



**Department of Computer Applications**

**WEB PROGRAMMING (USA20501J) LAB MANUAL**

**For**

**BCA Degree Programme**

**2020 REGULATIONS**

**Academic Year (2022 – 2023)**

**(ODD) Semester**

**III Year V Semester**

**Prepared By**

**Dr.N.VIJAYALAKSHMI  
Mrs. S .SINDHU  
Dr.S.JAYACHANDRAN**

**Approved By**

**HOD**

## **INDEX**

<b>Ex. No</b>	<b>Date</b>	<b>Title</b>	<b>Page_No</b>	<b>Signature</b>
<b>1</b>		Working with files and directory commands		
<b>2</b>		Working with ls commands		
<b>3</b>		Working with chmod commands		
<b>4</b>		Simple PHP Programs		
<b>5</b>		Operators in PHP		
<b>6</b>		Creating Simple Webpage Using PHP		
<b>7</b>		Use of Conditional statement in PHP		
<b>8</b>		Different types of Array		
<b>9</b>		Usage of Array Function		
<b>10</b>		Creating User defined function		
<b>11</b>		Creating simple applications using PHP		
<b>12</b>		Creating simple table with constraints		
<b>13</b>		Insertion, Updation and Deletion of rows in MYSQL tables		
<b>14</b>		Searching of Data by different criteria		
<b>15</b>		Sorting of Data		
<b>16</b>		Working with string and date function		
<b>17</b>		Database connectivity in PHP with MYSQL		

**EX.NO: 01**

**NAME:**

**DATE:**

**REG NO:**

**Working with files and directory commands**

**Aim**

To Work with files and directories commands in Linux.

**CAT Command:**

cat linux command concatenates files and print it on the standard output.

**SYNTAX:**

The Syntax  
is

cat [OPTIONS] [FILE]...

**OPTIONS:**

- A Show all.
- b Omits line numbers for blank space in the output.
- e A \$ character will be printed at the end of each line prior to a new line.
- E Displays a \$ (dollar sign) at the end of each line.
- n Line numbers for all the output lines.
- s If the output has multiple empty lines it replaces it with one empty line.
- T Displays the tab characters in the output.
- Non-printing characters (with the exception of tabs, new-lines and form-feeds)
- v are printed visibly.

**Example:**

To Create a new file:

cat > file1.txt

This command creates a new file file1.txt. After typing into the file press control+d (^d) simultaneously to end the file.

1. To Append data into the

file: cat >> file1.txt

To append data into the same file use append operator >> to write into the file, else the file will be overwritten (i.e., all of its contents will be erased).

2. To display a file: catfile1.txt

This command displays the data in the file.

3. To concatenate several files and display: cat file1.txt file2.txt
4. The above cat command will concatenate the two files (file1.txt and file2.txt) and it will display the output in the screen. Some times the output may not fit the monitor screen. In such situation you can print those files in a new file or display the file using lesscommand.

```
cat file1.txt file2.txt | less
```

5. To concatenate several files and to transfer the output to another file.

```
cat file1.txt file2.txt > file3.txt
```

In the above example the output is redirected to new file file3.txt. The cat command will create new file file3.txt and store the concatenated output into file3.txt.

## **rm COMMAND:**

rm linux command is used to remove/delete the file from the directory.

## **SYNTAX:**

The Syntax is

```
rm [options..] [file | directory]
```

## **OPTIONS:**

- f Remove all files in a directory without prompting the user.
- i Interactive. With this option, rm prompts for confirmation before removing any files.

-r (or) -R Recursively remove directories and subdirectories in the argument list. The directory will be emptied of files and removed. The user is normally prompted for removal of any write-protected files which the directory contains.

### **EXAMPLE:**

1. To Remove / Delete a file:

rm file1.txt

Here rm command will remove/delete the file file1.txt.

2. To delete a directory tree:

rm -ir tmp

This rm command recursively removes the contents of all subdirectories of the tmp directory, prompting you regarding the removal of each file, and then removes the tmp directory itself.

