and also compile our program in one execution. (*Note: it takes a few seconds when compiling the actual program, I believe due to the regex header that is included.*)

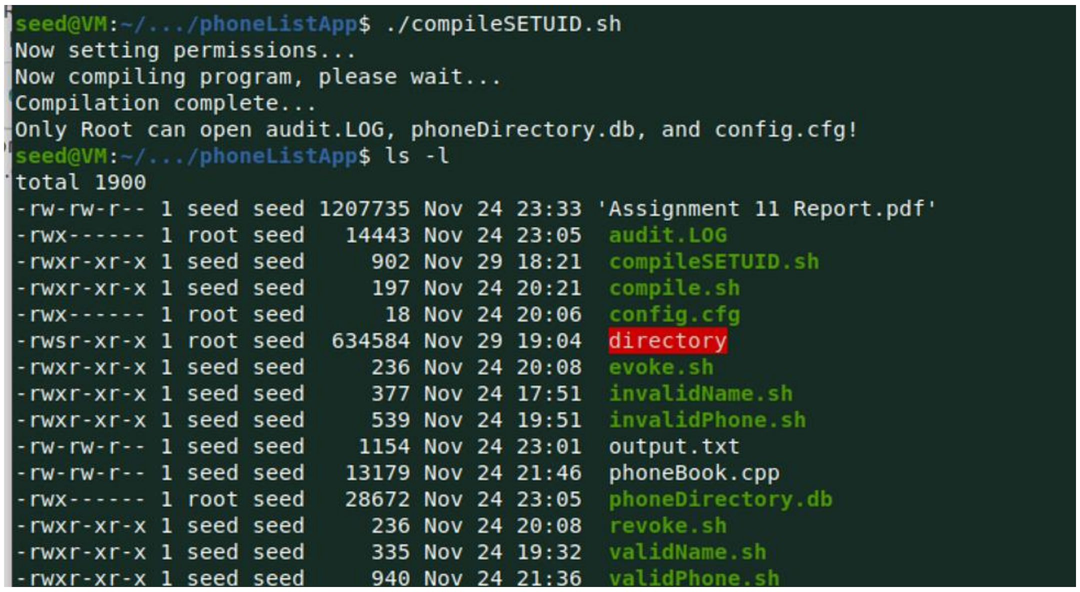


```
seed@VM:~/.../phoneListApp$ ./compileSETUID.sh
Now setting permissions...
Now compiling program, please wait...
Compilation complete...
Only Root can open audit.LOG, phoneDirectory.db, and config.cfg!
seed@VM:~/.../phoneListApp$ ls -l
total 1900
-rw-rw-r-- 1 seed seed 1207735 Nov 24 23:33 'Assignment 11 Report.pdf'
-rwx------ 1 root seed   14443 Nov 24 23:05  audit.LOG
-rwxr-xr-x 1 seed seed     902 Nov 29 18:21  compileSETUID.sh
-rwxr-xr-x 1 seed seed     197 Nov 24 20:21  compile.sh
-rwx------ 1 root seed      18 Nov 24 20:06  config.cfg
-rwsr-xr-x 1 root seed  634584 Nov 29 19:04  directory
-rwxr-xr-x 1 seed seed     236 Nov 24 20:08  evoke.sh
-rwxr-xr-x 1 seed seed     377 Nov 24 17:51  invalidName.sh
-rwxr-xr-x 1 seed seed     539 Nov 24 19:51  invalidPhone.sh
-rw-rw-r-- 1 seed seed    1154 Nov 24 23:01  output.txt
-rw-rw-r-- 1 seed seed   13179 Nov 24 21:46  phoneBook.cpp
-rwx------ 1 root seed   28672 Nov 24 23:05  phoneDirectory.db
-rwxr-xr-x 1 seed seed     236 Nov 24 20:08  revoke.sh
-rwxr-xr-x 1 seed seed     335 Nov 24 19:32  validName.sh
-rwxr-xr-x 1 seed seed     940 Nov 24 21:36  validPhone.sh
```

Compilation Instructions

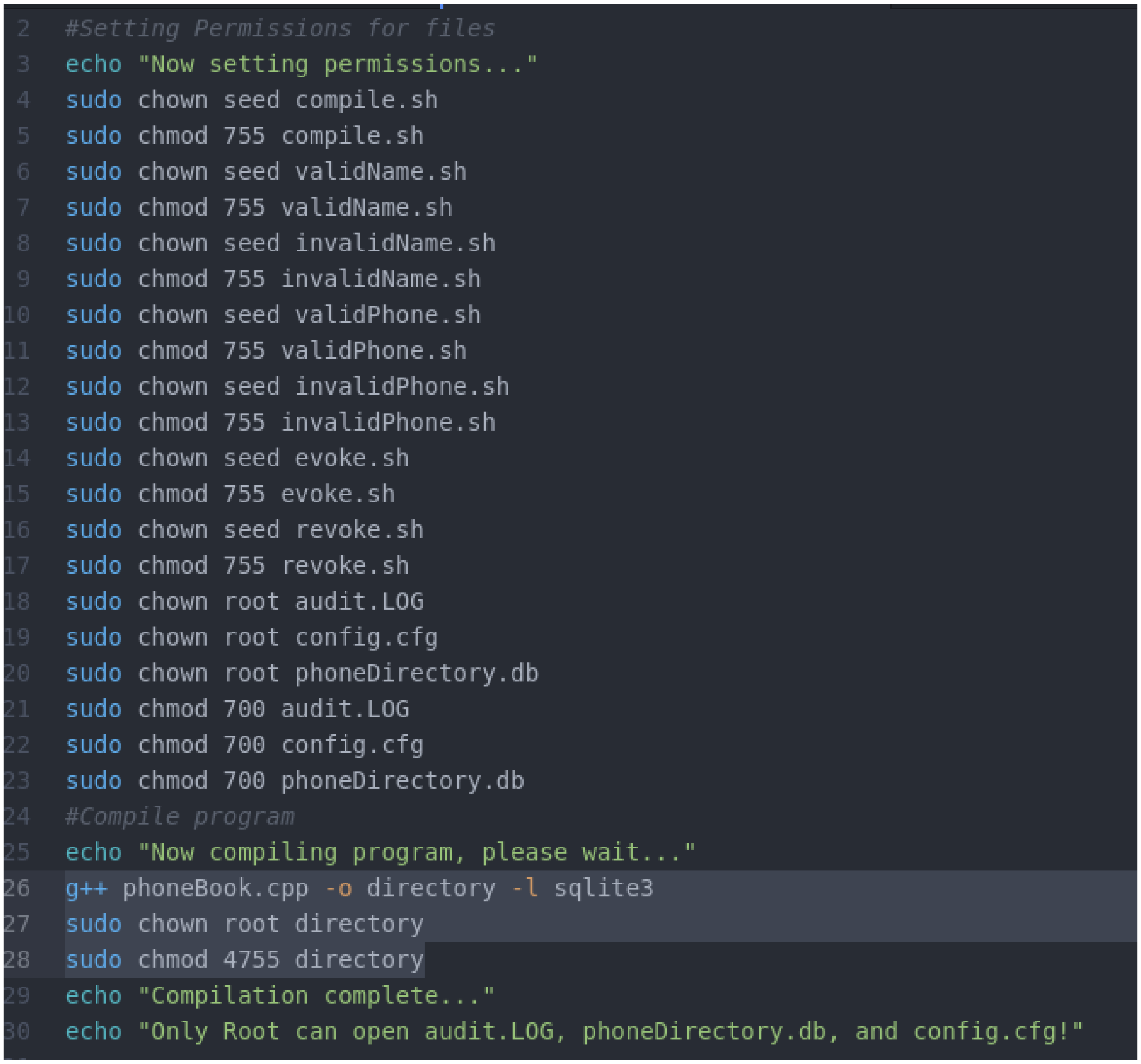
Run the following shell commands on the command line as user seed

1. Please complete the installation instructions given above to ensure that proper permissions are set to be able to run the shell scripts given. *(Note: This may not be needed if running directly from the ~/Share folder as it is root owned.)*
2. Run the command **./compileSETUID.**sh to set proper permissions and to compile the program.



