SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Ramapuram, Chennai

FACULTY OF SCIENCE AND HUMANITIES

Department of Computer Science

PRACTICAL RECORD

NAME :

REGISTER NO :

COURSE : **B.Sc. COMPUTER SCIENCE**

SEMESTER / YEAR : **V / III**

SUBJECT CODE : **UCS23501J**

SUBJECT NAME : **INTERNET PROGRAMMING**

NOVEMBER 2025

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

Ramapuram, Chennai

FACULTY OF SCIENCE AND HUMANITIES

Department of Computer Science

REGISTER NUMBER:

BONAFIDE CERTIFICATE

This is to certify that the bonafide work done by _____in the subject **INTERNET PROGRAMMING[UCS23501J]** at SRM Institute of Science and Technology, Ramapuram, Chennai in November 2025.

STAFF IN CHARGE **HEAD OF THE DEPARTMENT**

Submitted for the University Practical Examination held at SRM Institute of Science and Technology, Ramapuram, Chennai on_____

INTERNAL EXAMINER 1 **INTERNAL EXAMINER 2**

INDEX

S.NO	DATE	EXPERIMENTS	PAGE NO	SIGN
1	5/6/2025	Learning to work with Linux Server		
2	12/6/2025	Working with files and Directory commands		
3	19/6/2025	Working With file commands Creating and modifying files using Vi editor		
4	26/6/2025	Writing a simple PHP Programs		
5	3/7/2025	Operators and control flow statements		
6	10/7/2025	Embedding PHP Script in PHP HTML		
7	31/7/2025	Various types of parameters and types of functions in PHP		
8	7/7/2025	Working wit strings and arrays in PHP		
9	14/8/2025	Files and Directory operations in PHP		
10	21/8/2025	Exception handling and classes in PHP		
11	4/9/2025	Working with databases and tables		
12	11/9/2025	Working with queries and joins and sub queries in MySQL		
13	18/9/2025	Manipulation of cookies and sessions using PHP		
14	25/9/2025	Form validation connecting with sql data using PHP functions		
15	27/9/2025	Creating PHP web applications to manipulating data[CURD operations] from MYSQL table.		

EX.NO: 1 **NAME:**

DATE: **REG NO:**

Learning to work with Linux Server

AIM

To understand and perform basic operations in a Linux server environment, including file management, user management, and process handling.

PROCEDURE

1. **Login to the Linux Server**
    - Use SSH or terminal to connect.
    - Command:
    - ssh username@server_ip
2. **Check Current Directory and List Files**
    - Command:
    - pwd
    - ls -l
3. **Create and Manage Files**
    - Create a file:
    - touch sample.txt
    - Edit file using vi or nano:
    - vi sample.txt
4. **File Operations**
    - Copy file:
    - cp sample.txt copy_sample.txt
    - Move file:
    - mv sample.txt /home/user/docs/
    - Remove file:
    - rm copy_sample.txt
5. **Directory Operations**
    - Create directory:
    - mkdir testdir
    - Remove directory:
    - rmdir testdir
6. **User and Process Management**
    - Check current user:
    - whoami
    - Display running processes:

- ps -aux
- Kill a process:
- kill -9 <pid>

OUTPUT

Sample output after executing commands:

$ pwd

/home/student

$ ls -l

-rw-r--r-- 1 student student 0 Sep 22 12:30 sample.txt

$ cp sample.txt copy_sample.txt

$ ls

copy_sample.txt sample.txt

$ whoami

student

$ ps -aux

USER PID %CPU %MEM COMMAND

root 101 0.0 0.1 sshd

stud 202 0.1 1.0 bash

stud 303 0.5 2.0 vi

RESULT

The basic Linux server commands for file handling, directory management, user identification, and process management were successfully executed and the results were verified.

EX.NO: 2 **NAME:**

DATE: **REG NO:**

Working with files and Directory commands

AIM

To learn and execute various Linux commands for creating, managing, and manipulating files and directories.

PROCEDURE

1. **Check Current Directory**

   pwd

2. **List Files in a Directory**

   ls

   ls -l

   ls -a

3. **Create a New File**

   touch file1.txt

4. **Create a New Directory**

   mkdir testdir

5. **Copy a File**

   cp file1.txt copy_file.txt

6. **Move/Rename a File**

   mv file1.txt moved_file.txt

7. **Change Directory**

   cd testdir

8. **Remove a File**

   rm copy_file.txt

9. **Remove a Directory**

   rmdir testdir

OUTPUT

$ pwd

/home/student

$ ls

file1.txt testdir

$ ls -l

-rw-r--r-- 1 student student 0 Sep 22 12:40 file1.txt

drwxr-xr-x 2 student student 4096 Sep 22 12:41 testdir

$ cp file1.txt copy_file.txt

$ ls

copy_file.txt file1.txt testdir

$ mv file1.txt moved_file.txt

$ ls

copy_file.txt moved_file.txt testdir

$ cd testdir

$ pwd

/home/student/testdir

$ rm ../copy_file.txt

$ ls

moved_file.txt testdir

$ rmdir testdir

RESULT

The file and directory management commands (pwd, ls, touch, mkdir, cp, mv, cd, rm, rmdir) were executed successfully, and their outputs were verified.

EX.NO: 3 **NAME:**

DATE: **REG NO:**

Working With file commands Creating and modifying files using Vi editor

AIM

To understand and perform file management tasks such as creating, editing, saving, and modifying files using the Vi editor in a Linux/Unix environment.

