Exam #2: Practice Questions Last name: **SOLN** First name:

It is a closed-notes, closed-book, no-reference and no-calculator exam. *There may be more questions than the time allows*. Attempt as many questions as possible within the allowed time of **60 minutes**. **No question may be re-visited (backtracking is disabled).** Show all your work for solving/answering problems/questions in the space provided. Do not provide multiple answers. If more than one answer is provided, the grader will only select one answer at random. Take a break (if you need to) before starting the exam. Once you start, you need to finish and submit the exam.

You may use one 8.5"x11" double-sided blank sheet and pen/pencil/eraser for scratch work.

By proceeding to take this exam, I am certifying that I am

1) alone in the room that I am taking the test in,
2) not communicating with anyone (other than the instructor) during the time of the test,
3) not using any device (such as a smartphone, a computer) other than the one that I am taking the test on,
4) not using an application other than the browser I am using to take the test through,
5) not referencing any material (such as a book, printed/electronic notes, youtube, web), and
6) not recording questions in any shape or form.

---

1) a) When RPi is connected to a laptop through VNC, where is the VNC server running on (RPi or the laptop) and where is the VNC viewer running on (RPi or the laptop)? ***(circle the answer in each case)***

Server is running on RPi (since it is serving its desktop environment to the remote user).
Viewer is running on the laptop (since the laptop is viewing something that is being served by the remote).

b) Why is it important to properly shutdown RPi using ***sudo shutdown –h 0*** command?

To properly close any open files etc., and avoid corruption of HDISK (mostly SSD in our case)

c) Why is it important to secure RPi with proper login?

To secure RPi since some of the configuration files may have plain-text passwords

d) On 000webhost.com, what is PhpMyAdmin used for?

To manage the user database

2) a) Briefly describe REST (REpresentational State Transfer) type of API:

In REST API, the user side sends a complete stateless request to the application side (with or without an API-Key... as required). The response comes back to render the designated area on the user side.

b) What is an API-Key?

In an API, API-Key is a token that is sent with a request so that the application side can check the authorization aspect of the request to restrict servicing authorized users only.

c) Show a sample XML file to describe a record in person table (pname, city, state, zip):

```
<insurance>
        <person>
                <pname> john </pname> <city> hargil </city><state> texas </state><zip>76234</zip>
        </person>
</insurance>
```

d) What is the significance of DBOR?

A Database of Record (DBoR) is the authentic repository of data that all other pieces of data take their reference from. Even if there are copies of data floating around, they all need to sync-up with the DBOR.

3)

a) What is the language (Python, SQL, PHP, HTML, etc.) used for interacting with the database on the Internet Platform, 000webhost.com, that you are using?

Currently, we are using Python on the RPi side to trigger PHP scripts on the server side using JSONRequest to send a request and JSONResponse to receive a response. The PHP script on the server uses embedded SQL to read from the DBOR and write into the DBOR.

b) Based on the way that you are using 000webhost.com, would you consider 000webhost.com as an IaaS, PaaS, or SaaS?

Our cloud is a PaaS because it provides the required *mix of infrastructure, programs and software* for us to develop a client/server application.

c) Microsoft providing online Excel is an example of (IaaS/PaaS/SaaS). ***(circle one)***

Microsoft providing Excel software in the cloud is an example of SaaS because the users would only have access to the Excel software in order to use it.

4) a) What does VNC provide?

VNC (Virtual Network Computing) is a client/server application where a machine runs the server side of the application for remote users to have access to have a *desktop-level experience*. This is different from SSH, where the remote user may be limited to a CLI window.

b) How can one have access to your UTRGV username and password if you do not secure your RPi with a login?

One can access /etc/wpa_supplicant/wpa_supplicant.conf file to see the username and password which are stored in plaintext.

c) What is the OS recommended to be used on Raspberry Pi?

Raspbian (lately, rebranded as *Raspberry Pi OS*)

d) Which module provides management of a database on 000webhost.com?