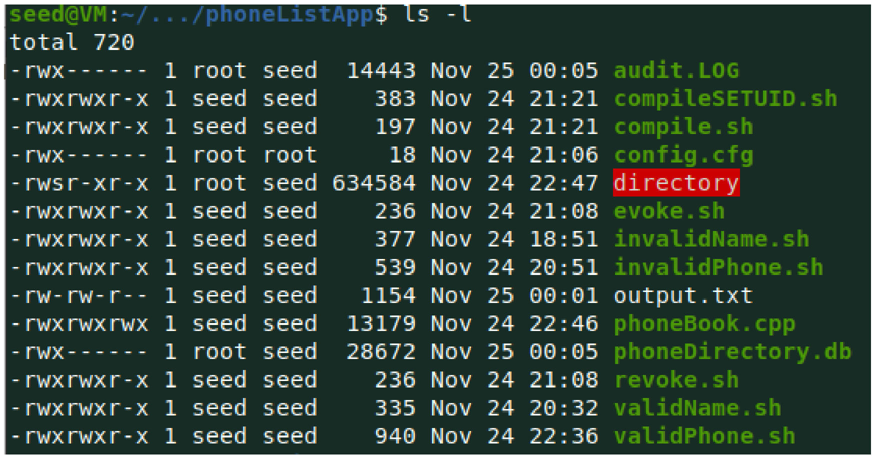
```
seed@VM:~/.../phoneListApp$ ./compileSETUID.sh
Now setting permissions...
Now compiling program, please wait...
Compilation complete...
Only Root can open audit.LOG, phoneDirectory.db, and config.cfg!
seed@VM:~/.../phoneListApp$ ls -l
total 1900
-rw-rw-r-- 1 seed seed 1207735 Nov 24 23:33 'Assignment 11 Report.pdf'
-rwx------ 1 root seed   14443 Nov 24 23:05  audit.LOG
-rwxr-xr-x 1 seed seed     902 Nov 29 18:21  compileSETUID.sh
-rwxr-xr-x 1 seed seed     197 Nov 24 20:21  compile.sh
-rwx------ 1 root seed      18 Nov 24 20:06  config.cfg
-rwsr-xr-x 1 root seed  634584 Nov 29 19:04  directory
-rwxr-xr-x 1 seed seed     236 Nov 24 20:08  evoke.sh
-rwxr-xr-x 1 seed seed     377 Nov 24 17:51  invalidName.sh
-rwxr-xr-x 1 seed seed     539 Nov 24 19:51  invalidPhone.sh
-rw-rw-r-- 1 seed seed    1154 Nov 24 23:01  output.txt
-rw-rw-r-- 1 seed seed   13179 Nov 24 21:46  phoneBook.cpp
-rwx------ 1 root seed   28672 Nov 24 23:05  phoneDirectory.db
-rwxr-xr-x 1 seed seed     236 Nov 24 20:08  revoke.sh
-rwxr-xr-x 1 seed seed     335 Nov 24 19:32  validName.sh
-rwxr-xr-x 1 seed seed     940 Nov 24 21:36  validPhone.sh
```

- This will compile the program named "**directory**" and will also modify this program to become a SetUID program to raise privileges. *(Note: alternatively, this program can be* *compiled with the **g++ phoneBook.cpp -o directory -l sqlite3**, **sudo chown root directory**, and **sudo chmod 4755 directory** commands which are all included in the **compileSETUID.sh** script. This is shown below)*



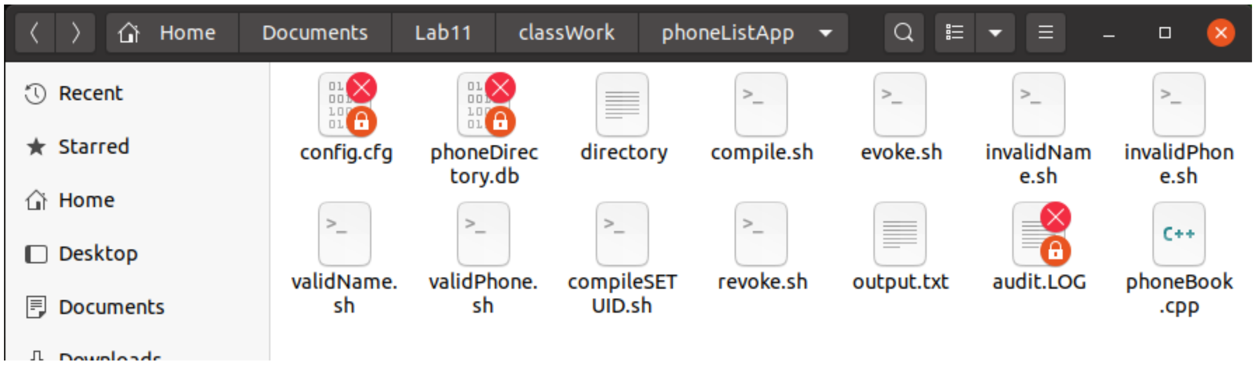
```
#Setting Permissions for files
echo "Now setting permissions..."
sudo chown seed compile.sh
sudo chmod 755 compile.sh
sudo chown seed validName.sh
sudo chmod 755 validName.sh
sudo chown seed invalidName.sh
sudo chmod 755 invalidName.sh
sudo chown seed validPhone.sh
sudo chmod 755 validPhone.sh
sudo chown seed invalidPhone.sh
sudo chmod 755 invalidPhone.sh
sudo chown seed evoke.sh
sudo chmod 755 evoke.sh
sudo chown seed revoke.sh
sudo chmod 755 revoke.sh
sudo chown root audit.LOG
sudo chown root config.cfg
sudo chown root phoneDirectory.db
sudo chmod 700 audit.LOG
sudo chmod 700 config.cfg
sudo chmod 700 phoneDirectory.db
#Compile program
echo "Now compiling program, please wait..."
g++ phoneBook.cpp -o directory -l sqlite3
sudo chown root directory
sudo chmod 4755 directory
echo "Compilation complete..."
echo "Only Root can open audit.LOG, phoneDirectory.db, and config.cfg!"
```

- This will also make the **audit.LOG**, **phoneDirectory.db**, and **config.cfg** files only readable and writeable by the root user. This is shown in the screenshot below:

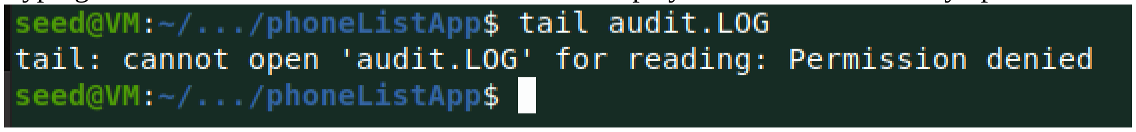


```
seed@VM:~/.../phoneListApp$ ls -l
total 720
-rwx------ 1 root seed  14443 Nov 25 00:05 audit.LOG
-rwxrwxr-x 1 seed seed    383 Nov 24 21:21 compileSETUID.sh
-rwxrwxr-x 1 seed seed    197 Nov 24 21:21 compile.sh
-rwx------ 1 root root     18 Nov 24 21:06 config.cfg
-rwsr-xr-x 1 root seed 634584 Nov 24 22:47 directory
-rwxrwxr-x 1 seed seed    236 Nov 24 21:08 evoke.sh
-rwxrwxr-x 1 seed seed    377 Nov 24 18:51 invalidName.sh
-rwxrwxr-x 1 seed seed    539 Nov 24 20:51 invalidPhone.sh
-rw-rw-r-- 1 seed seed   1154 Nov 25 00:01 output.txt
-rwxrwxrwx 1 seed seed  13179 Nov 24 22:46 phoneBook.cpp
-rwx------ 1 root seed  28672 Nov 25 00:05 phoneDirectory.db
-rwxrwxr-x 1 seed seed    236 Nov 24 21:08 revoke.sh
-rwxrwxr-x 1 seed seed    335 Nov 24 20:32 validName.sh
-rwxrwxr-x 1 seed seed    940 Nov 24 22:36 validPhone.sh
```

We can also see how files cannot be modified in the GUI of the OS here:



Typing **tail audit.LOG** on the command line also displays how this functionality operates:



```
seed@VM:~/.../phoneListApp$ tail audit.LOG
tail: cannot open 'audit.LOG' for reading: Permission denied
seed@VM:~/.../phoneListApp$
```

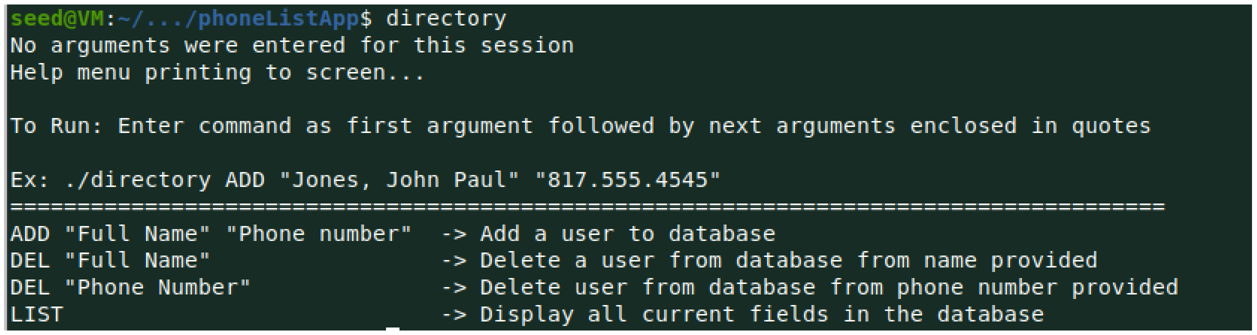
For Testing purposes, there is a shell file to compile the program as a setUID program, but retain seed read/write privileges for log, database, and configuration files. Alternatively, I have included revoke.sh bring back privileges and evoke.sh to take away privileges from the seed user.