Procedure

1. Open a file in Vi editor
    - To create or open a file, type:
    - vi filename
2. Understand the modes in Vi
    - Command Mode (default): Used for giving commands (delete, copy, save, quit).
    - Insert Mode: Used for typing text into the file. Press i, a, or o to enter insert mode.
    - Escape Key (Esc): Press Esc to return to command mode from insert mode.
3. Insert text into the file
    - Press i → start typing your text.
    - Press Esc to stop editing.
4. Save the file
    - In command mode, type:
    - :w

(saves the changes without exiting).

5. Save and exit the file
    - Type:
    - :wq

(writes the changes and quits).

    - Or, type:
    - :x
6. Exit without saving
    - Type:
    - :q!
7. Delete text

- x → deletes a single character.
- dd → deletes the current line.
- ndd → deletes *n* lines (e.g., 5dd deletes 5 lines).

8. Copy and paste text

    - yy → copies the current line.
    - nyy → copies *n* lines.
    - p → pastes the copied content after the cursor.

9. Search text

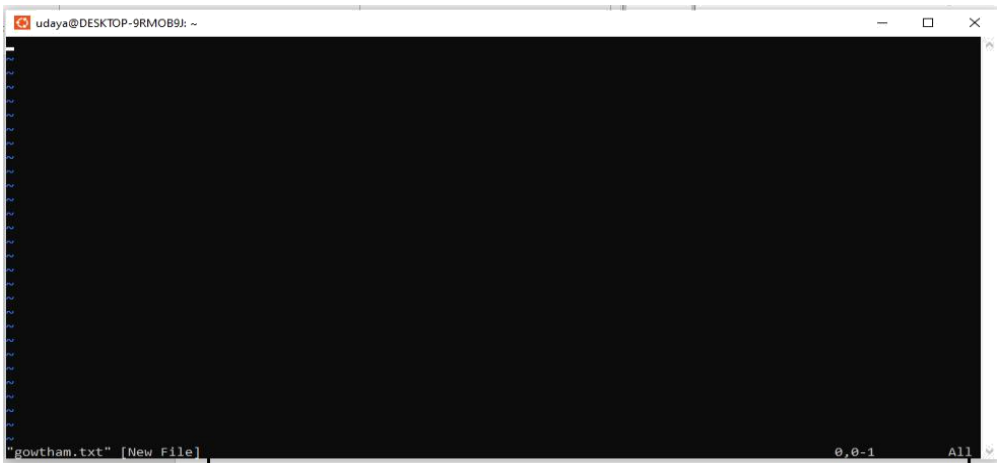
    - /word → searches for the word in forward direction.
    - ?word → searches backward.
    - n → repeats the search in the same direction.

10. Undo and redo changes

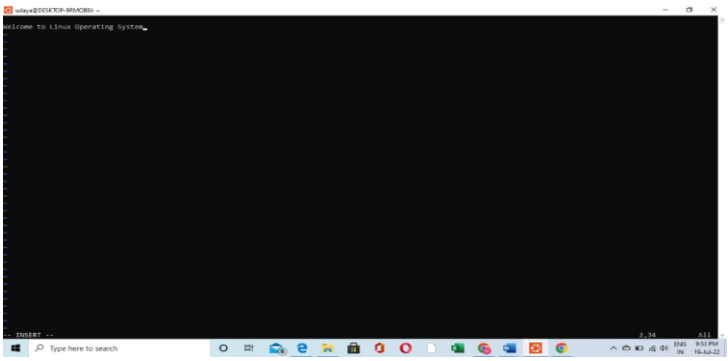
    - u → undo last change.
    - Ctrl + r → redo the undone change.

Commands

Labex:~$ vi gowtham.txt



For Insert press **i** key in the keyboard

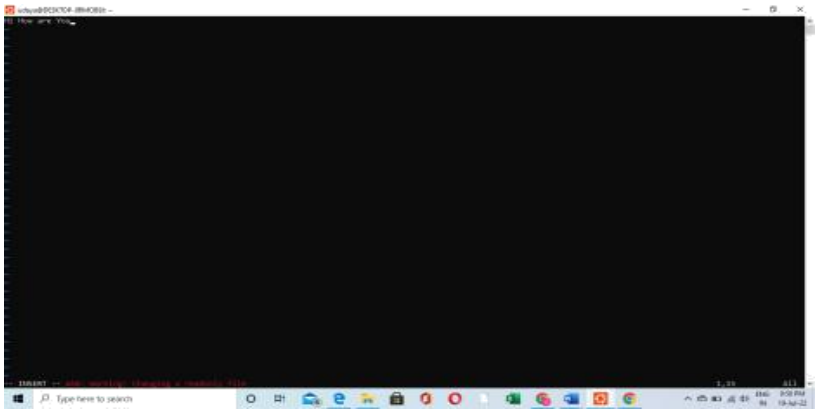


To save and Quit press esc key and press :wq and press Enter

Labex:~$ cat gowtham.txt

Welcome to Linux Operating System

Labex:~$ vi -R harish.txt



To save and Quit press esc key and press :wq! and

press Enter Labex:~$ cat harish.txt Hi How are You

Result: The above commands executed and verified successfully

EX.NO: 4(a) **NAME:**

DATE: **REG NO:**

Writing a simple PHP Programs

Student's Bio-Data Using the Print Statement.

AIM

To write a PHP program that displays a student's Bio-Data using the print statement.

PROCEDURE

1. Start the PHP script using <?php … ?>.
2. Use the print statement to display each field of the Bio-Data.
3. Apply HTML tags like
 for line breaks and <pre> for formatted alignment.
4. Save the file with .php extension (e.g., biodata.php).
5. Run the program on a PHP server (XAMPP/WAMP/LAMP) and verify the output in the browser.