3. To remove more files at once

rm file1.txt file2.txt

rm command removes file1.txt and file2.txt files at the same time.

### **cd COMMAND:**

cd command is used to change the directory.

### **SYNTAX:**

The Syntax is

cd [directory | ~ | ./ | ../ | - ]

### **OPTIONS:**

- L Use the physical directory structure.
- P Forces symbolic links.

**EXAMPLE:**1. cd linux-command

This command will take you to the sub-directory(linux-command) from its parent directory.

2. cd ..

This will change to the parent-directory from the current working directory/sub-directory.

3. cd ~

This command will move to the user's home directory which is "/home/username".

**cp COMMAND:**

cp command copy files from one location to another. If the destination is an existing file, then the file is overwritten; if the destination is an existing directory, the file is copied into the directory (the directory is not overwritten).

**SYNTAX:**

The Syntax is

cp [OPTIONS]... SOURCE DEST

cp [OPTIONS]... SOURCE... DIRECTORY

cp [OPTIONS]... --target-directory=DIRECTORY SOURCE...

**OPTIONS:**

-a	same as -dpR.
--backup[=CONTROL]	make a backup of each existing destination file
-b	like --backup but does not accept an argument.
-f	if an existing destination file cannot be opened, remove it and try again.
-p	same as --preserve=mode,ownership,timestamps.

-- preserve[=ATTR_LIST]	preserve the specified attributes (default: mode,ownership,timestamps) and security contexts, if possible additional attributes: links, all.
--no- preserve=ATTR_LIST	don't preserve the specified attribute.
--parents	append source path to DIRECTORY.

### **EXAMPLE:**

Copy two

Files: cp file1 file2

The above cp command copies the content of file1.php to file2.php.

1. To backup the copied

file: cp -b file1.php

file2.php

Backup of file1.php will be created with '~' symbol as file2.php~.

2. Copy folder and

subfolders: cp -R scripts

scripts1

The above cp command copy the folder and subfolders from scripts to scripts1.

### **RESULT:**

Thus the program has been successfully completed & executed.

**EX.NO: 02**

**NAME:**

**DATE:**

**REG NO:**

### **Working with ls commands**

#### **ls COMMAND:**

ls command lists the files and directories under current working directory.

#### **SYNTAX:**

The Syntax  
is

ls [OPTIONS]... [FILE]

#### **OPTIONS:**

- l Lists all the files, directories and their mode, Number of links, owner of the file, file size, Modified date and time and filename.
- t Lists in order of last modification time.
- a Lists all entries including hidden files.
- d Lists directory files instead of contents.
- p Puts slash at the end of each directories.
- u List in order of last access time.
- i Display inode information.
- ltr List files order by date.
- lsr List files order by file size.

#### **EXAMPLE:**

Display root directory contents:ls /

lists the contents of root directory.

1. Display hidden files and directories:ls -a

lists all entries including hidden files and directories.

2. Display inode information:ls -li

7373073 book.gif

7373074 clock.gif



7373082 globe.gif

7373078 pencil.gif

7373080 child.gif

7373081 email.gif

7373076 indigo.gif

The above command displays filename with inode value.

## **ln COMMAND:**

ln command is used to create link to a file (or) directory. It helps to provide soft link for desired files. Inode will be different for source and destination.

## **SYNTAX:**

The Syntax is

ln [options] existingfile(or directory)name newfile(or directory)name

## **OPTIONS:**

- f Link files without questioning the user, even if the mode of target forbids writing. This is the default if the standard input is not a terminal.
- n Does not overwrite existing files.
- s Used to create soft links.

## **EXAMPLE:**

1. ln -s file1.txt file2.txt

Creates a symbolic link to 'file1.txt' with the name of 'file2.txt'. Here inode for 'file1.txt' and 'file2.txt' will be different.

2. ln -s nimi nimi1

Creates a symbolic link to 'nimi' with the name of 'nimi1'.