PhpMyAdmin

5) a) Even if access to an API is free, why is it a good idea to require an API-Key?

To make sure that only authorized users have access to the application.

b) Why is it a bad idea to have DBOR on a local machine?

A local machine usually does not have the level of security that a server in the cloud would have. Also, mostly a local machine is not under a backup schedule.

c) What is XML used for?

XML is a format to describe the structure of a request/data, which is sent to or received from (e.g. using SFTP) a remote machine to exchange data and/or trigger an event.

d) What is JSON used for?

JSON is used to describe data and *send request to* and *receive response from* a remote machine to exchange data and/or trigger an event.

6) Consider the following sample HTML/PHP script:

```
<?php
require_once __DIR__ . '/../required/db_connect.php';
?>
<html>
   <head>
      <title>Welcome</title>
   </head>
   <body>
   Welcome to Jane Doe's Clients Main Page </br>
   ====================================== </br>
   <?php
   if ($stmt=$mysqli->prepare("SELECT * FROM person LIMIT 100")) {
      $stmt->execute();
      $stmt->bind_result($pname,$street,$city);
      printf("Name     Street    City</br>");
      while ($stmt->fetch()) {
         echo $pname . " " . $street . " " . $city . "</br>";
      }
      $stmt->close();
   }
   else{ echo "error";
   $mysqli->close();
   }
   ?>
   =======================================</br>
   </body>
</html>
```

Using the above as a template, generate the following webpage (retrieving relevant data):

```
         Welcome to Toyota Community
   ======================================
   <names of the persons, their streets and cities, along with the model and year of their Toyota vehicles>

   =======================================
```

Below is the code with changes in ***red***:

```
<?php
require_once __DIR__ . '/../required/db_connect.php';
?>
<html>
   <head>
      <title>Welcome</title>
   </head>
   <body>
   Welcome to Toyota Community </br>
   ====================================== </br>
   <?php
   if ($stmt=$mysqli->prepare("SELECT * FROM person,vehicle where person.pname=vehicle.pname and
model='toyota' LIMIT 100")) {
      $stmt->execute();
      $stmt->bind_result($pname,$street,$city,$make,$year);
      printf("Name     Street    City     Make   Year</br>");
      while ($stmt->fetch()) {
           echo $pname . "  " . $street . "  " . $city . "  " . $make . "  " . $year . "</br>";
      }
      $stmt->close();
   }
   else{ echo "error";
   $mysqli->close();
   }
   ?>
   ========================================</br>
   </body>
</html>
```

7)
Consider the following Python program as a reference:

```
#!/usr/bin/python
import RPi.GPIO as GPIO                  #import GPIO library
import time
GPIO.setmode(GPIO.BCM)                   #set the pins according to BCM scheme
GPIO.setup(4,GPIO.OUT)                   #configure BCM Pin #4 as OUTPUT
GPIO.setup(17,GPIO.IN)                   #configure BCM Pin #17 as INPUT
print "Powering LED ON... BCM pin 4"
GPIO.output(4,GPIO.HIGH)                 #set BCM Pin #4 to 1
time.sleep(2)                            #wait
print "Powering LED OFF... BCM pin 4"
GPIO.output(4,GPIO.LOW)                  #set BCM Pin #4 to 0
sw=GPIO.input(17)                        #read the status of BCM Pin #17
if sw==1: print "Switch is ON... BCM pin 17"
else: print "Switch is OFF... BCM pin 17"
GPIO.cleanup()                           #set BCM pins to default for next time
```

Based on the above, write a Python program to set LED connected to BCM Pin #22 to ON if the switch connected to BCM PIN #27 is OFF, otherwise, LED is set to OFF.