Program:

```
<?php
echo "<h3>Bio-Data</h3>";
echo "**********<br>";
echo "Name : Dr .P. Bhargavi Devi <br>";
echo "Date of Birth : 13-10-1983 <br>";
echo "Class : III B.Sc b Section <br>";
echo "Father Name : Mr. P.V. Chandra Mowly <br>";
echo "Mother Name : Mrs. P.Radha rani  <br>";
echo "Father Occupation :  <br>";
echo "Father Contact No : 9962157132 <br>";
echo "Address : no 2, Stunt,Somu street,MGR Nagar, Chennai <br>";
?>
```

Output

Bio-Data

Name : Dr. P. Bhargavi Devi
Date of Birth : 13-10-1983
Class : III B.Sc B Section
Father Name : Mr. P.V. Chandra Mowly
Mother Name : Mrs. P. Radha Rani
Father Occupation :
Father Contact No : 9962157132
Address : No 2, Stunt, Somu Street, MGR Nagar, Chennai

RESULT

The PHP program to display Bio-Data using the print statement was executed successfully, and the expected output was obtained.

EX.NO: 4(b) **NAME:**

DATE: **REG NO:**

Writing a simple PHP Programs

MULTIPLICATION TABLE

Aim:

Write a PHP program to display Multiplication Table 1 to 10.

Program:

```
<html>
<body>
<title> Multiplcation Table </title>
<head> <h1> Multiplication Table 1 to 5 </h1>
<br>
<br>
<table align="left" border="1" cellpadding="3" cellspacing="0">
<?php
for($i=1;$i<=10;$i++)
{
echo "<tr>";
for ($j=1;$j<=5;$j++)
  {
  echo "<td>$i * $j = ".$i*$j."</td>";
  }
  echo "</tr>";
  }
?>
</table>
</body>
</html>
```

Output

Multiplication Table 1 to 5

1 * 1 = 1	1 * 2 = 2	1 * 3 = 3	1 * 4 = 4	1 * 5 = 5
2 * 1 = 2	2 * 2 = 4	2 * 3 = 6	2 * 4 = 8	2 * 5 = 10
3 * 1 = 3	3 * 2 = 6	3 * 3 = 9	3 * 4 = 12	3 * 5 = 15
4 * 1 = 4	4 * 2 = 8	4 * 3 = 12	4 * 4 = 16	4 * 5 = 20
5 * 1 = 5	5 * 2 = 10	5 * 3 = 15	5 * 4 = 20	5 * 5 = 25
6 * 1 = 6	6 * 2 = 12	6 * 3 = 18	6 * 4 = 24	6 * 5 = 30
7 * 1 = 7	7 * 2 = 14	7 * 3 = 21	7 * 4 = 28	7 * 5 = 35
8 * 1 = 8	8 * 2 = 16	8 * 3 = 24	8 * 4 = 32	8 * 5 = 40
9 * 1 = 9	9 * 2 = 18	9 * 3 = 27	9 * 4 = 36	9 * 5 = 45
10 * 1 = 10	10 * 2 = 20	10 * 3 = 30	10 * 4 = 40	10 * 5 = 50

Result:

Thus the program has been executed successfully.

EX.NO: 5 **NAME :**

DATE: **REG NO:**

Operators and control flow statements

AIM

To understand and implement operators (arithmetic, relational, logical) and control flow statements (if, if-else, if-elseif-else, switch) in PHP.

PROCEDURE

1. Start a PHP script using <?php … ?>.
2. Declare variables and assign values.
3. Perform arithmetic operations (+, -, *, /, %) and display results using echo.
4. Use relational operators (==, !=, >, <, >=, <=) to compare values.
5. Apply logical operators (&&, ||, !) in conditional statements.
6. Implement control flow statements:
    - if to check a condition.
    - if-else for two alternatives.
    - if-elseif-else for multiple conditions.
    - switch for multiple choices based on a single variable.
7. Save the program with .php extension and run it on a PHP server (XAMPP/WAMP/LAMP).

PROGRAM

```
<?php
// Operators and Control Flow in PHP
$a = 15;
$b = 10;
echo "<h3>Operators in PHP</h3>";
echo "a = $a, b = $b <br>";
echo "Addition: " . ($a + $b) . "<br>";
echo "Subtraction: " . ($a - $b) . "<br>";
echo "Multiplication: " . ($a * $b) . "<br>";
echo "Division: " . ($a / $b) . "<br>";
echo "Modulo: " . ($a % $b) . "<br><br>";
// Relational Operators
echo "<h3>Relational Operators</h3>";
echo "a == b : " . ($a == $b ? "True" : "False") . "<br>";
echo "a != b : " . ($a != $b ? "True" : "False") . "<br>";
echo "a > b : " . ($a > $b ? "True" : "False") . "<br>";
echo "a < b : " . ($a < $b ? "True" : "False") . "<br><br>";
// Control Flow - if, if-else
echo "<h3>Control Flow Statements</h3>";
if($a > $b){
   echo "a is greater than b <br>";
}else{
   echo "a is not greater than b <br>";
}
// if-elseif-else
if($a > $b){
   echo "a is greater than b <br>";
}elseif($a == $b){
   echo "a is equal to b <br>";
```

```
}else{
    echo "a is less than b <br>";
}
// switch statement
$grade = 'B';
switch($grade){
    case 'A':
        echo "Excellent <br>";
        break;
    case 'B':
        echo "Good <br>";
        break;
    case 'C':
        echo "Average <br>";
        break;
    default:
        echo "Needs Improvement <br>";
}
?>
```

OUTPUT

Operators in PHP

a = 15, b = 10

Addition: 25

Subtraction: 5

Multiplication: 150

Division: 1.5

Modulo: 5

Relational Operators

a == b : False

a != b : True

a > b : True

a < b : False

Control Flow Statements

a is greater than b

a is greater than b

Good

RESULT

The PHP program successfully demonstrated the use of arithmetic, relational, and logical operators, along with control flow statements (if, if-else, if-elseif-else, switch). The output was verified and found correct.