## **RESULT:**

Thus the program has been successfully completed & executed.

**EX.NO: 03**

**NAME:**

**DATE:**

**REG NO:**

### **Working with Chmod commands**

#### **SYNTAX:**

The Syntax is

chmod [options] [MODE] FileName

#### **File Permission**

#	File Permission
0	none
1	execute only
2	write only
3	write and execute
4	read only
5	read and execute
6	read and write
7	set all permissions

#### **OPTIONS:**

- c        Displays names of only those files whose permissions are being changed
- f        Suppress most error messages
- R        Change files and directories recursively
- v        Output version information and exit.

#### **EXAMPLE:**

1. To view your files with what permission they are:ls -alt

This command is used to view your files with what permission they are.

2. To make a file readable and writable by the group and others.

chmod 066 file1.txt

3. To allow everyone to read, write, and execute the file

```
chmod 777 file1.txt
```

**RESULT:**

Thus the program has been successfully completed & executed.

**EX.NO: 04**

**NAME:**

**DATE:**

**REG NO:**

### **SIMPLE PHP PROGRAM**

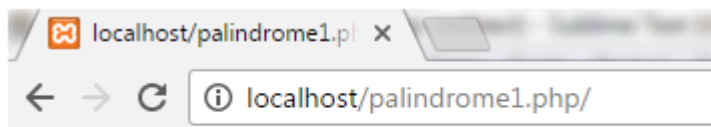
**AIM:**

To create a simple Program using php.

**PROGRAM:**

```
<?php
function palindrome($n){
$number = $n;
$sum = 0;
while(floor($number)) {
$rem = $number % 10;
$sum = $sum * 10 + $rem;
$number = $number/10;
}
return $sum;
}
$input = 1235321;
$num = palindrome($input);
if($input==$num){
echo "$input is a Palindrome number";
} else {
echo "$input is not a Palindrome";
}
?>
```

**OUTPUT:**



1235321 is a Palindrome number

**RESULT:**

Thus the program has been successfully completed & executed.

**EX.NO: 05**

**NAME:**

**DATE:**

**REG NO:**

## **OPERATORS IN PHP**

### **AIM:**

To write a PHP Program for various Operators.

### **a) PROGRAM:**

```
<?php
$a = 80;
$b = 50;
$c = "80";
var_dump($a == $c) + "\n";
var_dump($a != $b) + "\n";
var_dump($a <> $b) + "\n";
var_dump($a === $c) + "\n";
var_dump($a !== $c) + "\n";
var_dump($a < $b) + "\n";
var_dump($a > $b) + "\n";
var_dump($a <= $b) + "\n";
var_dump($a >= $b);
?>
```

### **OUTPUT:**

```
bool(true)
bool(true)
bool(true)
bool(false)
bool(true)
bool(false)
bool(true)
bool(false)
bool(true)
```

### **b) PROGRAM:**

```
<?php
```

```
$x = 2;  
echo ++$x, " First increments then prints \n";  
echo $x, "\n";
```

```
$x = 2;  
echo $x++, " First prints then increments \n";  
echo $x, "\n";
```

```
$x = 2;  
echo --$x, " First decrements then prints \n";  
echo $x, "\n";
```

```
$x = 2;  
echo $x--, " First prints then decrements \n";  
echo $x;  
?>
```

### **OUTPUT:**

```
3 First increments then prints  
3  
2 First prints then increments  
3  
1 First decrements then prints  
1  
2 First prints then decrements  
1
```

### **RESULT:**

Thus the program has been successfully completed & executed.

**EX.NO: 06**

**NAME:**

**DATE:**

**REG NO:**

**CREATING SIMPLE WEBPAGE USING PHP**

**AIM:**

To create a simple web page using php.