Below is the program according to the specifications:

```
#!/usr/bin/python
import RPi.GPIO as GPIO                  #import GPIO library
import time
GPIO.setmode(GPIO.BCM)                   #set the pins according to BCM scheme
GPIO.setup(22,GPIO.OUT)                  #configure BCM Pin #22 as OUTPUT
GPIO.setup(27,GPIO.IN)                   #configure BCM Pin #27 as INPUT

sw=GPIO.input(27)                        #read the status of BCM Pin #27
if sw==0:
    print "Switch is OFF... Turning ON LED"
    GPIO.output(22,GPIO.HIGH)
else:
    print "Switch is ON... Turning OFF LED"
    GPIO.output(22,GPIO.LOW)
GPIO.cleanup()                           #set BCM pins to default for next time
```

8)
Using the program given earlier as a reference, write a Python program to blink the LED connected to BCM Pin #22 if the switch connected to BCM PIN #27 is ON, otherwise, LED is set to OFF. The blinking interval time is 3 seconds (both on time and off time).

```
#!/usr/bin/python
import RPi.GPIO as GPIO                  #import GPIO library
import time
GPIO.setmode(GPIO.BCM)                   #set the pins according to BCM scheme
GPIO.setup(22,GPIO.OUT)                  #configure BCM Pin #22 as OUTPUT
GPIO.setup(27,GPIO.IN)                   #configure BCM Pin #27 as INPUT

while true:
        sw=GPIO.input(27)                        #read the status of BCM Pin #27
        if sw==1:
                GPIO.output(22,GPIO.HIGH)
                time.sleep(3)                    #blink LED if the switch is ON
                GPIO.output(22,GPIO.LOW)
                time.sleep(3)
        else:
                GPIO.output(22,GPIO.LOW)
GPIO.cleanup()                           #set BCM pins to default for next time
```

9)

a) Briefly explain what the following statements accomplish:

```
$login_check = hash('sha512', $password . $user_browser);
if (hash_equals($login_check, $login_string)) {
    return true;
}
```

The above statements are used in preventing hijacking of a session. The first statement takes the concatenation of the hashed password ($password) and the current browser type ($user_browser), and creates a hashed value and stores it in $login_check. The second statement compares this hashed value ($login_check) with the original corresponding hashed value ($login_string) created when the original user started the session using a certain browser. If there is a mismatch, then the user of the current browser is assumed to be a hacker, and the session is disconnected. *{Note that by browser type we mean $_SERVER['HTTP_USER_AGENT'] which is unique for each visitor. It does not mean that if both the original user and the hacker are using firefox, the hacker would be able to hijack the session.}*

This is covered under the "Tutorial_Intro_to_SecuringWebsites" available under ***Reference Material*** folder on Blackboard.

b) Add php code within the while statement (in the space provided) to print (on the website) persons from "harlingen" and also the persons living on the street named "freedom" in any city.

```
<?php
require_once __DIR__ . '/../required/db_connect.php';
?>
<html>
  <head>
    <title>Welcome</title>
  </head>
  <body>
  ======================================== </br>
  <?php
  if ($stmt=$mysqli->prepare("SELECT * FROM person LIMIT 100")) {
    $stmt->execute();
    $stmt->bind_result($pname,$street,$city);
    printf("Name</br>");
    while ($stmt->fetch()) {

        if ($city=='harlingen' || $street=='freedom') {
                echo $pname . "  " . $street . "  " . $city . "</br>";
        }

    }
    $stmt->close();
  }
  else{ echo "error";
  $mysqli->close();
  }
  ?>
  ========================================</br>
  </body>
</html>
```

10) Consider the following Python program as a reference:

```
#!/usr/bin/python
import RPi.GPIO as GPIO
import time
GPIO.setmode(GPIO.BCM)
GPIO.setup(4,GPIO.OUT)
GPIO.setup(17,GPIO.IN)
print "Powering LED ON... BCM pin 4"
GPIO.output(4,GPIO.HIGH)
time.sleep(2)
print "Powering LED OFF... BCM pin 4"
GPIO.output(4,GPIO.LOW)
sw=GPIO.input(17)
if sw==1: print "Switch is ON... BCM pin 17"
else: print "Switch is OFF... BCM pin 17"
GPIO.cleanup()
```

