Part A: HTTP

The following are the HTTP headers of a request sent by a web browser, and the corresponding response from the server.

Request

```
GET /latest/image HTTP/1.1
Host: www.image-host.org
Connection: keep-alive
Pragma: no-cache
Cache-Control: no-cache
Accept: image/webp,image/*,*/*;q=0.8
User-Agent: Mozilla/5.0 (Windows NT 6.1; WOW64)
Referer: http://www.myblog.net/
Accept-Encoding: gzip, deflate, sdch
Accept-Language: en-AU,en;q=0.8,en-US;q=0.6
```

Response

```
HTTP/1.1 200 OK
Date: Tue, 22 Sep 2015 02:56:14 GMT
Server: Apache/2
Last-Modified: Wed, 15 Dec 2010 09:53:45 GMT
Accept-Ranges: bytes
Content-Length: 4764
Cache-Control: max-age=2592000
Expires: Thu, 22 Oct 2015 02:56:14 GMT
Content-Type: image/jpeg; qs=0.8
```

Question 1 – HTTP Protocol

Examine the above HTTP request and response headers, and identify:

i. The HTTP version used version 1.1

ii. The full URL of the request, including the protocol and host name. http://www.myblog.net

iii. The URL the browser loaded that triggered this request. http://www.myblog.net/

iv. The HTTP status code returned by the server 200

v. What type of content the response will contain image/jpeg

[(1 + 1 + 1 + 1 + 1) = 5 marks]

Question 2 – Client-Server Responsibilities
Indicate for each of the following activities whether it should occur at the *server-side* or the *client-side.*

i. Applying CSS to style the document Client Side
ii. Connecting to a database to generate HTML Client Side
iii. Handling a "click" event to prevent submission of a form Client Side
iv. Password hashing Server Side
v. Storing sensitive session data Server Side
vi. Saving and clearing cookies Client Side

[(½ + ½ + ½ + ½ + ½ + ½) = 3 marks]

Part B: PHP and Forms

Question 3 - Arrays

Examine the following PHP code:

```php
<?php

    $ar = array('a' => 'x', 'b' => 'y', 'x' => 'z');

    $first = $ar['b'];
    $second = isset($ar['z']);
    $third = $ar[$ar['a']];
    $fourth = count($ar);

?>
```

On running this code, what values are in **$first, $second**, **$third** and **$fourth**?

[(1 + 1 + 1 + 1) = 4 marks]

$first = y
$second = null
$third = z
$fourth = 3

Question 4 – Scope

Inspect the following PHP code:

```php
<?php

$x = "leonardo";
$y = "da";
$z = "vinci";

function printIt($x) {
    global $z;
    $y = "the";
    $z = "turtle";

    echo "$x $y $z";
}

printIt($z);
```

Answer the following questions:

i. What text is the output of the above code? vinci the turtle

ii. At the end of this code, what values remain in the *global* **$x**, **$y** and **$z** variables? $x = leonardo $y = da $z =vinci

iii. How do **superglobal** variables differ in usage from **global** variables?

iv. Name **3** superglobal variables $GLOBAL, $_SERVER, $_GET

[(1 + 1 + 1 + 1) = 4 marks]

Question 5 – Forms and Sessions

Inspect the following PHP and HTML code:

```
<?php
  session_start();
  $help = "Write more";

  if(!isset($_SESSION['todos'])) {
    $_SESSION['todos'] = array();
    $help = "Write something";
  }

  if(isset($_POST['todo'])) {
    array_push($_SESSION['todos'], $_POST['todo']);
  }
?>
<!DOCTYPE html>
<html>
<head><title>Todo</title></head>
<body>
<h1>Todo List - <?php echo $help; ?></h1>
  <ul>
  <?php foreach ($_SESSION['todos'] as $msg) {
    echo "<li>" . $msg . "</li>";
  } ?>
  </ul>
  <form action="" method="POST">
    <input type="text" name="todo">
    <button>Post a todo</button>
  </form>
</body>
</html>
```

Answer the following questions:

i. Explain the purpose of the following **if** statement, and what effect it has:

```
if(isset($_POST['todo']))
```

To check form input value of todo is set or not on submitting form

ii. What data type is being stored in `$_SESSION['todos']`? Array

iii. Sketch what is displayed when a user enters the text "*Get Milk*" into the input element and clicks the button. Include all headings and form elements

Heading = Todo List - Write more
Li element = Get Milk
Form element will be same

iv. This page is vulnerable to both **XSS and CSRF attacks**. Give one way that each of these can be mitigated, and justify your answer making reference to the code above.