**PROGRAM:**

```
<!DOCTYPE html>
<html>
<title>new.com</title>
<style>
body
{
font-style:italic;
font-size:20px;
font-weight:bold;
}
.container
{
list-style-type: none;
border:2;
height:50px;
background-color:#555550;
color:red;
font-style:italic;
font-weight:bold;
font-size:35px;
text-align:center;
text-decoration:none;
margin : 0;
padding : 0;
overflow: hidden;
}
a:hover{color:red;
}
U1
{
list-style-type: none;
```

```
margin: 0;
padding: 0;
overflow: hidden;
background-color:black
}
li {
float: left;
}
li a{
display: block;
color: white;
text-align: center;
padding: 14px 16px;
text-decoration: none;
}
li a:hover {
background-color:green;
}
.active {
background-color:#555550;
font-size:30px;
font-weight:bold;
font-size:50px;
text-align:center;
text-decoration:none;
margin : 0;
padding : 0;
overflow: hidden;
}
a.class1 {color:red;}
a.class1:link {text-decoration: none; color:skyblue;}
a.class1:visited {text-decoration: none; color: black;}
a.class1:hover {text-decoration: underline; color: red;}
a.class1:active {text-decoration: none; color: skyblue;}
.button1 {background-color: #555555;}
div.img {
margin: 5px;
border: 2px solid #ccc;
float: left;
width:300px;
height:280px;
}
```



```

div.img:hover {
    border: 2px solid #777;
}
div.img img {
    width: 100%;
    height: auto;
}
div.desc {
    padding: 10px;
    text-align: center;
}
</style>
<body>
<div class="container">MobileNew.com</div>
<form><pre>
<ul> <li><a href="project.html">Home</a></li><li><a href="news">News</a></li><li><a
href="contact">Contact</a></li><li><a href="about">About</a></li><li><a
href="login.html">Login</a></li><li><a href="signup.html">SignUp</a></li>      <input
type="text" name="Search" placeholder="search.."></ul><br><br>
                <a class="active" href="below20k.html" style="color:skyblue;"> Phones Under
20k </a>
                <a class="active" href="above20k.html" style="color:skyblue;"> Phones Above
20k </a>
</pre></form></body></html>

```

### **OUTPUT:**



### **RESULT:**

Thus the program has been successfully completed & executed.

**EX.NO: 07**

**NAME:**

**DATE:**

**REG NO:**

## **USE OF CONDITIONAL STATEMENTS IN PHP**

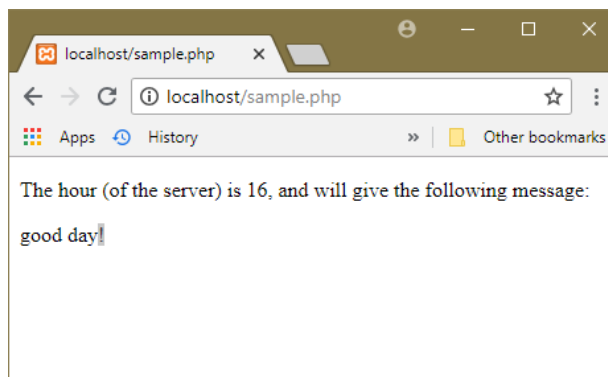
### **AIM:**

To create a php program using conditional statements

### **PROGRAM:**

```
<!DOCTYPE html>
<html>
<body>
<?php
$t = date("H");
echo "<p>The hour (of the server) is " . $t;
echo ", and will give the following message:</p>";
if ($t < "10") {
echo " good morning!";
} elseif ($t < "20") {
echo " good day!";
} else {
echo " good night!";
}
?>
</body>
</html>
```

### **OUTPUT:**



### **RESULT:**

Thus the program has been successfully completed & executed.

