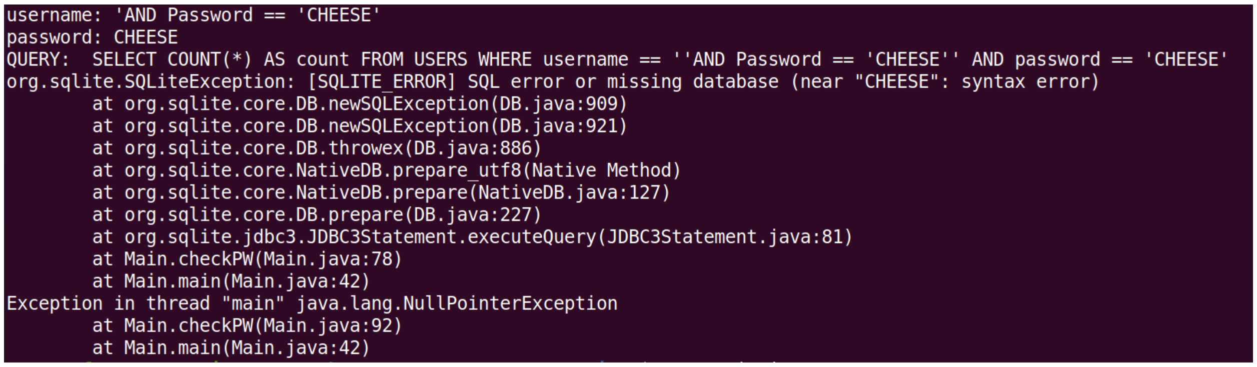
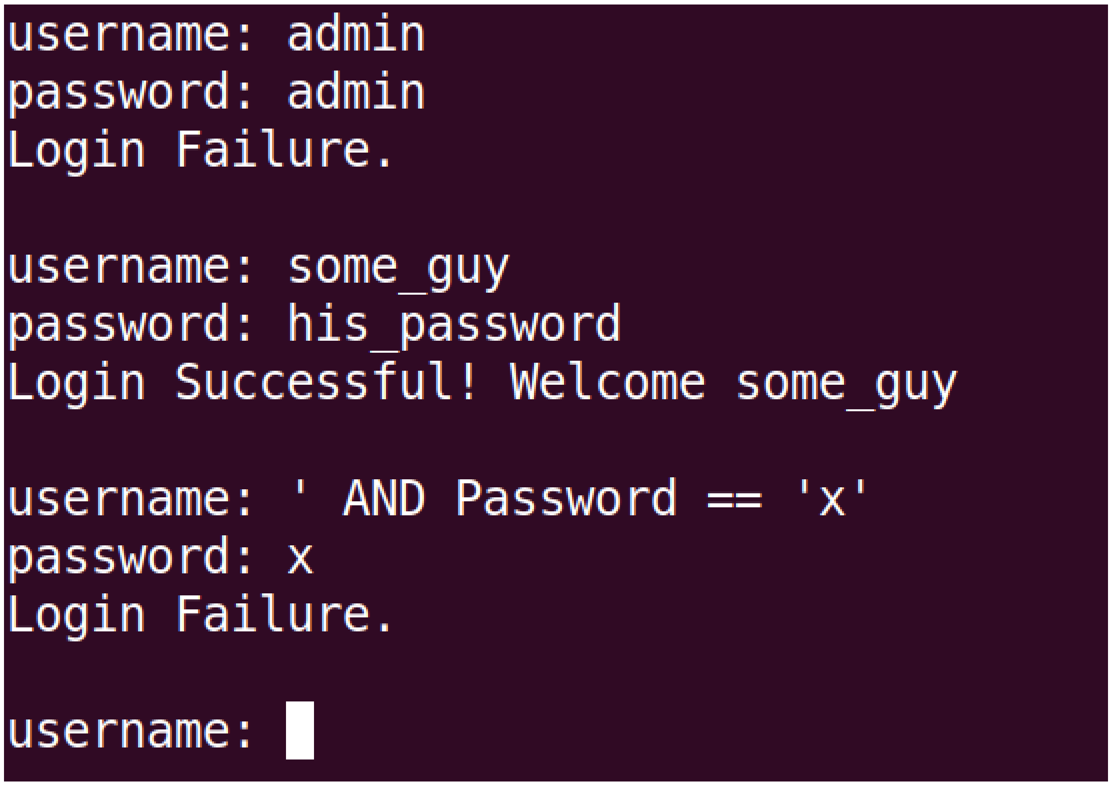


```
username: 'AND Password == 'CHEESE'
password: CHEESE
QUERY:  SELECT COUNT(*) AS count FROM USERS WHERE username == ''AND Password == 'CHEESE'' AND password == 'CHEESE'
org.sqlite.SQLiteException: [SQLITE_ERROR] SQL error or missing database (near "CHEESE": syntax error)
	at org.sqlite.core.DB.newSQLException(DB.java:909)
	at org.sqlite.core.DB.newSQLException(DB.java:921)
	at org.sqlite.core.DB.throwex(DB.java:886)
	at org.sqlite.core.NativeDB.prepare_utf8(Native Method)
	at org.sqlite.core.NativeDB.prepare(NativeDB.java:127)
	at org.sqlite.core.DB.prepare(DB.java:227)
	at org.sqlite.jdbc3.JDBC3Statement.executeQuery(JDBC3Statement.java:81)
	at Main.checkPW(Main.java:78)
	at Main.main(Main.java:42)
Exception in thread "main" java.lang.NullPointerException
	at Main.checkPW(Main.java:92)
	at Main.main(Main.java:42)
```

```
import java.io.Console;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.sql.PreparedStatement;


/**

 * Main execution class for exceptions exercise. Prompts user for username and

 * password to lookup in the accompanying sqlite3 database.