Based on the above, consider that an LED is connected to BCM Pin #22, and a switch connected to BCM Pin #27. Write a Python program to power ON the LED for 2 seconds and power OFF the LED for 2 seconds, if the switch is ON. However, if the switch is OFF, the LED should be powered ON for 5 seconds and powered off for 5 seconds. The checking of the switch and powering ON/OFF of the LED must be done in a continuous loop (forever).

```
#!/usr/bin/python
import RPi.GPIO as GPIO                 #import GPIO library
import time
GPIO.setmode(GPIO.BCM)                  #set the pins according to BCM scheme
GPIO.setup(22,GPIO.OUT)                 #configure BCM Pin #22 as OUTPUT
GPIO.setup(27,GPIO.IN)                  #configure BCM Pin #27 as INPUT

while true:
        sw=GPIO.input(27)                       #read the status of BCM Pin #27
        if sw==1:
                GPIO.output(22,GPIO.HIGH)
                time.sleep(2)                   #blink LED (delay= 2 seconds)
                GPIO.output(22,GPIO.LOW)
                time.sleep(2)
        else:
                GPIO.output(22,GPIO.HIGH)
                time.sleep(5)                   #blink LED  (delay= 5 seconds)
                GPIO.output(22,GPIO.LOW)
                time.sleep(5)

GPIO.cleanup()                          #set BCM pins to default for next time
```

11)

a) Answer the following:

i) Is $user_id an integer or a string?

It really depends on the context. In php, it is not necessary to initialize variables. If $user_id was set to a string, then it is a string. If it was set to an integer, then it is a string.

ii) What does hashing of the combination of $password and $user_browser accomplish in **$login_check = hash('sha512', $password . $user_browser)** statement?

The above statement is used in preventing hijacking of a session. The statement takes the ***concatenation*** of the *hashed password* ($password) and the current *browser type* ($user_browser), and creates a hashed value and stores it in *$login_check*. The environment has the original hashed value at the time when the original user logged in using a certain browser. *{Note that by browser type we mean $_SERVER['HTTP_USER_AGENT'] which is unique for each visitor. It does not mean that if both the original user and the hacker are using firefox, the hacker would be able to hijack the session.}*

In another statement (not shown here), this hashed value ($login_check) is compared with the original corresponding hashed value (e.g. $login_string). If there is a mismatch, then the user of the current browser is assumed to be a hacker, and the session is disconnected.

This is covered under the "Tutorial_Intro_to_SecuringWebsites" available under ***Reference Material*** folder on Blackboard.

b) Given the sample code at the end of the exam, write a Python code to implement the following:

-LED connected to Pin #17 blinks every 2 seconds if SW1 (connected to Pin #22) is OFF. Otherwise, the LED remains OFF.

-LED connected to Pin #23 blinks every second if SW1 (connected to Pin #22) is ON. Otherwise, the LED remains OFF.

```
#!/usr/bin/python
import RPi.GPIO as GPIO              #import GPIO library
import time
GPIO.setmode(GPIO.BCM)               #set the pins according to BCM scheme
GPIO.setup(17,GPIO.OUT)              #configure BCM Pin #17 as OUTPUT
GPIO.setup(23,GPIO.OUT)              #configure BCM Pin #23 as OUTPUT
GPIO.setup(22,GPIO.IN)               #configure BCM Pin #22 as INPUT

while true:
        sw=GPIO.input(22)            #read the status of BCM Pin #22
        if sw==0:
                GPIO.output(17,GPIO.HIGH)
                time.sleep(2)                   #blink LED (delay= 2 seconds)
                GPIO.output(17,GPIO.LOW)
                time.sleep(2)
        else:
                GPIO.output(23,GPIO.HIGH)
                time.sleep(1)                   #blink LED  (delay= 1 second)
                GPIO.output(23,GPIO.LOW)
                time.sleep(1)

GPIO.cleanup()                               #set BCM pins to default for next time
```

12)

Assume the following:

-RPi has the following table (primary key underlined) in MySQL: persons (**pname**, street, city, salary)

-RPi needs to send the salary of Kochab to the server https://approve.hello.org using JSONRequest

-The server responds with OK or NOT_OK as a response to a successful interaction

-If RPi receives OK from the server, it prints "Kochab's salary approved."