**EX.NO: 7A**

**NAME:**

**DATE:**

**REG NO:**

## **USE OF LOOPING STATEMENTS IN PHP**

### **FOR LOOP**

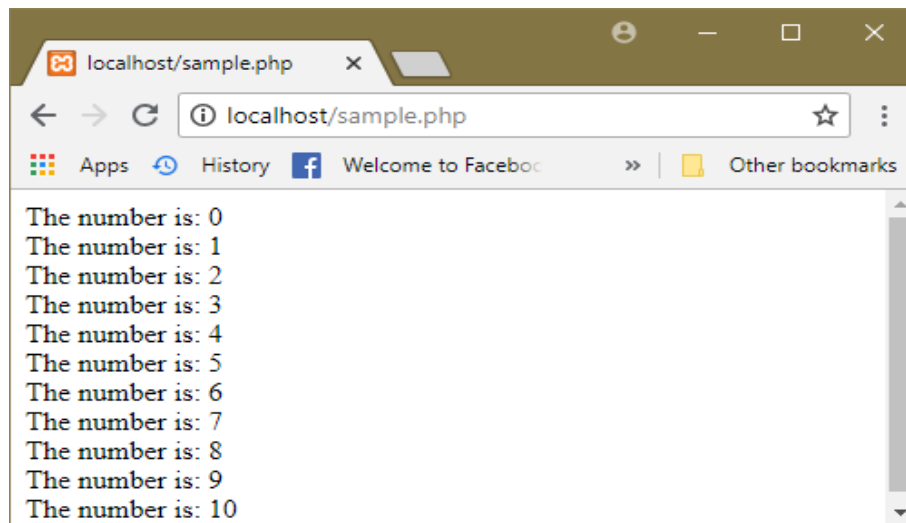
**AIM:**

To create a php program using FOR loop statement

**PROGRAM:**

```
<!DOCTYPE html> <html> <body>
<?php
for ($x = 0; $x <= 10; $x++) {
echo "The number is: $x <br>";
}
?>
</body> </html>
```

**OUTPUT:**



**RESULT:**

Thus the program has been successfully completed & executed.

**EX.NO: 7B**

**NAME:**

**DATE:**

**REG NO:**

## **USE OF LOOPING STATEMENTS IN PHP**

### **WHILE LOOP**

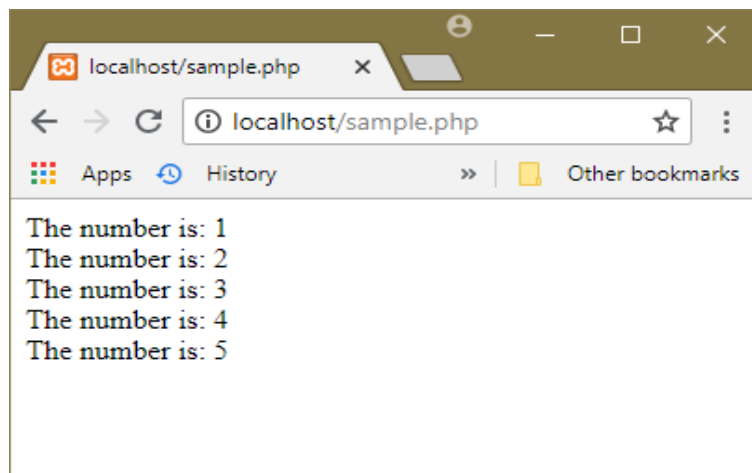
**AIM:**

To create a php program using WHILE loop statement

**PROGRAM:**

```
<!DOCTYPE html> <html> <body>
<?php
$x = 1;
while($x <= 5) {
echo "The number is: $x <br>";
    $x++;
}
?>
</body></html>
```

**OUTPUT:**



**RESULT:**

Thus the program has been successfully completed & executed.