EX.NO:6 **NAME:**

DATE: **REG NO:**

Embedding PHP Script in PHP HTML

AIM

To embed a PHP script within an HTML form to process a student marksheet, calculate total, average, grade, and display the pass/fail result.

PROCEDURE

1. **Create HTML Form**
    - Start with a basic HTML structure (<html>, <head>, <body>).
    - Set form method to POST and action to the PHP processing script (studentprocess.php).
    - Include input fields for Student ID, Name, Tamil, English, Maths, and Major marks.
    - Add a submit button to send form data.
2. **Embed PHP Script in studentprocess.php**
    - Use $_POST to collect form data.
    - Display the input values using echo.
    - Perform validation: check if marks in all subjects are above pass mark (35).
    - Calculate total marks by summing all subjects.
    - Calculate average marks by dividing total by number of subjects.
    - Determine grade based on average using if-elseif-else statements:
        1. O+ for >89, A for >79, B for >69, C for >49, D for fail.
    - Display total, grade, and result (Pass/Fail).
3. **Run the Program**
    - Save HTML file (e.g., marksheet.html) and PHP file (studentprocess.php) in your server directory.
    - Open the HTML form in a browser, enter the student details, and submit.
    - Verify the output on the PHP processing page.

Program:

```
<html>
<head>
<title>Student Marksheet Processing</title>
</head>
```

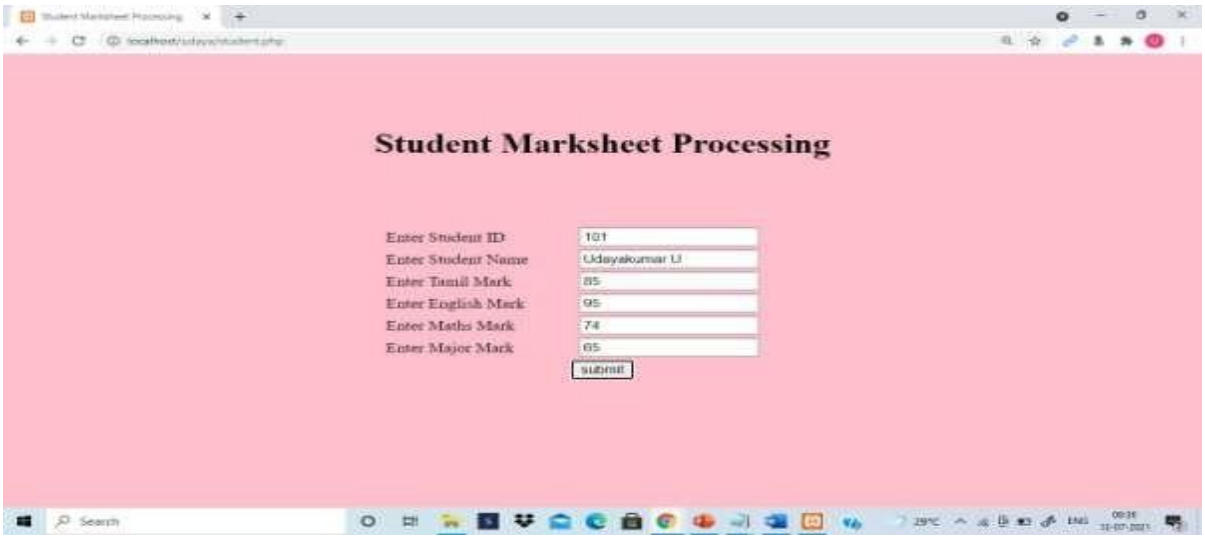
```
<form name="student" method="post" action="http://localhost/Bhargavi/stdprocess.php">
<br><br><br>
<body bgcolor="pink">

<h1><center>Student Marksheet Processing</h1></center>
<br><br><br>

<center>
<table border=0 width=400>
<tr><td>Enter Student ID <td><input type="text" name="stdid" value="">
<tr><td>Enter Student Name <td><input type="text" name="sname" value="">
<tr><td>Enter Tamil Mark <td><input type="text" name="tam" value="">
<tr><td>Enter English Mark <td><input type="text" name="eng" value="">
<tr><td>Enter Maths Mark <td><input type="text" name="mat" value="">
<tr><td>Enter Major Mark <td><input type="text" name="maj" value="">
<tr align=center><td colspan=2><input type="submit" value="submit">
</table>
</center>
</form>
</body>
</html>
```

Studentprocess.php

```
<html>
<body bgcolor="yellow">
</body>
</html>
<?php
echo "Your Rollnumber is:".$_POST["stdid"]."<br/>"; echo "Your Name
is:".$_POST["sname"]."<br/>"; echo "Your Tamil Mark is:".$_POST["tam"]."<br/>"; echo
"Your English Mark is:".$_POST["eng"]."<br/>"; echo "Your Maths Mark
is:".$_POST["mat"]."<br/>"; echo "Your Major Mark is:".$_POST["maj"]."<br/>";
if($_POST['tam']>35 && $_POST['eng']>35 && $_POST['mat']>35 && $_POST['maj']>35)
{

}
else{


}

$valid_res="Pass";

$valid_res="Fail";

$total=$_POST['tam']+$_POST['eng']+$_POST['mat']+$_POST['maj'];
$avg=$total/4;
if($avg>89)
{
$grade="O+";
```

```
}
else if($avg>79)

{
$grade="A";
}
else if($avg>69)
{
$grade="B";
}
else if($avg>49)
{

}
else
{

}

$grade="C";

$grade="D";

if($valid_res=="Fail")
$grade="        "; echo "Your Total is :".$total."<br/>";
echo "Your Grade is :".$grade."<br/>";
echo "Your Result is ".$valid_res."<br/>";
?>
```

Output



Result:

Thus the program has been executed successfully.