To test, Run the following commands in shell as user seed:

1. **./compile.sh, ./revoke.sh, or ./evoke.sh**

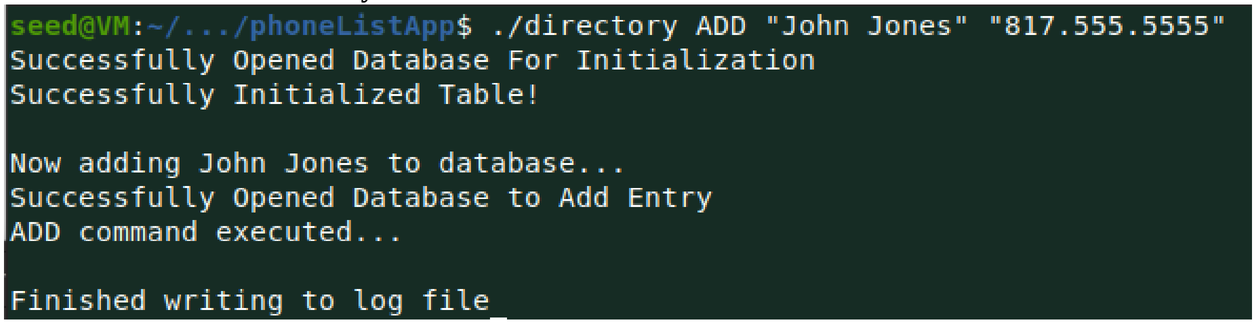
Execution Instructions

1. Begin the program by either typing **./directory** or simply **directory** on the command line followed by one of the commands **ADD**, **DEL**, and **LIST** and then the required arguments for each command.
    - If no arguments were typed in during the execution, the program will print out the **help menu** to display what commands are needed as the first argument to begin running
        i. Ex: **./directory** or **directory**



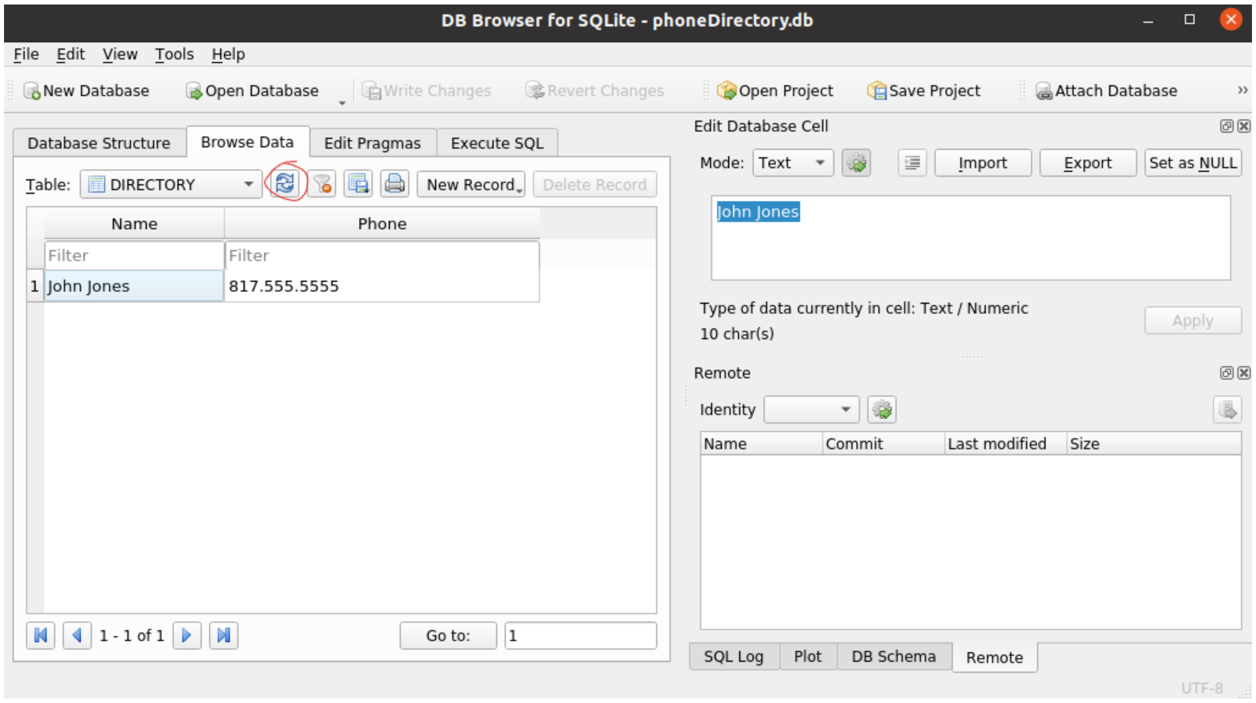
```
seed@VM:~/.../phoneListApp$ directory
No arguments were entered for this session
Help menu printing to screen...

To Run: Enter command as first argument followed by next arguments enclosed in quotes

Ex: ./directory ADD "Jones, John Paul" "817.555.4545"
=====================================================================================
ADD "Full Name" "Phone number"   -> Add a user to database
DEL "Full Name"                  -> Delete a user from database from name provided
DEL "Phone Number"               -> Delete user from database from phone number provided
LIST                             -> Display all current fields in the database
```

- The **ADD** command needs to be followed by a name and a phone number argument in the correct format in quotations.
    i. Ex: **./directory ADD "John Jones" "817.555.5555"**



```
seed@VM:~/.../phoneListApp$ ./directory ADD "John Jones" "817.555.5555"
Successfully Opened Database For Initialization
Successfully Initialized Table!

Now adding John Jones to database...
Successfully Opened Database to Add Entry
ADD command executed...

Finished writing to log file_
```

- We can see the changes with the **sudo sqlitebrowser phoneDirectory.db** command and clicking on the refresh button shown here to see the entries added.



	Name	Phone
	Filter	Filter
1	John Jones	817.555.5555

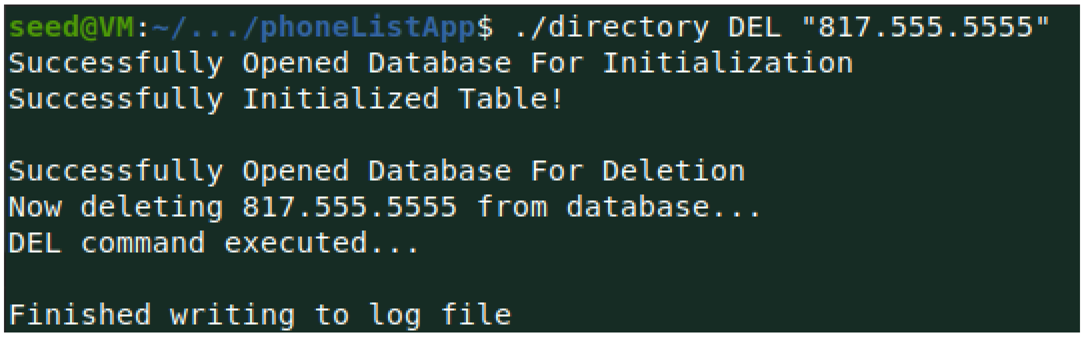
1 - 1 of 1 Go to: 1

John Jones

Type of data currently in cell: Text / Numeric
10 char(s)

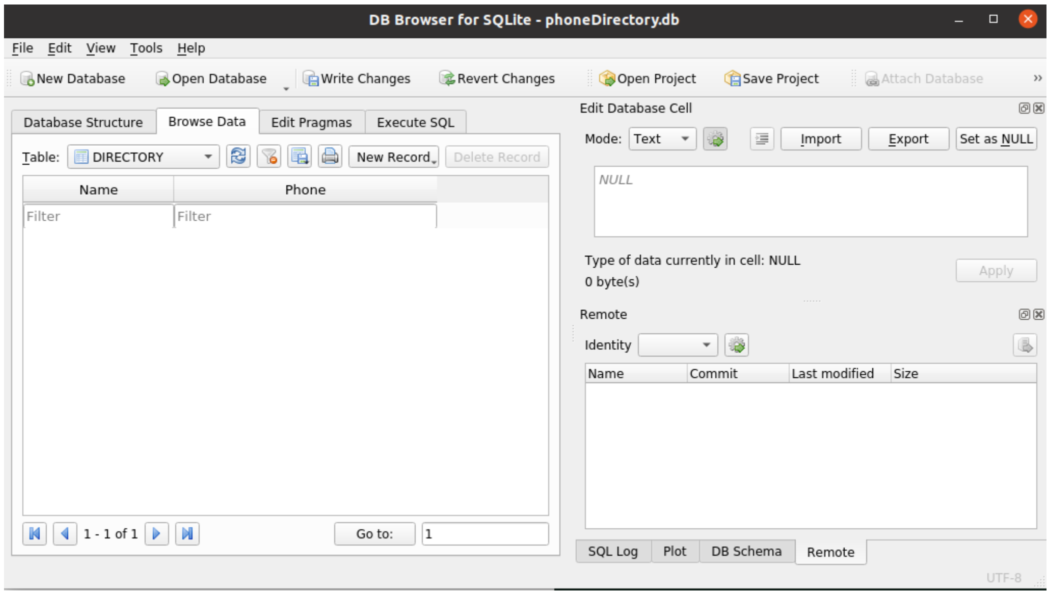
Name	Commit	Last modified	Size

- The **DEL** command needs to be followed by a **name** argument or a **phone number** argument, but not both!
    i. Ex: **./directory DEL "817.555.5555"**
    ii. Ex: **./directory DEL "John Jones"**



```
seed@VM:~/.../phoneListApp$ ./directory DEL "817.555.5555"
Successfully Opened Database For Initialization
Successfully Initialized Table!

Successfully Opened Database For Deletion
Now deleting 817.555.5555 from database...
DEL command executed...

Finished writing to log file
```