**EX.NO: 7C**

**NAME:**

**DATE:**

**REG NO:**

## **USE OF LOOPING STATEMENTS IN PHP**

### **DO WHILE LOOP**

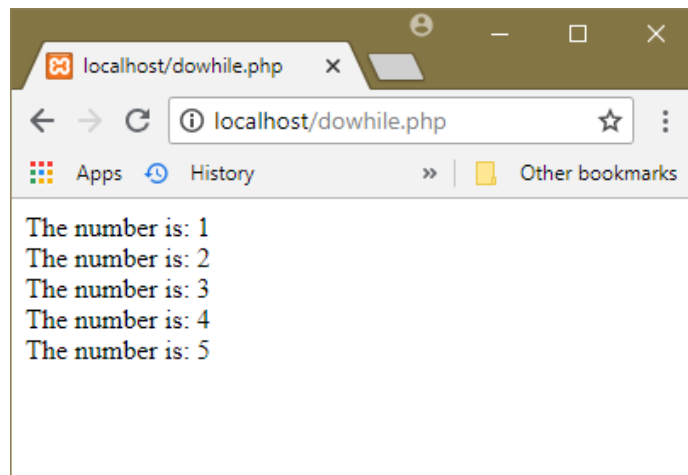
**AIM:**

To create a php program using DO WHILE loop statement

**PROGRAM:**

```
<!DOCTYPE html> <html> <body>
<?php
$x = 1;
do {
echo "The number is: $x <br>";
    $x++;
} while ($x <= 5);
?>
</body> </html>
```

**OUTPUT:**



**RESULT:**

Thus the program has been successfully completed & executed.

**EX.NO: 8**

**NAME:**

**DATE:**

**REG NO:**

## **CREATING DIFFERENT TYPES OF ARRAYS**

### **SINGLE DIMENSIONAL ARRAY**

**AIM:**

To create a php program using Single dimensional array

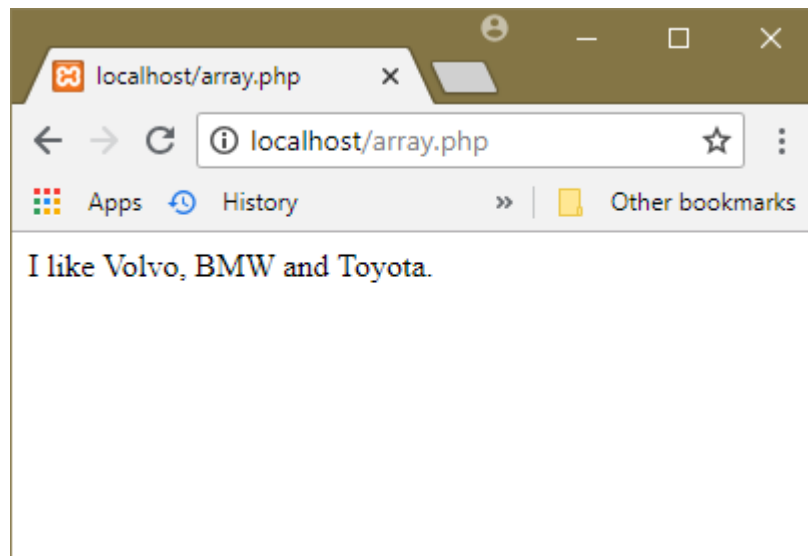
**PROGRAM:**

```
<!DOCTYPE html> <html> <body>

<?php
$cars = array("Volvo", "BMW", "Toyota");
echo "I like " . $cars[0] . ", " . $cars[1] . " and " . $cars[2] . ".";
?>

</body> </html>
```

**OUTPUT:**



**RESULT:**

Thus the program has been successfully completed & executed.