EX.NO:7 **NAME:**

DATE: **REG NO:**

Various types of parameters and types of functions in PHP

AIM

To demonstrate calling a function within another function (nested or inner function call) in PHP and perform arithmetic operations.

PROCEDURE

1. **Define basic arithmetic functions**:
    - Create a function add($a, $b) to return the sum of two numbers.
    - Create a function sub($a, $b) to return the difference of two numbers.
2. **Define a main function math($first, $second)**:
    - Inside this function, call add($first, $second) and sub($first, $second) functions.
    - Perform the calculation: divide the result of add() by sub().
    - Cast the result to an integer using (int) and return it.
3. **Call the main function**
    - Use echo math(200, 100); to display the result.

```
<?php
// Function to add two numbers
function add($a, $b){
   return $a + $b;
}

// Function to subtract two numbers
function sub($a, $b){
   return $a - $b;
}

// Function calling other functions
function math($first, $second){
```

```
    $res = add($first, $second) / sub($first, $second);
    return (int)$res;
}
// Display result
echo math(200, 100);
?>
```

OUTPUT

3

RESULT

The program successfully demonstrated calling a function within another function. The arithmetic operations were performed correctly, and the final integer result of dividing the sum by the difference was displayed.

EX.NO: 8 **NAME:**

DATE: **REG NO:**

Working with strings and arrays in PHP

AIM

To demonstrate string manipulation and array handling in PHP, including operations like concatenation, string length, array creation, and array element access.

PROCEDURE

1. **Working with Strings**
    - Declare string variables.
    - Perform concatenation using . operator.
    - Find the length of a string using strlen().
    - Convert strings to uppercase or lowercase using strtoupper() and strtolower().
    - Access individual characters using indexing.
2. **Working with Arrays**
    - Create indexed and associative arrays.
    - Access array elements using their index or key.
    - Count the number of elements using count().
    - Loop through arrays using for or foreach.
3. **Display the results** using echo or print.
4. **Save the program** with .php extension and run it on a PHP server.

PROGRAM

```
<?php
// Working with Strings
$str1 = "Hello";
$str2 = "World";
$concat = $str1 . " " . $str2;
echo "<h3>String Operations</h3>";
echo "String 1: $str1 <br>";
echo "String 2: $str2 <br>";
echo "Concatenated String: $concat <br>";
echo "Length of concatenated string: " . strlen($concat) . "<br>";
echo "Uppercase: " . strtoupper($concat) . "<br>";
echo "Lowercase: " . strtolower($concat) . "<br>";
echo "First character: " . $concat[0] . "<br><br>";
// Working with Arrays
$fruits = array("Apple", "Banana", "Cherry", "Mango");
echo "<h3>Array Operations</h3>";
echo "Fruits Array: <br>";
foreach($fruits as $fruit){
   echo $fruit . "<br>";}
echo "Total number of fruits: " . count($fruits) . "<br>";
// Associative Array
$marks = array("Math" => 95, "English" => 88, "Science" => 90);
echo "<br>Student Marks: <br>";
foreach($marks as $subject => $mark){
   echo "$subject : $mark <br>";}?>
```

OUTPUT

String Operations

String 1: Hello

String 2: World

Concatenated String: Hello World

Length of concatenated string: 11

Uppercase: HELLO WORLD

Lowercase: hello world

First character: H

Array Operations

Fruits Array:

Apple

Banana

Cherry

Mango

Total number of fruits: 4

Student Marks:

Math : 95

English : 88

Science : 90

RESULT

The PHP program successfully demonstrated string manipulations (concatenation, length, case conversion, indexing) and array operations (indexed arrays, associative arrays, element access, and iteration). The output was verified as correct.

EX.NO: 9 **NAME:**

DATE: **REG NO:**

Files and Directory operations in PHP

AIM

To demonstrate file and directory operations in PHP, including creating, reading, writing, and deleting files, as well as creating and removing directories.

PROCEDURE

1. **File Operations**
    - Use fopen() to create or open a file.
    - Use fwrite() to write content to a file.
    - Use fread() or file_get_contents() to read the content of a file.
    - Use fclose() to close the file.
    - Use unlink() to delete a file.
2. **Directory Operations**
    - Use mkdir() to create a new directory.
    - Use rmdir() to remove a directory.
    - Use scandir() to list files and directories.
3. Display all results using echo or print.
4. Save the program as a .php file and run it on a PHP server (XAMPP/WAMP/LAMP).

PROGRAM

```
<?php

echo "<h3>PHP File and Directory Operations</h3>";

// File Operations

$filename = "sample.txt";

// Create and write to a file

$file = fopen($filename, "w") or die("Unable to open file!");

fwrite($file, "Hello PHP!\nThis is a sample file.\n");

fclose($file);

echo "File '$filename' created and data written successfully.<br>";

// Read file content
```

```
$content = file_get_contents($filename);
echo "Reading content of '$filename':<br>";
echo nl2br($content) . "<br>";
// Delete the file
// unlink($filename);
// echo "File '$filename' deleted successfully.<br>";
// Directory Operations
$dir = "testdir";
// Create a directory
if(!is_dir($dir)){
    mkdir($dir);
    echo "Directory '$dir' created successfully.<br>";
}

// List files and directories
$files = scandir(".");
echo "Files and directories in current folder:<br>";
foreach($files as $f){
    echo $f . "<br>";
}
// Remove directory
// rmdir($dir);
// echo "Directory '$dir' removed successfully.<br>";
?>
```

OUTPUT

PHP File and Directory Operations

File 'sample.txt' created and data written successfully.

Reading content of 'sample.txt':

Hello PHP!

