ISIT 307 Web Server Programming

Table of content

Introduction (used in other questions)

1. Create PHP scripts
2. Create PHP code blocks
    a. <?php ?>
3. Work with variables and constants
    a. $variable_name = value; // variable needs to use $ to call
    b. define("num", value); // constant does not need $ to call
    c. number_format($num, 2); // format to 2dp
4. Study data types
    a. Strongly typed programming languages - requires user to declare the data type of variable
    b. Static or strong typing refers to data type that does not change after declare
    c. Loosely typed programming languages do not require you to declare the data types of variables
    d. Dynamic or loose typing refers to data types of the variable that can change after they have been declared
    e. Numeric Data Types
        i. Php supports two numeric data types:
            1. Integer
            2. Floating-point number is a number that contains decimal places or that is written in exponential notation(shortened format for writing very large numbers).
    f. Boolean value
    g. Declaring and Initializing Indexed Arrays
        i. $Class = array("John", "Tom", "Mary");
        ii. $Class[ ] = “John”
        $Class[ ] = “Mary”
        $Class[ ] = “Jim”
5. Use expression and operators
    a. Expression - literal value or variable that can be evaluated by the PHP scripting to produce a result
    b. Operands - variable and literals contained in an expression
    c. Literal - static value such as literal contained in an expression
    d. Operators - symbols (+)(*) that are used in expressions to manipulate operands
        i. Binary operator requires an operand before and after the operator
        ii. Unary operator requires a single operand either before or after operator
        iii. Arithmetic operators are used in PHP to perform mathematical calculations (+ - x ÷)
        iv. Prefix operator ++
        v. Postfix operator --

vi.

Symbol	Operation	Description
=	Assignment	Assigns the value of the right operand to the left operand
+=	Compound addition assignment	Adds the value of the right operand to the value of the left operand and assigns the new value to the left operand
-=	Compound subtraction assignment	Subtracts the value of the right operand from the value of the left operand and assigns the new value to the left operand
*=	Compound multiplication assignment	Multiplies the value of the right operand by the value of the left operand and assigns the new value to the left operand
/=	Compound division assignment	Divides the value of the left operand by the value of the right operand and assigns the new value to the left operand
%=	Compound modulus assignment	Divides the value of the left operand by the value of the right operand and assigns the remainder (modulus) to the left operand

Symbol	Operation	Description
==	Equal	Returns TRUE if the operands are equal
===	Strict equal	Returns TRUE if the operands are equal and of the same data type
!= or <>	Not equal	Returns TRUE if the operands are not equal
!==	Strict not equal	Returns TRUE if the operands are not equal or not of the same data type
>	Greater than	Returns TRUE if the left operand is greater than the right operand
<	Less than	Returns TRUE if the left operand is less than the right operand
>=	Greater than or equal to	Returns TRUE if the left operand is greater than or equal to the right operand
<=	Less than or equal to	Returns TRUE if the left operand is less than or equal to the right operand

vii. <=> The **rocket ship operator** (available in PHP7+)

viii. Conditional operator - conditional expression ? a : b;

```
$BlackjackPlayer1 = 20;
($BlackjackPlayer1 <= 21) ?
   $Result = "Player 1 is still in the game." :
   $Result = "Player 1 is out of the action.";
 echo "<p>", $Result, "</p>";
```

ix. Logical Operators are used for comparing two Boolean operands for equality

Symbol	Operation	Description
&& or AND	Logical And	Returns TRUE if both the left operand and right operand return a value of TRUE; otherwise, it returns a value of FALSE
|| or OR	Logical Or	Returns TRUE if either the left operand or right operand returns a value of TRUE; otherwise (neither operand returns a value of TRUE), it returns a value of FALSE
XOR	Logical Exclusive Or	Returns TRUE if only one of the left operand or right operand returns a value of TRUE; otherwise (neither operand returns a value of TRUE or both operands return a value of TRUE), it returns a value of FALSE
!	Logical Not	Returns TRUE if an expression is FALSE and returns FALSE if an expression is TRUE

x. Type casting

1. $NewVariable = (new_type) $OldVariable;

Symbol	Operation
[and]	Accesses an element of an array
=>	Specifies the index or key of an array element
,	Separates arguments in a list
? and :	Executes one of two expressions based on the results of a conditional expression
instanceof	Returns TRUE if an object is of a specified object type
@	Suppresses any errors that might be generated by an expression to which it is prepended (or placed before)
(int), (integer), (bool), (boolean), (double), (string), (array), (object)	Casts (or transforms) a variable of one data type into a variable of another data type

xi. GetType() function

1. is_int($a), is_string($a)

e. Operator precedence refers to the order in which operations in an expression are evaluated

f. Associativity is the order in which operators of equal precedence execute and is evaluated on a left to right or a right to left basis