[(1 + 1 + 2 + 2) = 6 marks]

Question 6 – OO PHP

The following code declares two classes which describe a data model for a local Burger shop point-of-sale system. Additionally, it creates an instance of a **"Tasty Burger"** which has four ingredients.

```php
<?php

class Burger {
    public $title = '';
    private $ingredients = array();

    public function __construct($n) {
        $this->name = $n;
    }

    public function addIngredient($ing) {
        array_push($this->ingredients, $ing);
    }

    public function getCost() {
        // TODO: write an implementation
    }
}

class Ingredient {
    public $name = 'Ingredient';
    public $costDollars = 0.0;

    public function __construct($n, $c) {
        $this->name = $n;
        $this->costDollars = $c;
    }
}

$myBurger = new Burger('Tasty Burger');
$myBurger->addIngredient(new Ingredient('Meat', 0.3));
$myBurger->addIngredient(new Ingredient('Cheese', 0.2));
$myBurger->addIngredient(new Ingredient('Beetroot', 0.2));
$myBurger->addIngredient(new Ingredient('Pineapple', 0.4));

echo $myBurger->getCost();
?>
```

Using a **foreach** loop, write an implementation of **Burger::getCost** which sums the cost of all the ingredients, and **returns it**.

[2 marks]

```
public function getCost() {
        $total_cost = 0;

        foreach($this->ingredients as $i){
                $total_cost += $i->costDollars;
        }
        return $total_cost;

    // TODO: write an implementation
        }
}
```

Part C: PHP and MySQL

Question 7 - MySQLli

The **cheesedb** database has the following schema:

Table: **cheese**	
id	Integer, primary key
name	Varchar(32)
hardness	Integer
manufacturer_id	Integer, foreign key

Table: **manufacturer**	
id	Integer, primary key
name	Varchar(32)
address	Varchar(128)

Examine the following PHP script, **cheese.php** that uses the MySQLi API to query the **cheesedb** database running on **localhost**, and answer the following questions (next page). The username is **cheeser** and the password is **ch33s3. The lines associated with questions *i, ii, iii* and *iv* are indicated using comments.**

```
<!DOCTYPE html>
<html><head><title>Cheeses</title></head>
<body>
  <ul>
  <?php
    $conn = mysqli_connect(/* ... */);                    // i)
    $hard = $_GET['hard'];                                // ii)

    $query = "SELECT name FROM cheese WHERE " .
             " hardness=?";                               // iii)
    $stmt = mysqli_prepare($conn, $query);
    mysqli_stmt_bind_param($stmt, "i", $hard);
    mysqli_stmt_execute($stmt);

    $results = mysqli_stmt_get_result($stmt);
    while($row = mysqli_fetch_assoc($results))
    {
      echo "<li>";
      /*  */                                              // iv)
      echo "</li>";
    }
    mysqli_close($conn);
  ?>
  </ul>
</body>
</html>
```

i. What are the **four arguments** required by mysqli_connect?

$conn = mysqli_connect('localhost', 'cheeser', 'ch33s3', 'cheesedb');

ii. Construct a URL which will display a page listing cheeses with a hardness of **6**.

localhost/cheese.php?hard=6

iii. Explain the purpose of the **?** (question mark) character in the SQL query.

This prevents injection of harmful codes and only pass integer value from get variable to query

iv. Write a line of PHP code that will display the **name** field for each cheese.