-If RPi receives NOT_OK from the server, it prints "Kochab's salary out of range."

Given the sample code at the end of the exam, write a Python code on the RPi side to implement the above.

```
#!/usr/bin/python
import requests                          #import JSONRequests library
import time                              #import time library for sleep function

import MySQLdb
db = MySQLdb.connect(host="localhost", user="user",passwd="pass",db="insurance")
cur = db.cursor()
cur.execute("select salary from persons where pname='kochab'")
n = cur.rowcount
If n==1:
        salary = cur.fetchone()
        data = {'username': 'ben', 'password': 'benpass', 'SAL': salary, 'EMP': 'KOCHAB'}
        res = requests.post("https://approve.hello.org/scripts/approve_salary.php", json=data)
        r = res.json()
        ts = datetime.datetime.now()          #get the time stamp
        print "==============Server Response at " + str(ts) + "=============="
        if r['success']==1:
                if r['approval']=='OK':
                        print "KOCHAB's salary approved."
                else:
                        print "KOCHAB's salary out of range."
        else:
                print ">>>>> There was a problem in processing your request - Error #" + str(r['error'])
else:
        print "DB either provided no results, or too many.  Request was not sent to the server."
```

13) Briefly describe the following:

a) XML:

XML is a format to describe the structure of a request/data, which is sent to or received from (e.g. using SFTP) a remote machine to exchange data and/or trigger an event.

b) JSONRequest:

JSON is used to describe data and *send request to* and *receive response from* a remote machine to exchange data and/or trigger an event.

c) Significance of hashing a password:

A hacker should not be able to extract the original password given the hashed value of the password. That is why the passwords of the users of a system are stored as hashed values in a DB. Even if these are compromised, it is not considered a major loss, even though the breach itself should be concerning.

d) A sensor:

A sensor generates a value and, provides this value to the system that it is interfaced with. The generated value may be a single bit value (e.g. a binary sensor such as a switch, water level sensor, flood sensor, and light sensor) or n-bit value (e.g. an analog sensor being digitized by an n-bit Analog to Digital Converter – ADC – such as a sensor for measuring sound level, light level, temperature, pressure, acceleration, and speed).

e) An Actuator:

An actuator does the prescribed action when signaled to do so. For example, sending a "start" signal to the starting mechanism/actuator of a vehicle would start the vehicle. As another example, sending an "open-door" signal to the open/door mechanism/actuator would open the door. There are *layers of abstraction* involved in these statements. For example, the real actuator in a traditional vehicle is the solenoid that engages the flywheel and spins it to start the vehicle. So, the "starting mechanism" may have multiple entities in it but the real actuator part that does the real motion/action could be a small part of it. Some other examples are:

-Stepper motor
-Open/close a valve (water, gas, oil)
-Relay to turn on/off a device
- . . .

Sample Python Code:

```
#!/usr/bin/python
        import MySQLdb
        db = MySQLdb.connect(host="localhost", user="user",passwd="pass",db="insurance")
        cur = db.cursor()
        cur.execute("select * from person")
        n = cur.rowcount
        print  "pname\t"  + "street\t"  + "city\t"
        print  "======\t"  + "=====\t"  + "====\t"
        for r in range (0,n):
                row = cur.fetchone()
                if row[2]== "harlingen":
                        print row[0]  +  "\t"  + row[1]  + "\t"  + row[2]
```