Functions and Control Structures (coding and things that are implemented in class)

1. Learn how to use functions to organise the PHP code
    a. <?php
    function name_of_function(parameters) {
     statements;
    }
    ?>
    b. Returning arrays -
    function multi_calc ($n1, $n2, $n3)
    {
    $sum = $n1+$n2+$n3; //$res[] = $n1+$n2+$n3;
    $prod = $n1*$n2*$n3; //$res[] = $n1*$n2*$n3;
    return array($sum, $prod); //return $res;
    }
    $result = multi_calc(5, 6, 7); echo "Results are: ", $result[0], " and ", $result[1]
    c. Function parameter by value or reference
        i. Value
            1. Does not change original value that is passed in
            function IncrementByValue($CountByValue) {
        ii. Reference
            1. Changes original value that is passed in
            function IncrementByReference(&$CountByReference) {
    d. function sum(...$num) {
            $acc = 0;
            foreach ($num as $n) {
                $acc += $n;
            }
            return $acc;
    }
    $sum = sum (1, 10, 100);
    echo “sum = $sum”;
    e. function makecoffee($type = "cappuccino") - specifying
    f. <?php
            declare (strict_types=1);
            function double_num (int|float $number) : int|float {
                Return $number *= 2;
            }
            $num = 4.8;
            echo ‘$num =’, $num, ‘
’;
            echo ‘double_num returns ‘, double_num($num), ‘
’;
    ?>
2. Learn about variable scope
    a. Variable scope is where in your program a declared can be used

    b. Global variable is one that is declared outside a function and is available to all parts of your program
    ```
    $globalvar = “this”
    function globalVar() {
            global $globalVar;
    }
    ```
    c. Local variable is declared inside a function and is only available within the function in which it is declared
3. Learn if state, if … else state, and switch statements
4. Learn while statements, do … while statements, for, and foreach statements
5. Learn about include require statements
    a. Allows reuse of content by allowing the insertion of content of an external file on multiple web pages.
        i. include - Generate warning if include file cannot be found
        ii. require - Halts the processing of web pages and display an error if the include file cannot be found
    b. include_once and require_once statements assure that the external files is added to the script only one time
6. Variadic function
    a. function a(...$numbers)
    b. This allows function to take in multiple value. Then do this
    ```
     foreach ($numbers as $number) {
          $total += $number;
       }
    ```

Manipulating String (theory or give some example)

1. Construct text string
    a. $StringVariable = “<p>PHP literal text string<p>”;
    b. echo $StringVariable;
2. Work with single strings
    a. Concatenation operator(.)
    b. \ is escape character
    c. $Vegetable = "carrot";
    echo "Do you have any {$Vegetable}s?";
    Using {} to add s to the variable when printing
    d. strlen($var) = length of string
    e. str_work_count($var) = number of words
    f. strtoupper() (uppercase all letters)
    g. strtolower() (lowercase all letters)
    h. ucfirst() (ensure that the first character of a string is uppercase)
    i. lcfirst() (ensure that the first character of a string is lowercase)
    j. ucwords() (uppercase first characters of each word)
    k. htmlspecialcharas() and html_specialcharacters_decode() (turns value into text even if there are special characters and vice versa)
    l. trim() removes leading or trailing spaces in string, ltrim() removes only leading spaces, rtrim() removes trailing spaces
    m. substr(string, start, optional length); —
    $ExampleString = "woodworking project"; echo substr($ExampleString,4) .
    n. strpos() performs case sensitive search between 2 variables. First argument is the string to be searched and the second argument is the characters to search. If not found, function returns boolean false
    o. strchr($x, “.”) removes the first part before the “.”
    p. strrchr($x, “.”) removes the part until the last “.”
    q. str_replace("email", "e-mail", $Email)
    r. $x = strtok($x, delimiter);
    s. $array = explode(delimiter, $x);
    t. $variable = implode(delimiter, array);
3. Compare Strings
    a. strcasecmp() case insensitive search and strncasecmp if only want to compare specific number of characters
    b. strcmp() case sensitive comparison and strncmp if only want to compare specific number of characters
    c. similar_text() returns number of characters that two strings have in common
    d. levenshtein() returns the number of characters needed to change for two strings to be the same
    e. soundex() returns value representing name phonetic equivalent
    f. metaphone() returns code representing english word approximate sound
4. Use regular expression