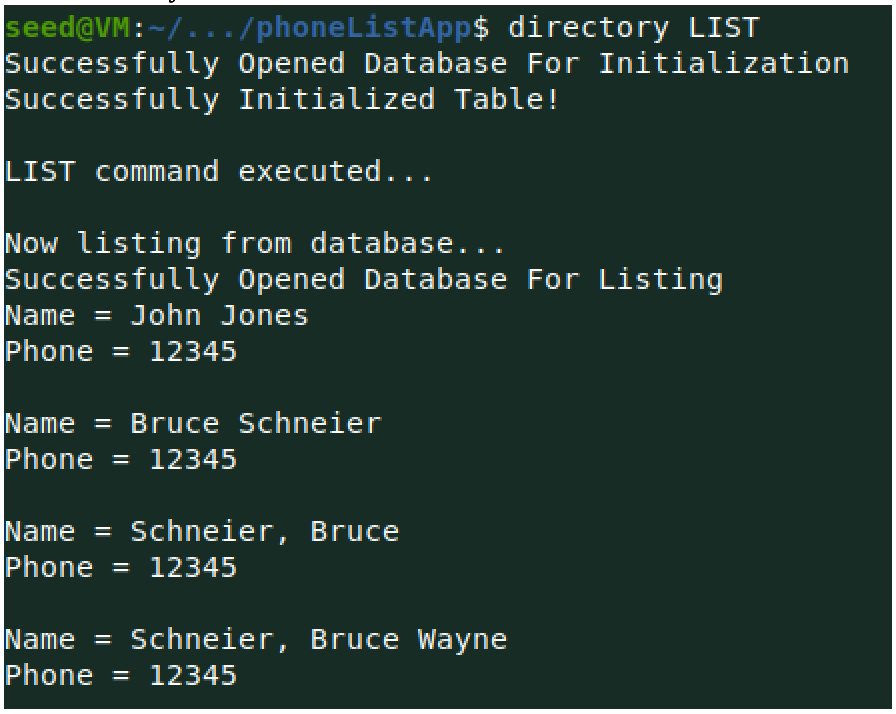
iii. The **sudo sqlitebrowser phoneDirectory.db** command can be run to view the contents of the database in the sqlitebrowser after using the delete command. Clicking the refresh button after executing commands will show what entries are deleted



Entries can be added after this to show how the **LIST** command works.

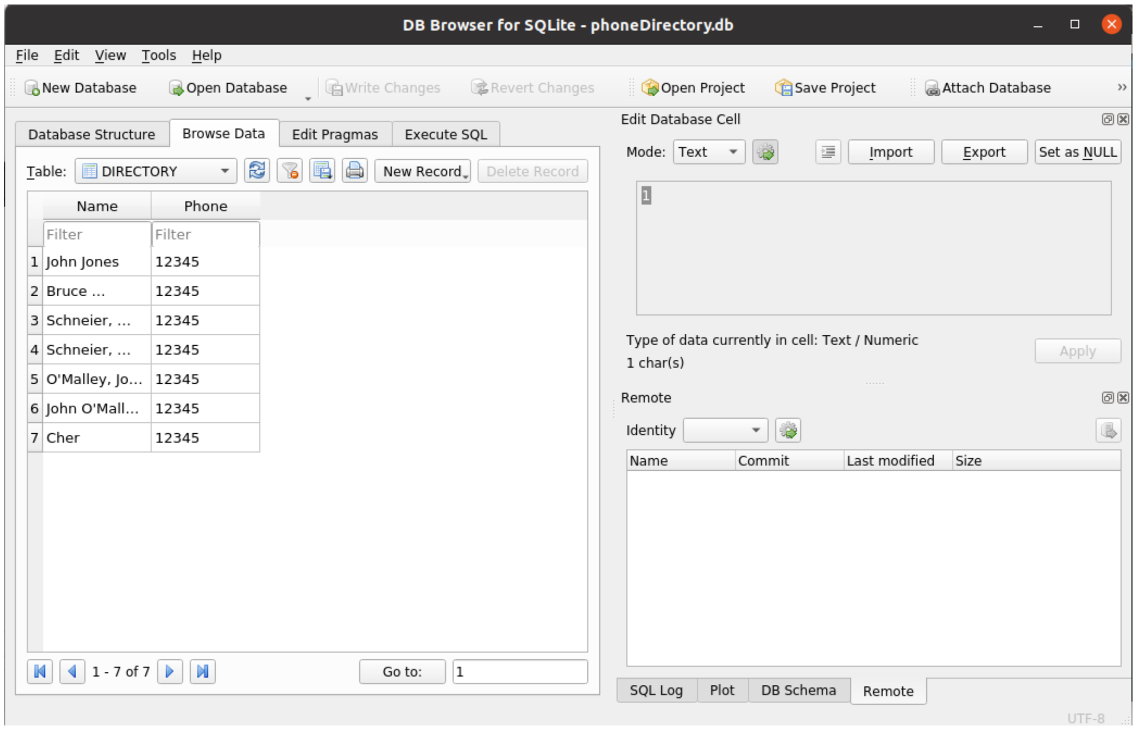
- The **LIST** command can be executed without any arguments to list out the values in the database.
- The commands **ADD**, **DEL**, and **LIST**. can also be typed in lowercase if needed.

i. Ex: **./directory LIST**



```
seed@VM:~/.../phoneListApp$ directory LIST
Successfully Opened Database For Initialization
Successfully Initialized Table!

LIST command executed...

Now listing from database...
Successfully Opened Database For Listing
Name = John Jones
Phone = 12345

Name = Bruce Schneier
Phone = 12345

Name = Schneier, Bruce
Phone = 12345

Name = Schneier, Bruce Wayne
Phone = 12345
```

ii. The **sudo sqlitebrowser phoneDirectory.db** command can be run to view the contents of the database in the sqlitebrowser. Clicking the refresh button after executing commands will show what entries are added or deleted. This is shown below:



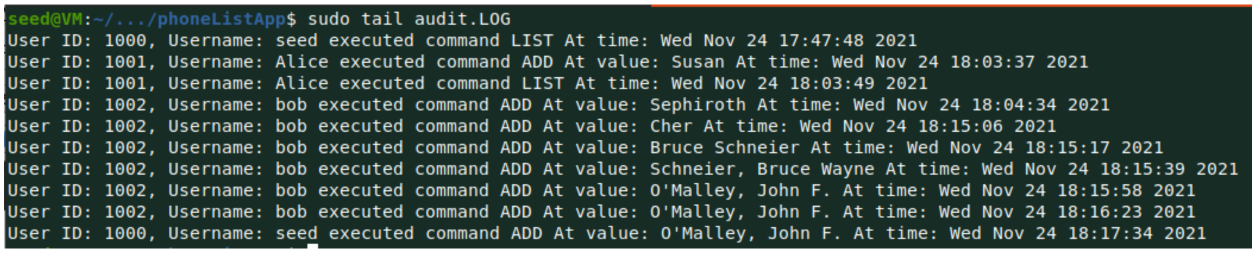
	Name	Phone
	Filter	Filter
1	John Jones	12345
2	Bruce ...	12345
3	Schneier, ...	12345
4	Schneier, ...	12345
5	O'Malley, Jo...	12345
6	John O'Mall...	12345
7	Cher	12345

1 - 7 of 7

Type of data currently in cell: Text / Numeric
1 char(s)

Name	Commit	Last modified	Size

- We can see by the execution of each of these commands here that a message is printed out to the screen that shows, "**Finished writing to log file.**" This file is written to in a manner that only the root user can view/modify this file. We can see some of the end results of the file by typing the command **sudo tail audit.LOG**, the results can be seen below:



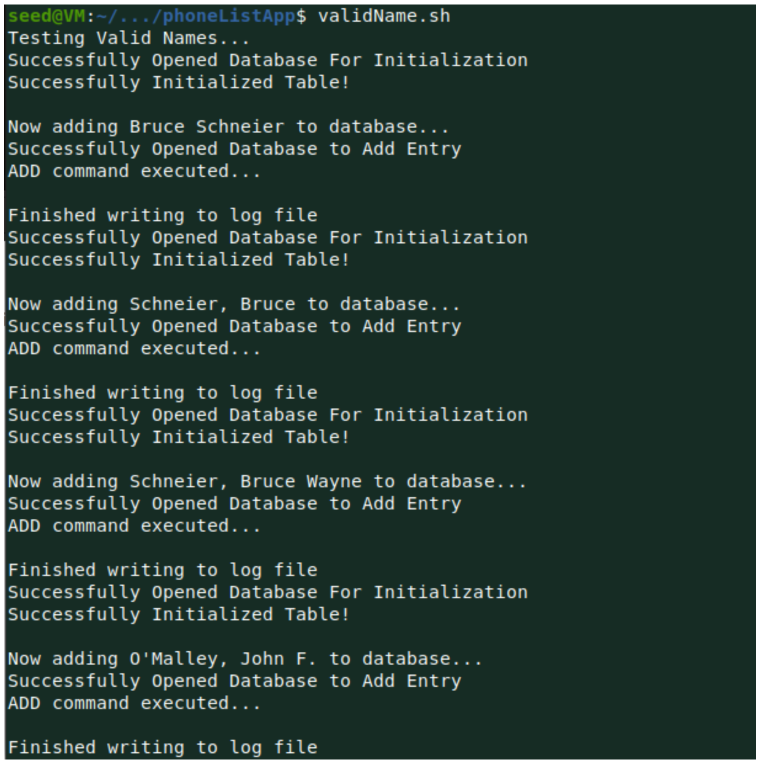
```
seed@VM:~/.../phoneListApp$ sudo tail audit.LOG
User ID: 1000, Username: seed executed command LIST At time: Wed Nov 24 17:47:48 2021
User ID: 1001, Username: Alice executed command ADD At value: Susan At time: Wed Nov 24 18:03:37 2021
User ID: 1001, Username: Alice executed command LIST At time: Wed Nov 24 18:03:49 2021
User ID: 1002, Username: bob executed command ADD At value: Sephiroth At time: Wed Nov 24 18:04:34 2021
User ID: 1002, Username: bob executed command ADD At value: Cher At time: Wed Nov 24 18:15:06 2021
User ID: 1002, Username: bob executed command ADD At value: Bruce Schneier At time: Wed Nov 24 18:15:17 2021
User ID: 1002, Username: bob executed command ADD At value: Schneier, Bruce Wayne At time: Wed Nov 24 18:15:39 2021
User ID: 1002, Username: bob executed command ADD At value: O'Malley, John F. At time: Wed Nov 24 18:15:58 2021
User ID: 1002, Username: bob executed command ADD At value: O'Malley, John F. At time: Wed Nov 24 18:16:23 2021
User ID: 1000, Username: seed executed command ADD At value: O'Malley, John F. At time: Wed Nov 24 18:17:34 2021
```