This is a sample file.

Directory 'testdir' created successfully.

Files and directories in current folder:

.

..

sample.txt

testdir

[other files in current directory]

RESULT

The PHP program successfully demonstrated file operations (create, write, read, delete) and directory operations (create, list, remove). All operations were executed and verified correctly.

EX.NO: 10 **NAME:**

DATE: **REG NO:**

Exception handling and classes in PHP

AIM

To demonstrate the use of classes and exception handling in PHP, including creating objects, defining methods, and handling runtime errors using try-catch blocks.

PROCEDURE

1. **Classes and Objects**
    - Define a class using the class keyword.
    - Declare properties and methods within the class.
    - Create objects of the class using the new keyword.
    - Call class methods using object references.
2. **Exception Handling**
    - Use a try block to execute code that may throw an exception.
    - Throw exceptions using throw new Exception("Error message").
    - Catch exceptions using a catch block and display the error message.
    - Use multiple catch blocks if handling different types of exceptions.
3. Save the program as a .php file and run it on a PHP server (XAMPP/WAMP/LAMP).

PROGRAM

```
<?php
echo "<h3>PHP Classes and Exception Handling</h3>";

// Define a class
class Student {
   public $name;
   public $marks;
   public function __construct($name, $marks) {
      $this->name = $name;
      $this->marks = $marks;
```

```
    }
    public function display() {
        echo "Student Name: " . $this->name . "<br>";
        echo "Marks: " . $this->marks . "<br>";
    }
    public function checkPass() {
        if($this->marks < 35) {
            throw new Exception("Result: Fail");
        } else {
            echo "Result: Pass<br>";
        }
    }
}
// Create object of class
$student1 = new Student("Bhargavi", 42);
$student2 = new Student("Anjali", 28);
// Display student details and check pass/fail
try {
    $student1->display();
    $student1->checkPass();
    echo "<br>";
    $student2->display();
    $student2->checkPass();
} catch (Exception $e) {
    echo "Exception caught: " . $e->getMessage() . "<br>";
}
?>
```

OUTPUT

PHP Classes and Exception Handling

Student Name: Bhargavi

Marks: 42

Result: Pass

Student Name: Anjali

Marks: 28

Exception caught: Result: Fail

RESULT

The program successfully demonstrated the use of classes and objects in PHP to store student information. It also illustrated exception handling, where a failure condition triggered an exception that was caught and displayed correctly.

EX.NO:11 **NAME:**

DATE: **REG NO:**

Working with databases and tables

AIM

To demonstrate database connectivity in PHP using MySQL, create a database and table, insert records, and retrieve data using PHP.

PROCEDURE

1. **Create a Database (via PHPMyAdmin or MySQL CLI)**

   CREATE DATABASE studentDB;

   USE studentDB;

2. **Create a Table**

   CREATE TABLE students (

   id INT AUTO_INCREMENT PRIMARY KEY,

   name VARCHAR(50) NOT NULL,

   age INT,

   grade VARCHAR(5));

3. **Connect PHP to MySQL Database**

   - Use mysqli_connect() to establish a connection.
   - Handle connection errors using die() or exception handling.

4. **Insert Records into Table**

   - Use SQL INSERT INTO query within PHP.

5. **Retrieve and Display Records**

   - Use SQL SELECT query.
   - Fetch results using mysqli_fetch_assoc() and display using echo.

6. Save the program as .php file and run it on a PHP server (XAMPP/WAMP/LAMP).

PROGRAM

```
<?php

// Database connection

$servername = "localhost";
```

```
$username = "root";
$password = "";
$dbname = "studentDB";

// Create connection
$conn = mysqli_connect($servername, $username, $password, $dbname);
// Check connection
if (!$conn) {
   die("Connection failed: " . mysqli_connect_error());
}
echo "<h3>Connected to Database Successfully</h3>";
// Create table (if not exists)
$sql = "CREATE TABLE IF NOT EXISTS students (
   id INT AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(50) NOT NULL,
   age INT,
   grade VARCHAR(5))";
mysqli_query($conn, $sql);
// Insert records
$sql = "INSERT INTO students (name, age, grade) VALUES
('Anjali', 20, 'A'),
('Bhargavi', 21, 'B'),
('Chitra', 19, 'O')";
mysqli_query($conn, $sql);
// Retrieve and display records
$result = mysqli_query($conn, "SELECT * FROM students");
echo "<h3>Student Records</h3>";
while($row = mysqli_fetch_assoc($result)){
   echo "ID: ".$row['id']." | Name: ".$row['name']." | Age: ".$row['age']." | Grade:
".$row['grade']."<br>";
}
```

```
// Close connection
mysqli_close($conn);
?>
```

OUTPUT

Connected to Database Successfully

Student Records

ID: 1 | Name: Anjali | Age: 20 | Grade: A

ID: 2 | Name: Bhargavi | Age: 21 | Grade: B

ID: 3 | Name: Chitra | Age: 19 | Grade: O

RESULT

The PHP program successfully connected to the MySQL database, created the student table, and inserted multiple records. It also retrieved and displayed the data correctly in the browser, demonstrating proper database interaction and data handling.

EX.NO:12 **NAME:**

DATE: **REG NO:**

Working with queries and joins and sub queries in MySQL

AIM

To demonstrate the use of SELECT queries, JOINs, and Subqueries in MySQL for retrieving and combining data from multiple tables.