a. preg_match(“/pattern/i”, string);
b. “.” matches any single character, “\” makes next character as literally value, ”^” anchor character to beginning string, “$” anchor character to end of string, ”()” specifies required characters to include in pattern, “[]” specifies alternate characters allowed in pattern match, [^]specifies characters to exclude in a pattern match, “-” identifies a possible range of characters to match, “ | “ specifies an alternate set of characters to include in the pattern.
c. preg_match(“/...../”, string); returns 1 if is 5 character string
d. “?” specify that the preceding character is optional so if it is https?, http will also return true. “+” specifies that one or more preceding characters must match “*” specifies zero or more of the preceding character can match, “{n}” specifies that the preceding character repeat exactly n times, “{n,}” specifies that preceding character repeat at least n times, “{,n}” specifies that preceding character repeat up to n times, “{n1, n2}” specifies that preceding character repeat at least n1 times and no more than n2 times
e. “\w” any letter, number or underscore character. “\W” anything that is not in \w, “\s is white space character, “\v” vertical white space character like if the line starts with \n. “\h” horizontal white space which is “ “ instead of \n
f. preg_match("/^[\w-]+(\.[\w-]+)*@[\w-]+ (\.[\w-]+)*(\.[a-zA-Z]{2,})$/", $Email);

Handling User Input (theory, codes to explain, autoglobal, get, post)

1. Learn about autoglobal variable

```
$globalVariable = “a”;
function example(){
        echo $GLOBALS[“globalVariable”];
}
example();
```

2. Build HTML Web formsProcess form data
    a. <form action = “www.example.com” method=”POST”>
    b. $GET[‘x’] or $POST[‘x’] can be used with $_REQUEST[‘x’]
3. Handle submitted form data
    a. is_numeric() or is_*() are used to determine if variable contains a number
    b. round() used to a numeric variable with an appropriate number of decimal place
    c. stripslashes() to remove leading slashes for escape sequences
4. Create an All-in-One form
    a. Conditions:
        i. If the data has been submitted and needs to be validated
        ii. If the form needs to be redisplayed(first time or error)
    b. isset($_POST(‘submit’)) is used to determine if the form has been submitted
5. Display dynamic data based on URL token

```
<?php
   if (isset($_GET['content'])) {
      switch ($_GET['content']) {
         case 'About Me':
            include('inc_about.html');
            break;
         case 'Contact Me':
            include('inc_contact.html');
            break;
         case 'Home': // A value of 'Home' means to display the default page
         default:
            include('inc_home.html');
            break;
      }
   }
   else // No button/link has been selected
      include('inc_home.html');
```

6. ?>

```
<a href = "WebTemplate.php?content=Home">Home</a><br />
<a href = "WebTemplate.php?content=About+Me">About Me</a><br />
<a href = "WebTemplate.php?content=Contact+Me">Contact Me</a><br />
```

7. Emailing the web form data

```
mail(recipient, subject, message);
```

Working with Files and Directories (theory question and coding)

1. Understand file type and permissions
    a. File type affects how information is stored in files and retrieved from them
    b. File permissions determine the actions that a specific user can and cannot perform on a file
    c. Binary file is a series of characters or bytes for which PHP attaches no special meaning and the structure is determined by the application that reads or writes to the file
    d. Text file has only printable characters and a small set of control or formatting characters
        i. Text files translate the end of line character sequences as \n or \r\n to carriage returns
    e. File permissions - calculated using four digit octal. First digit is always 0, the other 3 digit represents RWX dor:
        i. User
        ii. Group
        iii. Other
    f. chmod($filename, 0754)
    g. fileperms() is used to read permissions associated with a file

```
$perms =fileperms($testfile);
$perms = decoct($perms % 01000);
```

2. Work with directories
    a. opendir($handle) function is used to iterate through entries in a directory and handle is a special variable that is representing the directory
    b. readdir() returns the file and directory names of the opened directory and the directory pointer will move to the next file after it returns the current file or directory name
    c. scandir ($handle) also returns the whole entry to an array sorted in ascending alphabetical order and if 1 is passed as a second argument scandir($handle,1) it will do it in descending order
    d.

```
$DirOpen = opendir($dir);
while ( $curfile = readdir($DirOpen)){
if((strcmp($curfile, '.') != 0 && (strcmp($curfile,'..') != 0)){
        echo "<a href=\"./" . $curfile . "\">" . $curfile . "</a><br>\n";
}}
closedir($DirOpen);
```

    e.

```
$dir = ".";
$direntries = scandir($dir);
foreach ($direntries as $entry){
        if ((strcmp($entry, ".") != 0) && (strcmp($entry, "..") != 0)){
        echo "<a href=\"./" . $entry . "\">" . $entry . "</a><br>\n";
}}
```

f. mkdir("example") creates new directory in the current directory or use an absolute or relative path to create it in another location
g. file_exist() checks if file or directory exists