Input Validation Testing

I have included two script files for testing **valid** inputs that were specified in the instructions, **validName.sh** and **validPhone.sh.** I have also included 2 more scripts for testing **invalid** inputs against valid regular expressions named **invalidName.sh** and **invalidPhone.sh**.

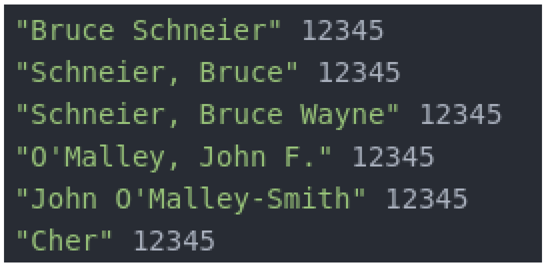
Testing Valid Names From Regular Expressions:

- To start testing, the database is cleared of all entries using the sqlite browser. This is done by selecting all entries and clicking the **delete record** button. Then clicking the **Write Changes** button. The command **./validName.sh** is then entered in the terminal or simply typing **validName.sh** is used to run the script. We can see that the script tests 6 test cases for valid names from regular expressions based on the instructions from the assignment. Any other formatted names are not allowed.



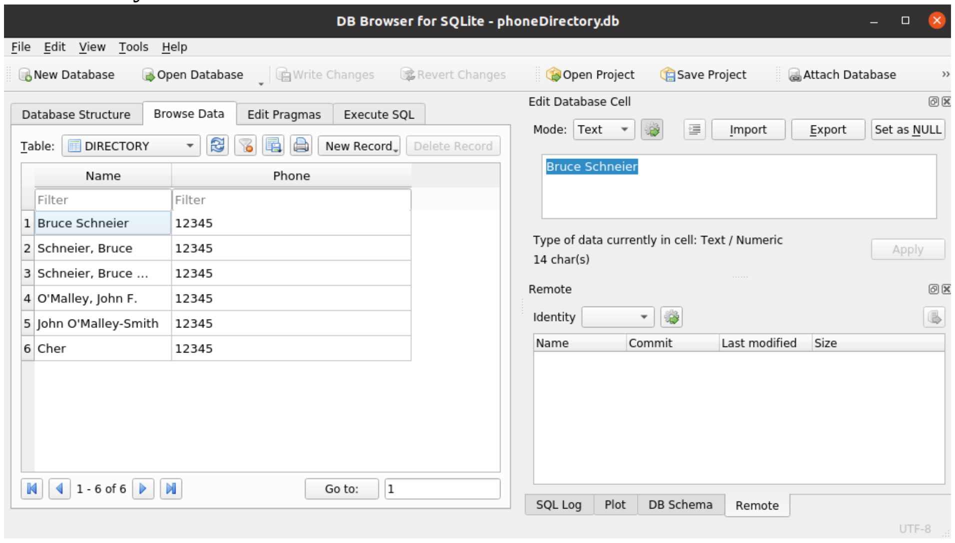
```
seed@VM:~/.../phoneListApp$ validName.sh
Testing Valid Names...
Successfully Opened Database For Initialization
Successfully Initialized Table!

Now adding Bruce Schneier to database...
Successfully Opened Database to Add Entry
ADD command executed...

Finished writing to log file
Successfully Opened Database For Initialization
Successfully Initialized Table!

Now adding Schneier, Bruce to database...
Successfully Opened Database to Add Entry
ADD command executed...

Finished writing to log file
Successfully Opened Database For Initialization
Successfully Initialized Table!

Now adding Schneier, Bruce Wayne to database...
Successfully Opened Database to Add Entry
ADD command executed...

Finished writing to log file
Successfully Opened Database For Initialization
Successfully Initialized Table!

Now adding O'Malley, John F. to database...
Successfully Opened Database to Add Entry
ADD command executed...

Finished writing to log file
```

The formats that are allowed are as follows:



```
"Bruce Schneier" 12345
"Schneier, Bruce" 12345
"Schneier, Bruce Wayne" 12345
"O'Malley, John F." 12345
"John O'Malley-Smith" 12345
"Cher" 12345
```

Clicking the refresh button in the sqlite browser, we can see that the 6 entries tested here are successfully written to the database.



	Name	Phone
1	Bruce Schneier	12345
2	Schneier, Bruce	12345
3	Schneier, Bruce ...	12345
4	O'Malley, John F.	12345
5	John O'Malley-Smith	12345
6	Cher	12345

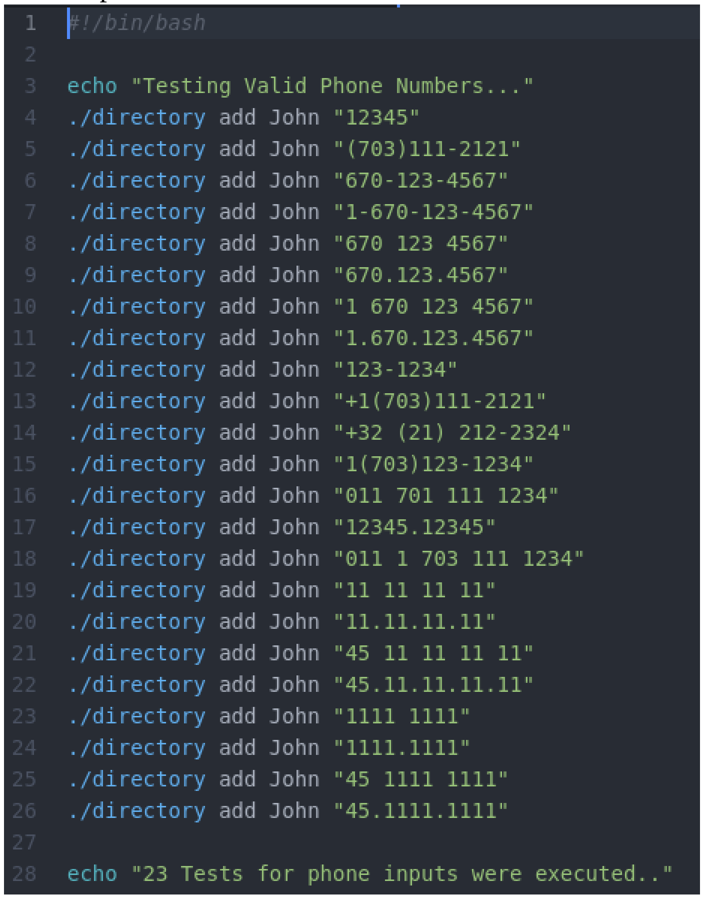
1 - 6 of 6 Go to: 1

Bruce Schneier

Type of data currently in cell: Text / Numeric
14 char(s)

