```
================================================
SQL Injection Lab Description
https://seedsecuritylabs.org/Labs_16.04/PDF/Web_SQL_Injection.pdf
================================================

SETUP
  0.1.  Start Apache server
        $ sudo service apache2 start
  0.2.  URL:    http://www.SEEDLabSQLInjection.com
        Folder: /var/www/SQLInjection/
  0.3.  Login with username "admin" and password "seedadmin".
        How does the site communicate to database via the its url?

+ TASK 01: MySQL Console
  1.1.  Login mysql console
        $ mysql -u root -pseedubuntu
  1.2.  Load database "Users"
        mysql> use Users;
  1.3.  View all tables in the database
        mysql> show tables;
  1.4.  View all rows in the table "credential"
        mysql> ??????
  1.5.  View info of user name "Alice"
        mysql> ??????

================================================
TURN-IN TASKS
================================================
+ TASK 02: SQL Injection attack on "SELECT" statement
  2.1.  SQL Injection attack via webpage
        - Read the code of "unsafe_home.php".
        - Where should you inject SQL code to "$sql" query (SELECT, FROM or WHERE)?
        - How would your injected code complete the query? Try it.
  2.2.  SQL Injection attack via command line
        - Recall what you observed in step 0.3
        - Can you create an attack url with the data you have in step 2.1?
        e.g., 'http://www.seed...tion.com/unsafe_home.php?<your_sql_injection_code>'
        - Make sure you use URL encoding.
        - Finally, try it with "curl" in a terminal
        $ curl 'http://www.seed...tion.com/unsafe_home.php?<your_sql_injection_code>'
  2.3.  Append a new injection string
        - Alice did not have a nickname. We'll set her nickname to
"<whoever_you_like>"
        - To do this, we need to inject an "UPDATE" command to username field.
        - Form an "UPDATE" command to update Alice's nickname to, say, "byebyeAlice"
        e.g, "UPDATE <table> SET <column>=<new_nickname> WHERE ..."
        - Try it. Do you succeed? Why/why not? What does the site complain?

+ TASK 03: SQL Injection attack on "UPDATE" statement
  3.1.  Modify Alice's salary
        - Alice wanted double salary. Boss said "no". SQL injection said "yes"
        - Which field in the "Profile Edit" was vulnerable to SQJ injection?
        - Read the code of "Profile Edit", how would you inject SQL code in "$sql"?
  3.3.  Modify Boby's salary
        - Boby wanted double salary. Boss said "yes". SQL injection said "not yet"
        - Use similar approach in 3.1 to inject your code and generously pay Boby 1c/
year.
  3.4.  Modify Boby's password
        - Inject your code and change his password to "1cPerYear".

+ TASK 04: Countermeasure - Prepared Statements
  4.1.  Make copies of "unsafe_home.php" and "unsafe_edit_backend.php".
  4.2.  Modify "unsafe_home.php" and "unsafe_edit_backend.php" using prepared
statements
```

- Hint: prepare(), bind_param(), execute(), bind_result(), fetch(), etc.

4.3. Re-test the site for SQL Injection vulnerability.