3. Upload and download
    a. <input type="file" name="pic'/>
    By adding <input type="hidden" name="MAX_FILE_SIZE" value = "500000"/>, it gives the file input 500000 byte limit
    b. The uploaded file will be stored in the $_FILES auto global array and the key to the uploaded file is the name of the input field. Is 2 dimension array that also stores error(error code associated with the file), tmp_name(temporary location of file contents), name(name of original file), size(of file in bytes) and type
    c. Considerations before moving to permanent position:
        i. Whether the file should be immediately available or verified first like scanning for virus
        ii. Determine if the file is for the public or or private users
    d. move_uploaded_file($file, $destination), where $file is the tmp_name in the $_FILES global variable

```php
<?php
    $Dir = "."; //../examples";
if (isset($_POST['upload'])) {
    if (isset($_FILES['new_file'])) {
        if (move_uploaded_file($_FILES['new_file']['tmp_name'],
            $Dir . "/" . $_FILES['new_file']['name']) == TRUE) {
            chmod($Dir . "/" . $_FILES['new_file']['name'], 0644);
            echo "File \"" .
                htmlentities($_FILES['new_file']['name'])
                ."\"successfully uploaded. <br />\n";
        }
        else
            echo "There was an error:".
                $_FILES['new_file']['error'] . " uploading \"" .
                htmlentities($_FILES['new_file']['name']) .
                "\".<br />\n";
    }
}
?>
```

    e. Downloading the files
        i. Files in public directory can be downloaded with html hyperlink
        ii. Private directory requires a three step process.
        iii. When users click on the download button, href them into the script, <a href=\"fileDownloader.php?filename$Entry where $Entry is file name.
        iv. fileDownloader.php

```
if (isset($_GET['filename'])) {
     $FileToGet = $Dir . "/" . stripslashes($_GET['filename']);
     if (is_readable($FileToGet)) {
          header("Content-Description: File Transfer");
          header("Content-Type: application/force-download");
          header("Content-Disposition: attachment; filename=\"" .
              $_GET['filename'] . "\"");
          header("Content-Transfer-Encoding: base64");
          header("Content-Length: " . filesize($FileToGet));
          readfile($FileToGet);
          $ShowErrorPage = FALSE;
     }
```

4. Write and read data to files
    a. file_put_contents($file,$content) - deletes and replace the content with new one
    b. file_put_contents($file,$content,FILE_APPEND)
    c. filewrite
    d. file_get_content() and readfile() returns a string value and removes the \n.
    e. $content = file($filename);
       $x = count($content);
       for($i=0;$i=<$x;i++){
       echo “$content[$i]”;
       }
5. Open and close a file stream
    a. fopen() to open stream and fclose() to close stream
    b. $file = fopen(“file.txt”,”a = append, w = write which deletes everything, r = read”)
    c. fwrite($file, $content). Can limit length of user input by putting length at the end to specify number of characters. fwrite($file,$content,20)
    d. fclose($file)
    e. $flock($file, argument), LOCK_EX(exclusive lock for writing), LOCK_NB(block the file until it is unlocked), LOCK_SH(open for reading), LOCK_UN(unlock)
    f. fgets($file) reads line by line as it is being called. Needs fopen()
    g. feof() returns true when pointer reaches EOL
6. Manage files and directories
    a. copy($source, $destination) - can be used to move but needs to unlink old file
    b. rename($oldpath,$newpath) -can be used to move
    c. rmdir() - directory needs to be empty
    d. unlink() to remove file

Manipulating Arrays (theory and explain code)