 *

 * @author Joseph Eichenhofer

 *

 */

public class Main {
	private static final String DB_URL = "jdbc:sqlite:users.db";


	/**
	 * Prompt user for username and password. Displays login success or failure
	 * based on lookup in user database.
	 *
	 * @param args
	 *            n/a
	 */
	public static void main(String[] args) {
		Console terminal = System.console();


		if (terminal == null) {
			System.out.println("Error fetching console. Are you running from an
IDE?");
			System.exit(-1);
		}


		while (true) {
			// get username and password from user
			String username = terminal.readLine("username: ");
			if (username == null || username.toLowerCase().equals("exit"))
```

```
                break;
            String password = terminal.readLine("password: ");

            // check username and password
            if (checkPW(username, password))
                System.out.println("Login Successful! Welcome " + username);
            else
                System.out.println("Login Failure.");

            // separate iterations for repeated attempts
            System.out.println();
        }
    }

    /**
     * Connect to the sample database and check the supplied username and password.
     *
     * @param username
     *            username to check
     * @param password
     *            password to check for given username
     * @return true iff the database has an entry matching username and password
```

```
 */

private static boolean checkPW(String username, String password) {

        // declare database resources

        Connection c = null;

        Statement statement = null;

        ResultSet rs = null;

    PreparedStatement pstmt = null;



        boolean success = true;



        String sqlQuery = "SELECT COUNT(*) AS count FROM USERS WHERE
username == '" + username + "' AND password == '"

                + password + "'";

        try {

            // connect to the database

            c = DriverManager.getConnection(DB_URL);

            // create outline for SQL statement that we want implemented instead of

            // allowing new Query elements into our statement, ensuring that input is
a string

        pstmt = c.prepareStatement(

            "SELECT COUNT(*) AS count FROM USERS WHERE username == ?
AND password == ?");

        pstmt.setString(1, username);

        pstmt.setString(2, password);
}

                              // check for username/password in database
                    rs = pstmt.executeQuery();
                              // if username/password not in database return false
                    if (rs.getInt("count") == 0)
                           return false;

                    } catch (SQLException ex) {
                              // sql error, debug info:
                              System.err.print("QUERY:\t");
                              System.err.println(sqlQuery);
                    System.out.println("failure in catch");
                              ex.printStackTrace(System.err);
                    }

                    // cleanup sql objects
                    try { rs.close(); } catch (SQLException ex) { System.out.println("try 1"); }
                    try { pstmt.close(); } catch (SQLException ex) { System.out.println("try 2"); }
                    try { c.close(); } catch (SQLException ex) { System.out.println("try 3"); }

                    return success;
          }
}
```



```
username: admin
password: admin
Login Failure.

username: some_guy
password: his_password
Login Successful! Welcome some_guy

username: ' AND Password == 'x'
password: x
Login Failure.

username: 
```

The attack worked by changing the SQL Query that was being used to check the username and password. When an invalid SQL Query was passed, there was a SQLException that caused the program to fail. The mitigation to this is to convert the input username and password to strings, insert them into the SQL Query in a safer manner, and then catch exceptions relating to this.

The NullPointerException that we were getting was related to the invalid SQL Query being passed. When ‘results’ was being assigned its value, that value was invalid because the Query failed to execute, resulting in the calling of ‘results’ being a NPE