$row['name']

[(1 + 1 + 1 + 1) = 4 marks]

Part D – JavaScript

Question 8

The following HTML and JavaScript code implements a linear search through **items** based on some condition:

```
<!DOCTYPE html>
<html><head><title>Array Find</title></head>
<body><script>
    var items = ['John', 'Paul', 'George', 'Ringo'];

    function findMatch(matchFn) {
        var match = null;
        for (var i = 0; i < items.length; i++) {
            var item = items[i];

            if( matchFn( item )) {
                match = item;
                break;
            }
        }
        return match;
    }

    var callback = function(item) {
        return (item.length > 4); // greater than 4
    };

    var lookupResults = {
        'gt': findMatch(callback)
    };

    console.log(lookupResults.gt);
</script></body></html>
```

Analyse the above code, and answer the following questions:

i. What is the effect of the `break;` statement? When the condition is matched, it breaks the for loop and it stops checking further item and provides the matched result

ii. What is the console output when the page is loaded in a browser? George

iii. Is `findMatch` an anonymous or named function? named function

iv. Which JavaScript **data types** are stored in the following variables during the execution of the script?

- `items` Array
- `lookupResults` object
- `callback` function
- `lookupResults.gt` string

[(1 + 1 + 1 + (½ + ½ + ½ + ½)) = 5 marks]

Question 9

In a **browser**, which of the following objects contains all declared global variables? (choose one):

A. `document`

B. `window`

C. `HTMLElement`

D. `global`

E. `root`

Answer - D

[1 mark]

Question 10

Function objects in JavaScript have a *prototype* property. Explain how this property is used in object-oriented JavaScript programming, including its role in inheritance and its relationship to object prototypes.

[2 marks]

Part E – XML and Document Object Model

Question 11

Is the following XML document well-formed? Justify your answer

```
<?xml version="1.0" encoding="utf-8"?>
<users>
    <user id="10">Graeme Garden</user>
    <user id="11">Tim Brooke-Taylor</user>
    <user id="12">Bill Oddie</user>
</users>
<friendships>
    <pair first="10" second="12" />
</friendships>
```

users and friendships must be wrapped in a single xml tags so given xml document is not well-formed

[2 marks]

Question 12

The following XML document contains information about articles for an online publication, *The Wigan Gazette*.

```
<?xml version="1.0" encoding="utf-8"?>
<articles>
  <article id="pest_control" views="621">
    <headline>Amazing new pest control technique</headline>
    <author>Lady Tottington</author>
    <categories>
      <category>Science</category>
    </categories>
  </article>
  <article id="cheese_the_worst" views="120">
    <headline>Cheese: the worst?</headline>
    <author>Wendolene Ramsbottom</author>
    <categories>
      <category>Science</category>
      <category>Lifestyle</category>
    </categories>
  </article>
  <article id="amazing_baguette" views="791">
    <headline>Amazing baguette recipe</headline>
    <author>Piella Bakewell</author>
    <categories>
      <category>Lifestyle</category>
    </categories>
  </article>
</articles>
```

Write **single DOM expressions**, using **either** PHP or JavaScript, which will return the following specified DOM nodes or values, without using loops.

The document is stored in a variable called **doc** or **$doc**

example:

*Return the **articles** element:*
doc.getElementsByTagName('articles')[0] // JavaScript
or
$doc->getElementsByTagName('articles')->item(0) // PHP

i. Return the first **author** element $first_author = (string)$xml->article[0]->author ;

ii. Return the **text value** of the headline of the article with id **amazing baguette**

iii. Return the number of views of the first article $first_article_views = ((integer)$xml->article[0]['views']);

iv. Return a NodeList of **all headline elements**

v. Return the first **article element** that contains the Lifestyle category

[(1 + 1 + 1 + 1 + 1) = 5 marks]

Part F – APIs

Question 13

An XMLHttpRequest object allows dynamic HTTP requests to be triggered from JavaScript. It allows the use of both **synchronous** and **asynchronous** requests.

Explain the difference between **synchronous and asynchronous** requests, including the user interface impact of using synchronous requests.

[2 marks]

Question 14

The following JSON is the response to a GET request for a resource at http://api.example.com/items/ and represents a collection of in-app purchase items for a game.

```
{
    "items": [
        {
            "id": 1,
            "title": "Pudding Upgrade"
        },
        {
            "id": 2,
            "title": "Skill: Ecky Thump"
        }
    ]
}
```

i. What are two limitations of representing data in JSON compared to JavaScript?

ii. One of the principals of REST is HATEOAS – Hypermedia As The Engine Of Application State.
Adapt the above JSON API response to follow HATEOAS for each item resource. You may use any resource identifier structure you like.

[(2 + 3) = 5 marks]

END OF EXAMINATION