Name	Commit	Last modified	Size

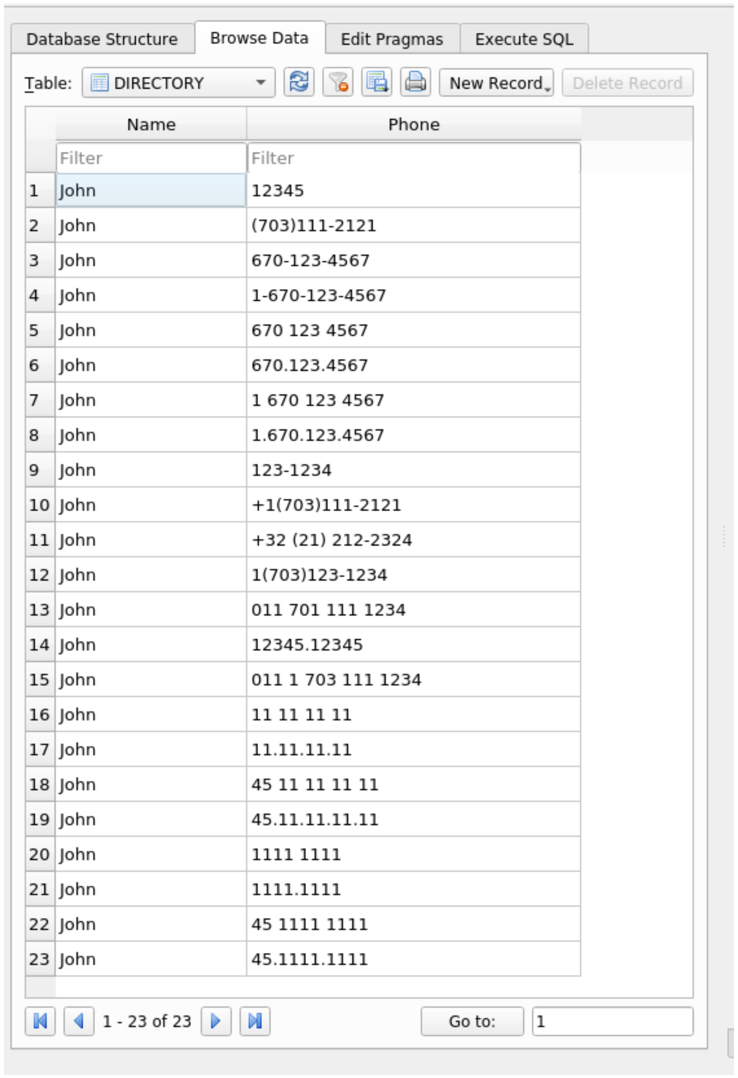
The script written/used is shown below:



```bash
#!/bin/bash

echo "Testing Valid Phone Numbers..."
./directory add John "12345"
./directory add John "(703)111-2121"
./directory add John "670-123-4567"
./directory add John "1-670-123-4567"
./directory add John "670 123 4567"
./directory add John "670.123.4567"
./directory add John "1 670 123 4567"
./directory add John "1.670.123.4567"
./directory add John "123-1234"
./directory add John "+1(703)111-2121"
./directory add John "+32 (21) 212-2324"
./directory add John "1(703)123-1234"
./directory add John "011 701 111 1234"
./directory add John "12345.12345"
./directory add John "011 1 703 111 1234"
./directory add John "11 11 11 11"
./directory add John "11.11.11.11"
./directory add John "45 11 11 11 11"
./directory add John "45.11.11.11.11"
./directory add John "1111 1111"
./directory add John "1111.1111"
./directory add John "45 1111 1111"
./directory add John "45.1111.1111"

echo "23 Tests for phone inputs were executed.."
```

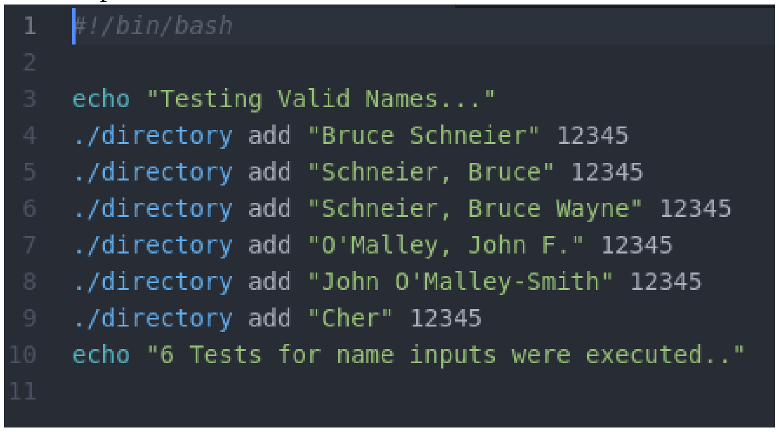
Testing Valid Phone Numbers From Regular Expressions:

- To start testing, I begin by clearing all records from the database from within the sqlite browser. I then run the **./validPhone.sh** command to test the **23 whitelisted** regular expressions. We can see below that all 23 inputs are written to the database file.



	Name	Phone
1	John	12345
2	John	(703)111-2121
3	John	670-123-4567
4	John	1-670-123-4567
5	John	670 123 4567
6	John	670.123.4567
7	John	1 670 123 4567
8	John	1.670.123.4567
9	John	123-1234
10	John	+1(703)111-2121
11	John	+32 (21) 212-2324
12	John	1(703)123-1234
13	John	011 701 111 1234
14	John	12345.12345
15	John	011 1 703 111 1234
16	John	11 11 11 11
17	John	11.11.11.11
18	John	45 11 11 11 11
19	John	45.11.11.11.11
20	John	1111 1111
21	John	1111.1111
22	John	45 1111 1111
23	John	45.1111.1111

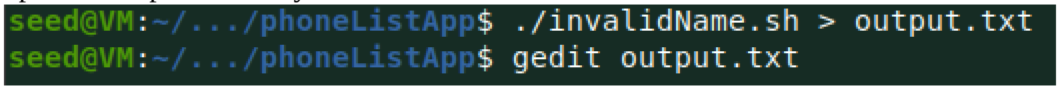
The script written/used is shown below:



```bash
#!/bin/bash

echo "Testing Valid Names..."
./directory add "Bruce Schneier" 12345
./directory add "Schneier, Bruce" 12345
./directory add "Schneier, Bruce Wayne" 12345
./directory add "O'Malley, John F." 12345
./directory add "John O'Malley-Smith" 12345
./directory add "Cher" 12345
echo "6 Tests for name inputs were executed.."

```

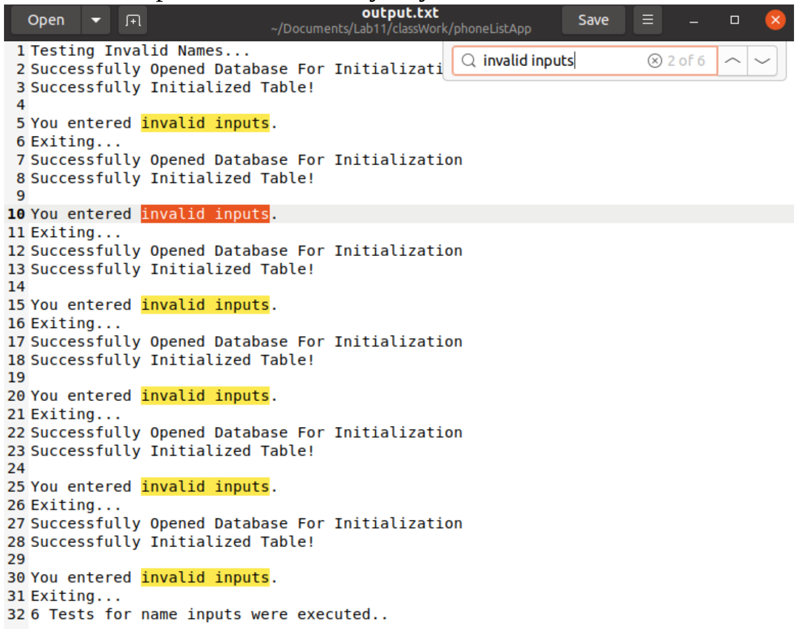
Testing Invalid Name Inputs With Regular Expressions:

- To start this task, I begin by clearing the database once again. I then run the command **./invalidName.sh > output.txt** on the command line. I then run the **gedit output.txt** command to open the output file in my text editor.



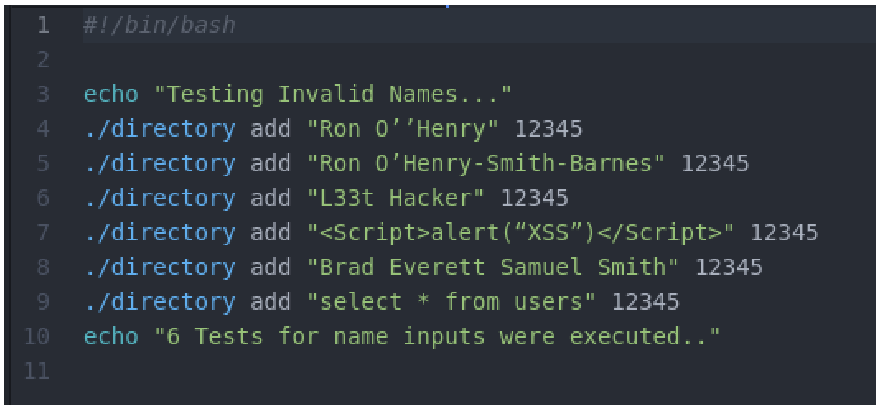
```
seed@VM:~/.../phoneListApp$ ./invalidName.sh > output.txt
seed@VM:~/.../phoneListApp$ gedit output.txt
```

- Once here, I press **ctrl+F** on my keyboard to find all instances of the phrase **“invalid inputs.”**



```
Testing Invalid Names...
Successfully Opened Database For Initialization
Successfully Initialized Table!

You entered invalid inputs.
Exiting...
Successfully Opened Database For Initialization
Successfully Initialized Table!

You entered invalid inputs.
Exiting...
Successfully Opened Database For Initialization
Successfully Initialized Table!

You entered invalid inputs.
Exiting...
Successfully Opened Database For Initialization
Successfully Initialized Table!

You entered invalid inputs.
Exiting...
Successfully Opened Database For Initialization
Successfully Initialized Table!

You entered invalid inputs.
Exiting...
Successfully Opened Database For Initialization
Successfully Initialized Table!

You entered invalid inputs.
Exiting...
6 Tests for name inputs were executed..
```