. . .

```
#!/usr/bin/python
import requests                         #import JSONRequests library
import time                             #import time library for sleep function
import datetime                         #import datetime library for timestamp
import RPi.GPIO as GPIO                 #import GPIO library
GPIO.cleanup()
GPIO.setmode(GPIO.BCM)                  #set the pins according to BCM scheme
GPIO.setup(4,GPIO.OUT)                  #configure BCM Pin #4 as OUTPUT
GPIO.setup(17,GPIO.IN)                  #configure BCM Pin #17 as INPUT
i=0; n=5; delay=20                      #limit number of tries to 5 (initially set it to 1 for debugging)
while i<n:
        LED1=GPIO.input(4)                      #read what BCM Pin #4 is set to (LED1)
        SW1=GPIO.input(17)                      #read the status of BCM Pin #17 (SW1)
        data = {'username': 'ben', 'password': 'benpass', 'SW1': SW1, 'LED1': LED1}
        res = requests.post("https://yourdomain.000webhostapp.com/scripts/sync_rpi_data.php", json=data)
#in case of errors (especially, syntax) , you may want to print res.text and comment out the statements below
        r = res.json()
        ts = datetime.datetime.now()            #get the time stamp
        print "==============Server Response at " + str(ts) + "=============="
        if r['success']==1:
        print "+++++Server request successful: "
                if LED1!=r['LED1']:
                        print "Changing LED status as requested by the server"
                        if r['LED1']==1:
                                GPIO.output(4,GPIO.HIGH)
                        else: GPIO.output(4,GPIO.LOW)
                        print "The status of LED1 is " + str(r['LED1'])
                        print "The status of SW1 is " + str(r['SW1'])
        else: print ">>>>> Server request failed - Error #" + str(r['error'])
        time.sleep(delay)               #wait for delay seconds before sending another request
        i+=1
GPIO.cleanup()
```

Sample PHP Code:

```php
<?php
require_once __DIR__ . '/../../required/db_connect.php';
$input = file_get_contents("php://input");
$error=0; $out_json = array(); $out_json['success'] = 1; //assume success
$SW1_status=0; $LED1_status=0;
if ($input) {
    $json = json_decode($input, true);       //check if it json input
    if (json_last_error() == JSON_ERROR_NONE) {
        if (isset($json["username"]) && isset($json["password"]) && isset($json["SW1"])
               && isset($json["LED1"])) {
            $in_username = $json["username"];
            $in_password = $json["password"];  //if the expected fields are not null, get them
            $in_SW1 = $json["SW1"];
            $in_LED1 = $json["LED1"];
            if ($stmt=$mysqli->prepare("SELECT password FROM webuser WHERE pname = ? LIMIT 1")) {
                $stmt->bind_param('s', $in_username);
                $stmt->execute();  $stmt->store_result();   //store_result to get num_rows etc.
                $stmt->bind_result($db_password);    //get the hashed password
                $stmt->fetch();
                if ($stmt->num_rows == 1) {          //if user exists, verify the password
                    if (password_verify($in_password, $db_password)) {
                        $stmt->close();
                        if ($stmt = $mysqli->prepare("UPDATE device set status=?
                                                      where devname = 'SW1'")) { //update LED1
                          $stmt->bind_param('i', $in_SW1); $stmt->execute();
                        } else {$error=1;}
                        $stmt->close();
                        if (!$error && ($stmt = $mysqli->prepare("SELECT status FROM device
                                                      where devname = 'SW1'"))) { //read SW1
                          $stmt->execute(); $stmt->bind_result($SW1_status); $stmt->fetch();
                        } else {$error=2;}
                        $stmt->close();
                        if (!$error && ($stmt = $mysqli->prepare("SELECT status FROM device
                                                      where devname = 'LED1'"))) { //read LED1
                          $stmt->execute(); $stmt->bind_result($LED1_status); $stmt->fetch();
                        } else {$error=3;}
                        $stmt->close();
                    } else {$error=4;}
                } else {$error=5;}
            } else {$error=6;}
        } else {$error=7;}
    } else {$error=8;}
} else {$error=9;}
if ($error){
   $out_json['success'] = 0;  //flag failure
}
$out_json['SW1'] = $SW1_status; $out_json['LED1'] = $LED1_status;
$out_json['error'] = $error;  //provide error (if any) number for debugging
echo json_encode($out_json);  //encode the data in json format
?>
```