1. Manipulate array elements
    a. array_shift($array) removes the first element from the beginning of an array
    b. array_unshift($array, “one”, “two”) - adds to the front of the array
    c. array_pop($array) removes the last element from the array
    d. array_push($array, “two”, “last”) - adds to the end of the array where last is the last value
    e. array_splice() adds or remove array elements located anywhere in the array
        i. array_splice($array , 3, 0, array(“one”, “two”) - slots one and two to third index of array
        ii. array_splice($array, 2,4) - removes items 2,3,4
    f. unset($array[2]); - removes value from index 2
    g. $array = array_values($array) - re index the array
    h. $unique = array_unique($array) - removes the duplicates and returns new array but does not renumber the index
2. Declare and initialise associative arrays
    a. $array = array(key=>value, …);
    b. If array is initialized where $array[“a”] = “a” and $array[] = “b”, $array[0] is “b”
    c. If array is initialized where $array[20] = “a” and $array[] = “b”, $array[21] is “b”
3. Iterate through an arrayFind and extract elements and values
    a. foreach($array as $key=>$value)
    b. in_array(“a”, $array) returns boolean true if value exist in array
    c. array_search(“a”, $array) determines whether a given value exist in array and returns the index of the first matching element if value exist or return false if it does not exist
    d. array_key_exists(“a”, $array) searches for the key and returns true false
    e. $key = $array_keys($keyvalarray) returns whole key array
    f. $slice = array_slice($array, 2, 3); $slice will consist of items in $array from index 2,3,4. From index 2 and 3 items
4. Sort, combine, and compare arrays
    a. Shuffle cannot be used with associative array as it will delete the key
    b. sort() and rsort() is used for indexed arrays only
    c. Associative arrays use asort(), ksort(), arsort(), krsort(). A for values, k for key, in ascending, 1234. arsort is descending.
    d. $keyval = array_merge($key,$val); arraymerge 3 arrays or more will make it a normal array
    e. array_diff() returns an array that compares the first array with all the other arrays that are in the argument and the keys or indexes are not renumbered
    f. array_intersect() returns an array that compares the first array with all the other arrays that are in the argument that have a matching value and the keys or indexes are not renumbered (all three need to have matching value)
5. Use arrays in Web forms
    a. Use name=”answer[]” for inputs.

b.
```
if (is_array($_POST['answers']) {
$Index = 0;
foreach ($_POST['answers'] as $Answer) {
        ++$Index;
        echo "The answer for question $Index is '$Answer' <br />\n";
        }
}
```

6. Multi Dimensional arrays

Working with MYSQL and Manipulating MYSQL Database with PHP (theory, coding)

1. Connect to MYSQL from PHP
    a. $conn = mysqli_connect(“localhost”, “user”, “password”, “databaseName”)
    b. mysqli_close($conn)
    c. mysqli_connect_error() returns error code and description from the attempt to connect to the database
    d. try{
        $conn = mysqli_connect(“localhost”,”username”, “password”,”database”)
    }
    catch(mysqli_sql_exception $e){
        die(“connection failed:” . mysqli_connect_errno() . “=” .
    mysqli_connect_error());
    }
    $result = mysqli_query($conn, $sql)
2. Work with MYSQL databases using PHP
    a. Create - CREATE DATABASE myDB
    b. Delete - DROP DATABASE myDB
    c. mysqli_select_db($conn, “database”)
3. Create, modify, and delete MYSQL tables with PHP
    a. CREATE TABLE test (id INT(10) UNSIGNED AUTO-INCREMENTAL PRIMARY KEY, name VARCHAR(30) NOT NULL);
    b. DROP TABLE test;
4. Use PHP to manipulate MYSQL records and retrieve database records
    a. INSERT INTO test(name, number) VALUES (‘test’,’99999999’), (‘test1’,’99999999’),
    b. $id = mysqli_insert_id($conn)
    c. UPDATE test SET name= ‘$name’ , number = ‘$number’ WHERE id = ’$id’
    d. SELECT * FROM test
    e. DELETE FROM test where id = ‘1’
    f. $result = mysqli_query($conn, $sql)
    g. if(mysqli_num_rows($result) >0){
    $i=0;
     while(($row = mysqli_fetch_assoc($result)) != FALSE){
      $keyval[$i] = $row;
       $i++;
    }
    }
5. PHP prepared statements
    a. $stmt = mysqli_prepare($conn, “INSERT INTO test (name, number) VALUES (?,?);”);
    mysqli_stmt_bind_param($stmt,”si”, $name, $number);
    $name = “test”; $number = “99999999”;
    mysqli_stmt_execute($stmt);

```
mysqli_stmt_close($stmt)
```

b. //select

c.
```
$stmt = mysqli_prepare($conn, “SELECT name, number FROM test;”);
mysqli_stmt_execute($stmt);

//can do this
mysqli_bind_result($stmt, $name, $number);
while (mysqli_stmt_fetch($stmt)){
  echo $name . $number;
}

//or
$result = mysqli_stmt_get_result($stmt)
While (($row = mysqli_fetch_assoc($result)) != FALSE){
        Echo $row[‘name’]
}
```

Managing State Information (theory and explain code)