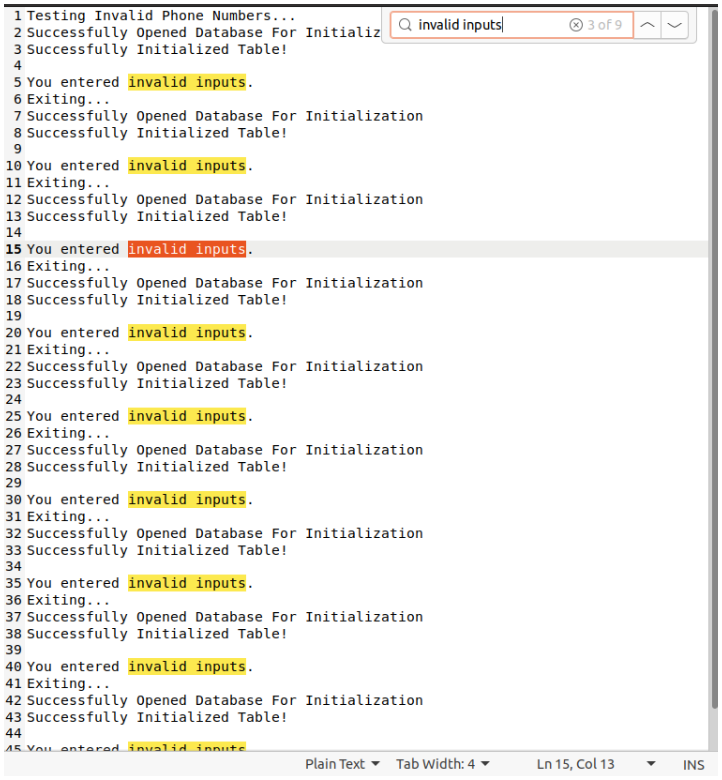
We can see here that 6 invalid inputs are found, which is what has been tested from the shell script. We can also refresh the database to see that nothing was written there. The script written and used here is shown below:



```
#!/bin/bash

echo "Testing Invalid Names..."
./directory add "Ron O’’Henry" 12345
./directory add "Ron O’Henry-Smith-Barnes" 12345
./directory add "L33t Hacker" 12345
./directory add "<Script>alert(“XSS”)</Script>" 12345
./directory add "Brad Everett Samuel Smith" 12345
./directory add "select * from users" 12345
echo "6 Tests for name inputs were executed.."
```

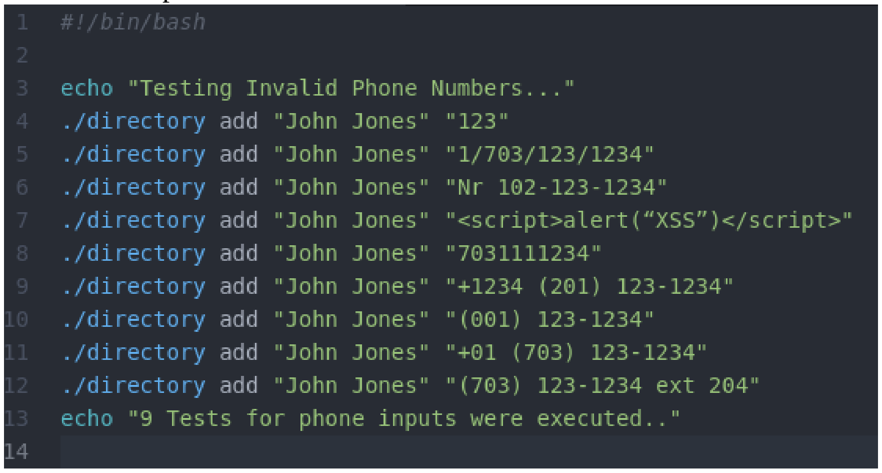
Testing Invalid Phone Numbers From Regular Expressions:

- I start by once again clearing the database. I then run **./invalidPhone.sh** > **output.txt** on the command line. I then run the **gedit output.txt** command to open the output file in my text editor. We can see here, using **ctrl+F** that **9 invalid inputs** are found, which is what was tested in the script. No values are written to the database this time as well.



```
Testing Invalid Phone Numbers...
Successfully Opened Database For Initializ
Successfully Initialized Table!

You entered invalid inputs.
Exiting...
Successfully Opened Database For Initialization
Successfully Initialized Table!

You entered invalid inputs.
Exiting...
Successfully Opened Database For Initialization
Successfully Initialized Table!

You entered invalid inputs.
Exiting...
Successfully Opened Database For Initialization
Successfully Initialized Table!

You entered invalid inputs.
Exiting...
Successfully Opened Database For Initialization
Successfully Initialized Table!

You entered invalid inputs.
Exiting...
Successfully Opened Database For Initialization
Successfully Initialized Table!

You entered invalid inputs.
Exiting...
Successfully Opened Database For Initialization
Successfully Initialized Table!

You entered invalid inputs.
Exiting...
Successfully Opened Database For Initialization
Successfully Initialized Table!

You entered invalid inputs.
Exiting...
Successfully Opened Database For Initialization
Successfully Initialized Table!

You entered invalid inputs.
```

The shell script used is shown here:



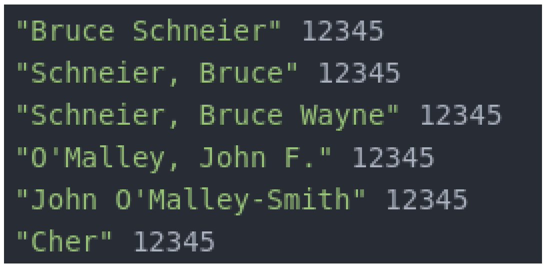
```bash
#!/bin/bash

echo "Testing Invalid Phone Numbers..."
./directory add "John Jones" "123"
./directory add "John Jones" "1/703/123/1234"
./directory add "John Jones" "Nr 102-123-1234"
./directory add "John Jones" "<script>alert(“XSS”)</script>"
./directory add "John Jones" "7031111234"
./directory add "John Jones" "+1234 (201) 123-1234"
./directory add "John Jones" "(001) 123-1234"
./directory add "John Jones" "+01 (703) 123-1234"
./directory add "John Jones" "(703) 123-1234 ext 204"
echo "9 Tests for phone inputs were executed.."

```

Assumptions

- User has **SQLite version 3** installed **(If not, see installation instructions above)**
- User is running the **Seed-Ubuntu 20.04 VM** provided by Seed labs
- User who is ***installing*** this program has root privileges to be able to execute **sudo** commands **(This is functionality is present in the Seed-Ubuntu 20.04 VM.)**
- User ***running*** this program **does not** have root privileges
- User can execute **.sh** scripts from the command line
- **Only** the **whitelisted name/phone number** formatting from the regular expressions in my code can be used as **valid inputs**. Anything else will not be allowed. These whitelist values come

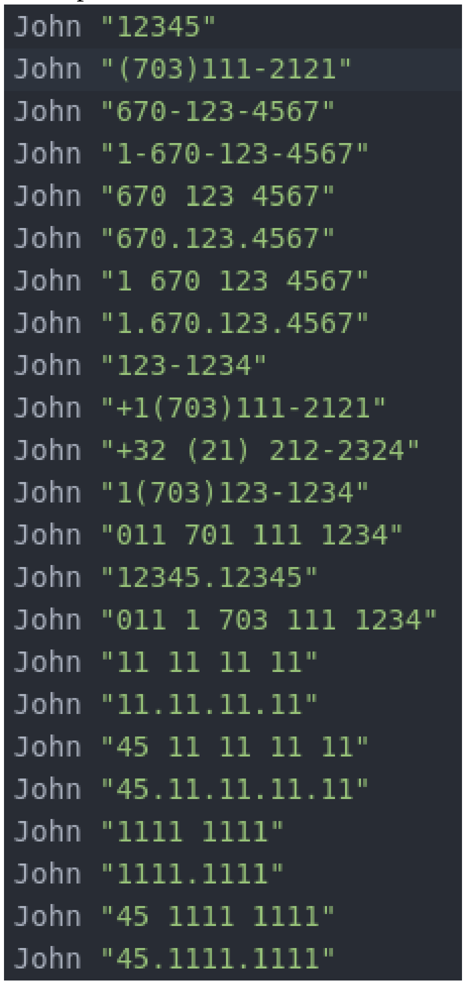
directly from the assignment instructions. These values can be tested by executing the **validName.sh** and **validPhone.sh** files. (Instructions, along with a demonstration will be shown below in the section titled **"Testing."**

- **Valid** name formats can be seen here:



```
"Bruce Schneier" 12345
"Schneier, Bruce" 12345
"Schneier, Bruce Wayne" 12345
"O'Malley, John F." 12345
"John O'Malley-Smith" 12345
"Cher" 12345
```

- Valid phone number formats can be seen here:



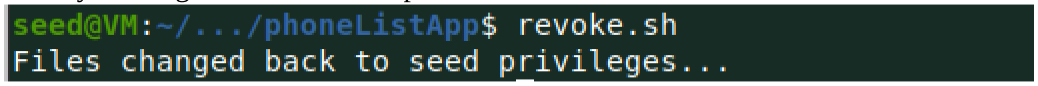
```
John "12345"
John "(703)111-2121"
John "670-123-4567"
John "1-670-123-4567"
John "670 123 4567"
John "670.123.4567"
John "1 670 123 4567"
John "1.670.123.4567"
John "123-1234"
John "+1(703)111-2121"
John "+32 (21) 212-2324"
John "1(703)123-1234"
John "011 701 111 1234"
John "12345.12345"
John "011 1 703 111 1234"
John "11 11 11 11"
John "11.11.11.11"
John "45 11 11 11 11"
John "45.11.11.11.11"
John "1111 1111"
John "1111.1111"
John "45 1111 1111"
John "45.1111.1111"
```