PROCEDURE

1. **Create Databases and Tables**
    - Create a sample database: studentDB.
    - Create two tables:
        1. students (id, name, age, class)
        2. marks (id, student_id, subject, marks)
2. **Insert Sample Data**
    - Insert records into both tables using INSERT INTO.
3. **Querying Data**
    - Retrieve all records using SELECT * FROM table_name.
    - Use WHERE to filter records based on conditions.
4. **Using Joins**
    - Use **INNER JOIN** to combine students and marks tables based on student_id.
    - Optionally use LEFT JOIN or RIGHT JOIN for different purposes.
5. **Using Subqueries**
    - Use subqueries to find students with marks greater than average or highest marks.
6. Execute queries in MySQL (via CLI or phpMyAdmin) and verify results.

PROGRAM

```
CREATE DATABASE IF NOT EXISTS studentDB;

USE studentDB;
```

```
CREATE TABLE IF NOT EXISTS students (
   id INT AUTO_INCREMENT PRIMARY KEY,
   name VARCHAR(50),
   age INT,
   class VARCHAR(10));

CREATE TABLE IF NOT EXISTS marks (
   id INT AUTO_INCREMENT PRIMARY KEY,
   student_id INT,
   subject VARCHAR(50),
   marks INT,
   FOREIGN KEY(student_id) REFERENCES students(id));

INSERT INTO students (name, age, class) VALUES
('Anjali', 20, 'A'),
('Bhargavi', 21, 'B'),
('Chitra', 19, 'A');
INSERT INTO marks (student_id, subject, marks) VALUES
(1, 'Math', 90),
(1, 'English', 85),
(2, 'Math', 70),
(2, 'English', 75),
(3, 'Math', 95),
(3, 'English', 80);

SELECT * FROM students;
-- INNER JOIN to get student names with their marks
SELECT s.name, m.subject, m.marks
FROM students s
INNER JOIN marks m ON s.id = m.student_id;
```

```
-- Subquery: Students with marks above 80
SELECT name FROM students
WHERE id IN (
   SELECT student_id FROM marks WHERE marks > 80);
-- Subquery to find student with highest Math marks
SELECT name FROM students
WHERE id = (
   SELECT student_id FROM marks WHERE subject='Math' ORDER BY marks DESC LIMIT 1
);
```

OUTPUT

1. Simple SELECT Query

id | name | age | class

1 | Anjali | 20 | A

2 | Bhargavi | 21 | B

3 | Chitra | 19 | A

2. INNER JOIN Output

name | subject | marks

Anjali | Math | 90

Anjali | English | 85

Bhargavi | Math | 70

Bhargavi | English | 75

Chitra | Math | 95

Chitra | English | 80

3. Subquery – Marks above 80

name

Anjali

Chitra

4. Subquery – Highest Math Marks

name

Chitra

RESULT

Successfully executed SELECT queries, INNER JOINs, and subqueries to retrieve and combine data. All queries returned correct and expected results.

EX.NO:13 **NAME:**

DATE: **REG NO:**

Manipulation of cookies and sessions using PHP

AIM

To demonstrate the creation, retrieval, and deletion of **cookies** and **sessions** in PHP for maintaining user data across pages.

PROCEDURE

Cookies

1. Use setcookie() to create a cookie with a name, value, and expiration time.
2. Access cookie values using $_COOKIE['cookie_name'].
3. Delete a cookie by setting its expiration time in the past.

Sessions

1. Start a session using session_start().
2. Store data in session variables using $_SESSION['key'] = value.
3. Retrieve session data using $_SESSION['key'].
4. Destroy the session using session_destroy() to clear all session variables.
5. Display all outputs using echo.
6. Save the program as .php file and run it on a PHP server (XAMPP/WAMP/LAMP).

PROGRAM

```
<?php
// --- COOKIE DEMONSTRATION ---
// Set a cookie
setcookie("username", "Anjali", time() + 3600); // expires in 1 hour

echo "<h3>Cookies in PHP</h3>";
if(isset($_COOKIE['username'])){
   echo "Welcome, " . $_COOKIE['username'] . "<br>";
```

```
} else {
    echo "Cookie 'username' is not set.<br>";}
// To delete the cookie, uncomment the following line
// setcookie("username", "", time() - 3600);
// --- SESSION DEMONSTRATION ---
session_start(); // Start the session
$_SESSION['user'] = "Bhargavi";
$_SESSION['email'] = "bhargavi@example.com";
echo "<h3>Sessions in PHP</h3>";
echo "User: " . $_SESSION['user'] . "<br>";
echo "Email: " . $_SESSION['email'] . "<br>";
// To destroy the session, uncomment the following line
// session_destroy();   ?>
```

OUTPUT

Cookies in PHP

Welcome, Anjali

Sessions in PHP

User: Bhargavi

Email: bhargavi@example.com

RESULT

The program successfully demonstrated the creation, retrieval, and deletion of cookies and sessions in PHP. User data was stored and accessed correctly across pages using both methods.

EX.NO:14 **NAME:**

DATE: **REG NO:**

Form validation connecting with sql data using PHP functions

AIM

To validate user input from an HTML form and insert the data into a MySQL database using PHP functions.

PROCEDURE

1. **Create Database and Table**

   CREATE DATABASE studentDB;

   USE studentDB;

2. CREATE TABLE students (

   id INT AUTO_INCREMENT PRIMARY KEY,

   name VARCHAR(50) NOT NULL,

   email VARCHAR(50) NOT NULL,

   age INT NOT NULL);

3. **Create HTML Form**

   - Create a form with fields: Name, Email, Age, and Submit button.
   - Use POST method to submit data to a PHP script.

4. **Form Validation in PHP**

   - Check for empty fields using empty().
   - Validate email format using filter_var($email, FILTER_VALIDATE_EMAIL).
   - Validate numeric values for age using is_numeric().

5. **Database Connection**

   - Use a PHP function to connect to MySQL using mysqli_connect().
   - Use another function to insert validated form data into the database.

6. **Display messages**

   - Display success message on successful insertion.
   - Display error message if validation fails.