1. Learn about state information
    a. State information is information about an individual visit to a website
    b. Maintaining state is to store persistent data about website visits
    c. Stores information temporarily for a user
    d. Four Tools to manage state information
        i. Hidden field
        ii. Query strings
        iii. Cookies
        iv. Sessions
2. Use hidden form fields to save state information
    a. Input type hidden
    b. When submitting a form to a script, access variables through $_GET[] and $POST[]
    c. Potential to be altered by user maliciously
    d. Does not permanently maintain state
3. Use query strings to save state information
    a. click
    b. Use $_GET[‘name’] and $_GET[‘email’]
    c. Does not permanently maintain state as well
4. Use cookies to save state information
    a. Temporary cookie - remain available only for the current browser session
    b. Persistent cookie - remain available beyond the current browser session and are stored in a text file on client computer (set by using a longer expiration date)
    c. Cookies need to be set before sending any output to the web browser
    d. setcookie(name, value, expires, path, domain, secure). For secure, 1 is true and 0 is false
    e. setcookie(“name”, “test”, time() + 3600, “/test” ,1)
    f. echo $_COOKIE(‘name’);
    g. setcookie(“array[name]”, “test”)
    h. setcookie(“name”, “” time() -3600) to delete cookie
5. Use sessions to save state information
    a. session_start() starts a new session or continues an existing one and it generates a unique session ID to identify the session
    b. Session id is a random alphanumeric string
    c. session_start() creates a text file on web server that is the same name as the sessionID preceded by sess_ and is stored in the web server directory specified by the session.save_path directive in the php.ini configuration file
    d. Web browser is automatically configured to assign session id to cookies. It can also be changed to be sent as a query string or it can be sent as hidden form field to any web pages
    e. SID or session_id() to get id

f. $_SESSION[‘name’] = test;
g. Deleting session -
h. session_start()
   $_SESSION = array();
   session_destroy()

Developing Object-Oriented PHP (theory, coding)

1. Study object-oriented programming concepts
    a. Refers to the merging of related variables and functions into a single interface
    b. Object is a component and is treated like one
    c. Considered as cooperative problem solving through objects communicating with one another
    d. Data refers to information contained within variable or storage structure
    e. Variable that are associated with an object are called properties or attributes
    f. Functions associated with objects are methods
2. Encapsulation
    a. Encapsulation is the hiding of internal code and data containing it within the object itself.
    b. Object interface refers to the methods and properties that are required for a source program to communicate with an object
    c. Allows user to only see what is allowed
    d. Reduces complexity of code
    e. Prevents other programmer from introducing bug or stealing code
    f. Everything that makes up an object are organised into a class and an instance is an object that is created from an existing class
    g. The functions in a class are class members
    h. Class variable is attribute
    i. Class functions are methods
3. Use objects in PHP scripts
    a. $x = new ClassObj()
    b. -> is used to access methods and properties in the object
    c. class test extends parent{
        private $x = 0;
        public $y = “”;
    }
    d. get_class($obj) returns the class of obj
    e. class_exist(‘test’) returns true or false
    f. if($obj instanceof test) returns true or false depending on whether obj is a instance of test
    g. Public, private, protected access specifiers
    h. Using $this-> refers to using the attribute or method within the class
4. Declare data members in classes
    a. class bank{
        private $accountNumber;
        private $name;
        private $balance;
        function __construct(){
            $this->accountNumber = 0;
            $this->name = “”;

```
                $this->balance = 0;
        }
        public function setName($name){
                $this->$name=$name;
        }
        public function getName(){
                return $this->$name;
        }
        public function __destruct() {
                $conn->close();  // destruct used mainly for connection
         }

}
```

Accessor/Mutator functions are the get and set respectively

5. Work with class member functions
6. Inheritance
    a. Build new class based on existing class without having to rewrite the code contained in the existing one
    b. class test extends parent{
        private $x = 0;
        public $y = “”;
    }
7. Polymorphism
    a.
```
<?php
class Shape {
   public function calculateArea() {
      return 0;
   }
}

class Circle extends Shape {
   private $radius;

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

   public function calculateArea() {
      return pi() * $this->radius * $this->radius;
   }
}

class Rectangle extends Shape {
   private $width;
```

```
    private $height;

    public function __construct($width, $height) {
        $this->width = $width;
        $this->height = $height;
    }

    public function calculateArea() {
        return $this->width * $this->height;
    }
}

// Polymorphic function
function printArea(Shape $shape) {
    echo "Area: " . $shape->calculateArea() . "\n";
}

$circle = new Circle(5);
$rectangle = new Rectangle(4, 6);

printArea($circle);
printArea($rectangle);
?>
```

8. Interfaces
    a.
```
interface Vehicle {
    public function start();
    public function stop();
}

class Car implements Vehicle {
    public function start() {
        echo "Car started.\n";
    }

    public function stop() {
        echo "Car stopped.\n";
    }
}

class Motorcycle implements Vehicle {
    public function start() {
        echo "Motorcycle started.\n";
    }
```

```
    public function stop() {
        echo "Motorcycle stopped.\n";
    }
}
```