Pros/Cons Of My Approach

Pros

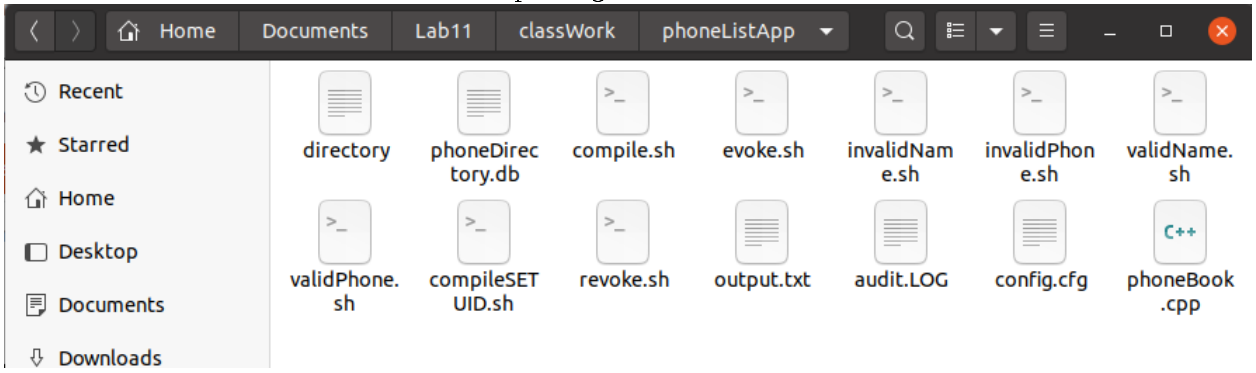
- Program seems to execute properly with the commands that are required by the assignment instructions.
- There are not many command line arguments needed to execute the code
- I wrote shell files to compile the code and set up the configuration and log files for the user's environment once the program directory is copied to their virtual machine. The user does not have to type in many arguments to compile, make the program SetUID and setup the log/config files to be root owned.

- For testing purposes, I also included a shell script to compile the code without root privileged log/config/db files. This file is called **compile.sh** There are shell scripts to simply change the privileges for these files also. They are named **revoke.sh (gives log/config/db files seed privileges)** and **evoke.sh (gives log/config/db files root privileges)** This functionality is shown here by running the revoke.sh script in the terminal:

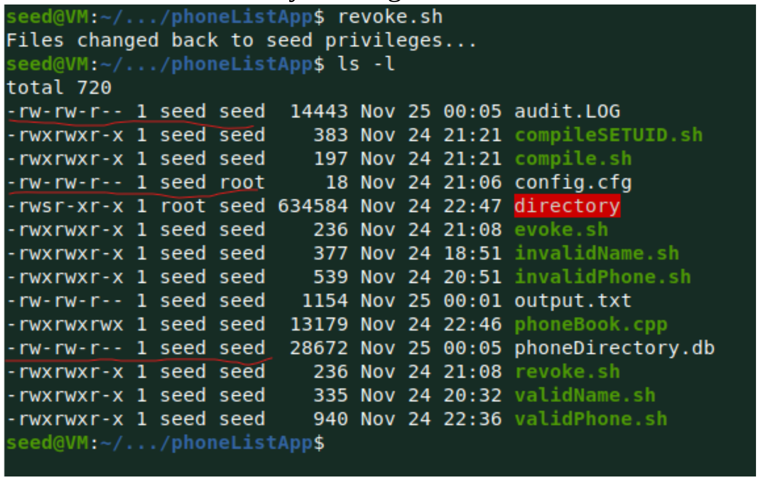


```
seed@VM:~/.../phoneListApp$ revoke.sh
Files changed back to seed privileges...
```

We can see in the GUI that the r/w privileges are returned to the user seed.

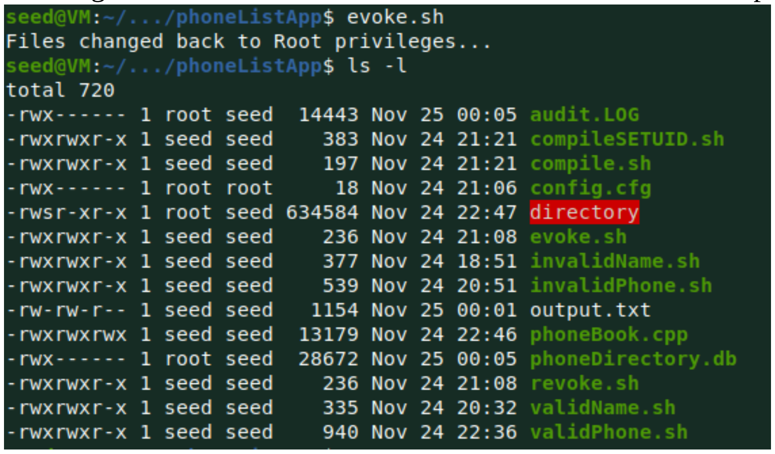


This can be confirmed by running the **ls -l** command in the terminal.



```
seed@VM:~/.../phoneListApp$ revoke.sh
Files changed back to seed privileges...
seed@VM:~/.../phoneListApp$ ls -l
total 720
-rw-rw-r-- 1 seed seed  14443 Nov 25 00:05 audit.LOG
-rwxrwxr-x 1 seed seed    383 Nov 24 21:21 compileSETUID.sh
-rwxrwxr-x 1 seed seed    197 Nov 24 21:21 compile.sh
-rw-rw-r-- 1 seed root     18 Nov 24 21:06 config.cfg
-rwsr-xr-x 1 root seed 634584 Nov 24 22:47 directory
-rwxrwxr-x 1 seed seed    236 Nov 24 21:08 evoke.sh
-rwxrwxr-x 1 seed seed    377 Nov 24 18:51 invalidName.sh
-rwxrwxr-x 1 seed seed    539 Nov 24 20:51 invalidPhone.sh
-rw-rw-r-- 1 seed seed   1154 Nov 25 00:01 output.txt
-rwxrwxrwx 1 seed seed  13179 Nov 24 22:46 phoneBook.cpp
-rw-rw-r-- 1 seed seed  28672 Nov 25 00:05 phoneDirectory.db
-rwxrwxr-x 1 seed seed    236 Nov 24 21:08 revoke.sh
-rwxrwxr-x 1 seed seed    335 Nov 24 20:32 validName.sh
-rwxrwxr-x 1 seed seed    940 Nov 24 22:36 validPhone.sh
seed@VM:~/.../phoneListApp$
```

Running the evoke.sh command will do return this back to root privilege shown here:



```
seed@VM:~/.../phoneListApp$ evoke.sh
Files changed back to Root privileges...
seed@VM:~/.../phoneListApp$ ls -l
total 720
-rwx------ 1 root seed  14443 Nov 25 00:05 audit.LOG
-rwxrwxr-x 1 seed seed    383 Nov 24 21:21 compileSETUID.sh
-rwxrwxr-x 1 seed seed    197 Nov 24 21:21 compile.sh
-rwx------ 1 root root     18 Nov 24 21:06 config.cfg
-rwsr-xr-x 1 root seed 634584 Nov 24 22:47 directory
-rwxrwxr-x 1 seed seed    236 Nov 24 21:08 evoke.sh
-rwxrwxr-x 1 seed seed    377 Nov 24 18:51 invalidName.sh
-rwxrwxr-x 1 seed seed    539 Nov 24 20:51 invalidPhone.sh
-rw-rw-r-- 1 seed seed   1154 Nov 25 00:01 output.txt
-rwxrwxrwx 1 seed seed  13179 Nov 24 22:46 phoneBook.cpp
-rwx------ 1 root seed  28672 Nov 25 00:05 phoneDirectory.db
-rwxrwxr-x 1 seed seed    236 Nov 24 21:08 revoke.sh
-rwxrwxr-x 1 seed seed    335 Nov 24 20:32 validName.sh
-rwxrwxr-x 1 seed seed    940 Nov 24 22:36 validPhone.sh
```

Cons

- Regex values (**Whitelist for valid inputs**) are hardcoded in the code, this could possibly benefit from being read in from a file. Time constraints prevented me from implementing this feature.
- It is possible that the number of regex values could be shortened considerably **(by possibly forming more efficient regular expressions)** to reduce the size of the code
- Prepared statements were not utilized
- Including the regex header seems to slow down the compilation of the c++ file.
- There was not enough permitted time to test the program for bugs in the regular expressions or bugs that are present elsewhere in the code.
- Not enough information was known about different international phone number formatting to program proper regular expressions in every case. I attempted to match the possible valid inputs from the instructions as closely as possible.
- Many lines of code are present (a little over four hundred lines), this can possibly use some better, more efficient algorithms.

Bonus Attempt

- The first part of the bonus was attempted. I managed to use SQLite version 3 to store my input data in a database.
- Prepared statements were not implemented due to time constraints.