PROGRAM

HTML Form (form.html)

```
<!DOCTYPE html>
<html>
<head>
   <title>Student Form</title>
</head>
<body>
<h3>Student Registration</h3>
<form method="post" action="process.php">
   Name: <input type="text" name="name"><br><br>
   Email: <input type="text" name="email"><br><br>
   Age: <input type="text" name="age"><br><br>
   <input type="submit" name="submit" value="Submit">
</form>
</body>
</html>
```

PHP Script (process.php)

```
<?php
// Function to connect to database
function db_connect() {
   $conn = mysqli_connect("localhost", "root", "", "studentDB");
   if (!$conn) {
      die("Connection failed: " . mysqli_connect_error());
   }
   return $conn;
}

// Function to insert student data
function insert_student($name, $email, $age) {
```

```
    $conn = db_connect();
    $sql = "INSERT INTO students (name, email, age) VALUES ('$name', '$email', $age)";
    if (mysqli_query($conn, $sql)) {
        echo "Record inserted successfully!";
    } else {
        echo "Error: " . mysqli_error($conn);
    }
    mysqli_close($conn);
}
// Form Validation
if(isset($_POST['submit'])){
    $name = trim($_POST['name']);
    $email = trim($_POST['email']);
    $age = trim($_POST['age']);
    if(empty($name) || empty($email) || empty($age)){
        echo "All fields are required!";
    } elseif(!filter_var($email, FILTER_VALIDATE_EMAIL)){
        echo "Invalid email format!";
    } elseif(!is_numeric($age)){
        echo "Age must be numeric!";
    } else {
        insert_student($name, $email, $age);
    }
}?>
```

OUTPUT

Case 1 – Successful Submission:

Record inserted successfully!

Case 2 – Validation Failure:

All fields are required!

Invalid email format!

Age must be numeric!

Database Table After Submission:

id	name	email	age
1	Anjali	Anjali@mail.com	20
2	Bhargavi	bhargavi@mail.com	22

RESULT

The program successfully validated form inputs using PHP functions, connected to the MySQL database, and inserted data correctly. Validation errors were handled appropriately.

EX.NO:15 **NAME:**

DATE: **REG NO:**

Creating PHP web applications to manipulating data [CURD operations] from MYSQL table.

AIM

To create a PHP web application that performs **CRUD operations (Create, Read, Update, Delete)** on a MySQL database table.

PROCEDURE

- **Create Database and Table**

CREATE DATABASE studentDB;

USE studentDB;

CREATE TABLE students (

id INT AUTO_INCREMENT PRIMARY KEY,

name VARCHAR(50) NOT NULL,

email VARCHAR(50) NOT NULL,

age INT NOT NULL);

- **Create a PHP Script for Database Connection**

  Use mysqli_connect() to connect to the MySQL database.
- **Implement CRUD Operations**
- **Create (Insert)**: Add new student records using an HTML form.
- **Read (Select)**: Display all records from the table.
- **Update**: Modify existing records based on id.
- **Delete**: Remove records using id.
- **Use PHP Functions**
- Functions for insert, update, delete, and display data.
- **Display results** using HTML tables for clarity.
- Save all PHP files and run them on a PHP server (XAMPP/WAMP/LAMP).

PROGRAM

Database Connection (db.php)

```
<?php
$conn = mysqli_connect("localhost", "root", "", "studentDB");
if(!$conn){
   die("Connection failed: " . mysqli_connect_error());
}
?>
```

CRUD Operations (index.php)

```
<?php
include 'db.php';

// Insert
if(isset($_POST['submit'])){
   $name = $_POST['name'];
   $email = $_POST['email'];
   $age = $_POST['age'];
   mysqli_query($conn, "INSERT INTO students (name,email,age) VALUES
('$name','$email',$age)");
}

// Delete
if(isset($_GET['delete'])){
   $id = $_GET['delete'];
   mysqli_query($conn, "DELETE FROM students WHERE id=$id");
}

// Update
if(isset($_POST['update'])){
   $id = $_POST['id'];
   $name = $_POST['name'];
   $email = $_POST['email'];
   $age = $_POST['age'];
   mysqli_query($conn, "UPDATE students SET name='$name', email='$email',
age=$age WHERE id=$id");
}

// Fetch All Records
$result = mysqli_query($conn, "SELECT * FROM students");
?>
<h3>Student Records</h3>
<table border="1">
<tr><th>ID</th><th>Name</th><th>Email</th><th>Age</th><th>Action</th></tr>
<?php while($row = mysqli_fetch_assoc($result)){ ?>
<tr>
<td><?php echo $row['id']; ?></td>
<td><?php echo $row['name']; ?></td>
<td><?php echo $row['email']; ?></td>
```

```
<td><?php echo $row['age']; ?></td>
<td>
<a href="index.php?delete=<?php echo $row['id']; ?>">Delete</a>
</td>
</tr>
<?php } ?>
</table>

<h3>Add Student</h3>
<form method="post">
Name: <input type="text" name="name"><br>
Email: <input type="text" name="email"><br>
Age: <input type="text" name="age"><br>
<input type="submit" name="submit" value="Add Student">
</form>
```

OUTPUT:

Student Records Table

ID	Name	Email	Age	Action
1	Alice	alice@mail.com	20	Delete
2	Bob	bob@mail.com	22	Delete

Add Student Form
Name: [_____]
Email: [_____]
Age: [_____]
[Add Student Button]

- Clicking **Delete** removes the record.
- Adding a new student updates the table dynamically.

RESULT:

The PHP web application successfully demonstrated all CRUD operations. New student records were added (Create), all existing records were displayed correctly (Read), records were updated as required (Update), and selected records were deleted (Delete). All operations were executed successfully, ensuring that data integrity was maintained in the MySQL table.