9. Abstract classes
    a. Only abstract class can declare abstract method
    b. Cannot be instantiated
    c.
```
abstract class AbstractClassA {
    abstract public function methodA();
}
class MyClass extends AbstractClassA {
    public function methodA() {
        // Implement methodA
    }
}
```

10. Traits
    a.
```
class test{
        use trait;
}
```
    b. Traits cannot have abstract functions

PHP-XML, PHP-AJAX Recursion and Data Structures in PHP (theory and what is the output)

1. XML, PHP XML Parsers
    a. XML does not define a specific set of tags to use and must have strict structure
    b. Value of a tag can have other tags within it and they are known as parent tag and child tag
    c. $testData=
    “<?xml version='1.0' ?>
    <contact idx='37' >
    <name>Tom</name>
    <phone type='home'>66666666</phone>
    <meta id='x634724' />
    </contact>”;
    d. XML Parsers:
        i. Tree-Based Parsers (SimpleXML. DOM)
            1. Allow manipulation with XML data, transform an XML document into a data structure (making it easy to iterate through a collection of arrays and objects)
            2. Provide easy way of getting element name attributes and textual content
            3. $xml = simplexml_load_string($testData);
            print_r($xml);
            4. $xml = simplexml_load_file(test.xml’);
            foreach ($xml->value as $v){
            echo ‘<li>’ . $v->name . $v[‘idx’];
            foreach ($v->phone as $p){
            echo $p[‘type’] . $p;
            }
            }
            To convert XML string or file to array, do this after loading
            $json = json_encode($xml);
            $array = json_decode($json, true);
            5. DOM parser:
            a. echo $xmlDoc->saveXML(); //prints all the value in the tags so tom 66666666
            Have #path =
            b. <?php
            $xmlDoc = new DOMDocument();
            $xmlDoc->load("list.xml");
            $x = $xmlDoc->documentElement;
            foreach ($x->childNodes AS $item) {
            echo $item->nodeName . " = "
            . $item->nodeValue . "
";

```
} ?>

No #path =
```

c.
```
<?php
$xmlDoc = new DOMDocument();
$xmlDoc->load("contacts.xml");
$xml = simplexml_import_dom($xmlDoc);
echo "<p>First Contact: {$xml->contact[0]->name}</p>\n";
echo "<p>First Contact id: {$xml->contact[0]['idx']}</p>\n";
echo "<p>Second Contact: {$xml->contact[1]->name}</p>\n";
?>
```

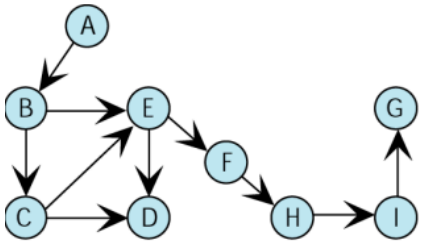
ii. Event-Based Parsers (XMLReader, XML Expat Parser)

2. AJAX, Using AJAX with PHP
    a. Uses XML to communicate
3. Recursion
    a. Calling the function again e.g. factorial ($x - 1); in the function will reset the thing back to the top until the function escapes through an if statement. The thing will then unravel from the latest call to the oldest call which is the first call when user calls it
    b. Use a static variable to have a variable that can be kept when the function is called again. Like this
    c.
```
function test()
{
   static $step = 0;
   if ($step < 10) {
                $step++;
      echo "<p>into function step = $step </p>";
      test();
      $step--;
      echo "<p>out of function step = $step </p>";
   }
```

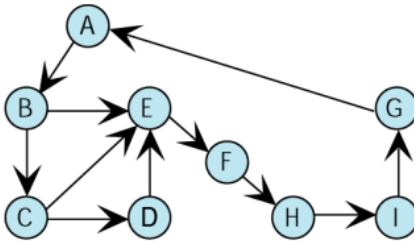
4. Data Structure
    a. Way of handling data through a conceptual model is known as Abstract data type
    b. Data structure are concrete representation
    c. Some common ADT:
        i. List - left to right array
        ii. Map
        iii. Set
        iv. Stack - top to bottom array. Bottom is first item
        v. Queue - front to back where front exits and back enters. Enqueue is used to insert things to the back and dequeue is to remove from the front

vi. Priority queue

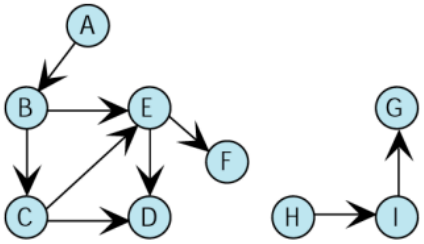
vii. Graph



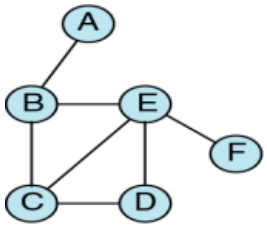
(a) Weakly connected



(b) Strongly connected



(c) Disjoint graph



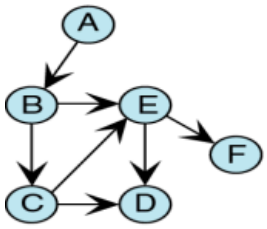
A
B
C
D
E
F

Vertex vector

	A	B	C	D	E	F
A	0	1	0	0	0	0
B	1	0	1	0	1	0
C	0	1	0	1	1	0
D	0	0	1	0	1	0
E	0	1	1	1	0	1
F	0	0	0	0	1	0

Adjacency matrix

Adjacency matrix for non-directed graph



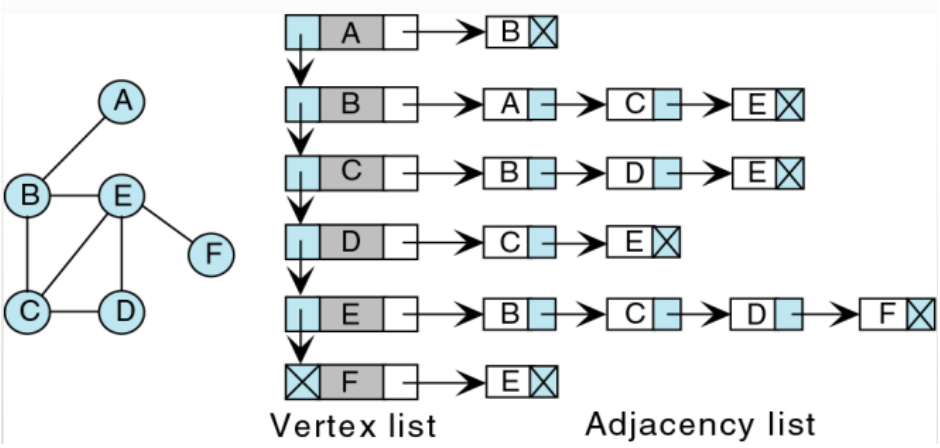
A
B
C
D
E
F

Vertex vector

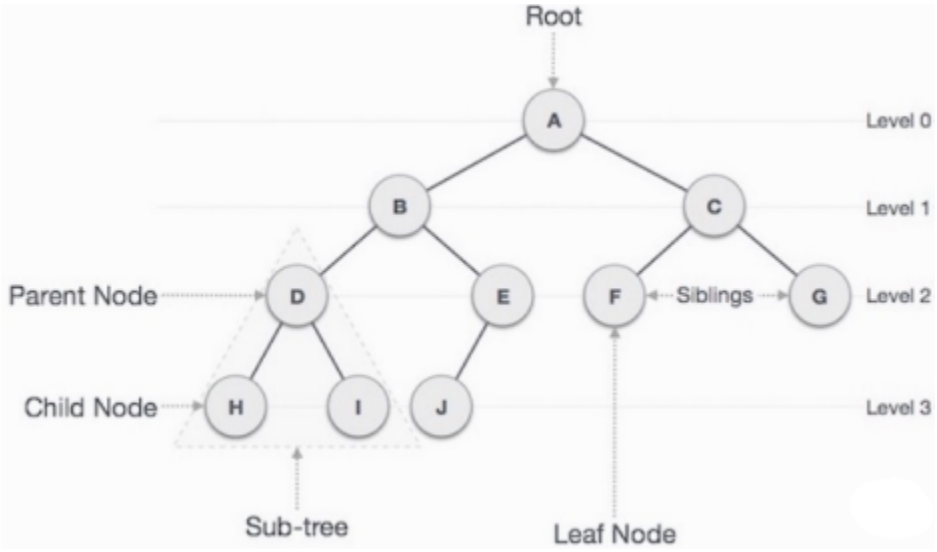
	A	B	C	D	E	F
A	0	1	0	0	0	0
B	0	0	1	0	1	0
C	0	0	0	1	1	0
D	0	0	0	0	0	0
E	0	0	0	1	0	1
F	0	0	0	0	0	0

Adjacency matrix

Adjacency matrix for directed graph


A
B
C
D
E
F
Vertex list
Adjacency list

viii. Tree


Root
A
B
C
D
E
F
G
H
I
J
Level 0
Level 1
Level 2
Level 3
Parent Node
Child Node
Siblings
Sub-tree
Leaf Node