**EX.NO: 8A**

**NAME:**

**DATE:**

**REG NO:**

## **CREATING DIFFERENT TYPES OF ARRAYS**

### **MULTIDIMENSIONAL ARRAY**

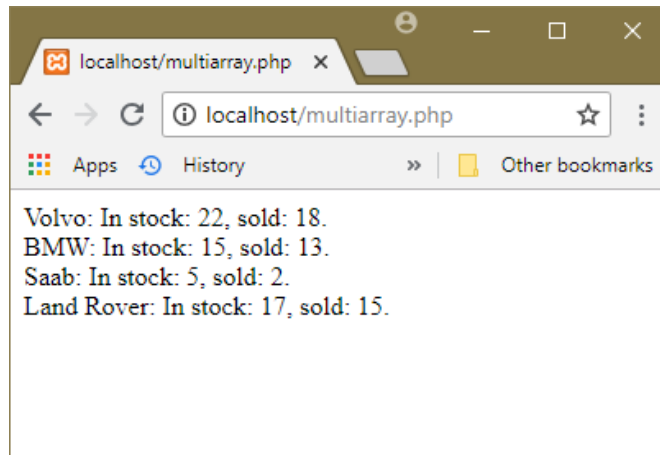
**AIM:**

To create a php program using Multiple dimensional array

**PROGRAM:**

```
<!DOCTYPE html> <html> <body>
<?php
$cars = array (array("Volvo",22,18),
               array("BMW",15,13),
               array("Saab",5,2),
               array("Land Rover",17,15) );
echo $cars[0][0].": In stock: ".$cars[0][1].", sold: ".$cars[0][2]."<br>";
echo $cars[1][0].": In stock: ".$cars[1][1].", sold: ".$cars[1][2]."<br>";
echo $cars[2][0].": In stock: ".$cars[2][1].", sold: ".$cars[2][2]."<br>";
echo $cars[3][0].": In stock: ".$cars[3][1].", sold: ".$cars[3][2]."<br>";
?>
</body> </html>
```

**OUTPUT:**



**RESULT:**

Thus the program has been successfully completed & executed.

**EX.NO: 9**

**NAME:**

**DATE:**

**REG NO:**

## **USAGE OF ARRAY FUNCTIONS**

**(A)**

### **ARRAY USING RANGE**

**AIM:**

To create a php program with usage of array functions

**PROGRAM:**

```
<?php
$fruit = array("orange", "pineapple", "peach", "apple", "pear", "cherry");
sort($fruit);
print_r($fruit);
echo "<br>";
echo count($fruit)."<br>";
$range_array = range(0, 50, 5);
print_r($range_array);
?>
```

**OUTPUT:**

```
Array ( [0] => apple [1] => cherry [2] => orange [3] => peach [4] => pear [5] => pineapple )
6
Array ( [0] => 0 [1] => 5 [2] => 10 [3] => 15 [4] => 20 [5] => 25 [6] => 30 [7] => 35 [8] => 40 [9]
=> 45 [10] => 50)
```

### **(B) ARRAY USING KEYS AND VALUS**

**PROGRAM:**

```
<?php
$characters['pig'] = "Porky Pig";
$charctes['duck'] = "Daffy Duck";
$characters['mouse'] = "Speedy Gonzales";
foreach($characters as $key=>$value){
echo $value."isa".$key."<br/>";
}?>
```

**OUTPUT:**

```
Porky Pig is a pig.
Daffy Duck is a duck.
Speedy Gonzales is a mouse.
```



**(C)**

### **ARRAY USING ARRAY CHANGE KEY CASE**

#### **PROGRAM:**

```
<?php
$input_array = array("FirSt" => 1, "SecOnd" => 4);
print_r(array_change_key_case($input_array, CASE_UPPER));
?>
```

#### **OUTPUT:**

```
( [FIRST] => 1
  [SECOND] => 4 )
```

**(D)**

### **ARRAY USING ARSORT()**

#### **PROGRAM:**

```
<?php
$fruits = array("d" => "lemon", "a" => "orange", "b" => "banana", "c" => "apple");

arsort($fruits);

foreach ($fruits as $key => $val)
{
    echo "$key = $val\n";
}
?>
```

#### **OUTPUT:**

```
a = orange
d = lemon
b = banana
c = apple
```

**(E)**

**ARRAY USING COUNT()**

**PROGRAM:**

```
<?php
$a[0] = 1;
$a[1] = 3;
$a[2] = 5;
$result = count($a);
echo $result;

$b[0] = 7;
$b[5] = 9;
$b[10] = 11;
$b[14] = 20;
$result = count($b);
echo $result;

$result = count(null);
echo $result;

$result = count(false);
echo $result;
?>
```

**OUTPUT:**

```
$result == 3
$result == 4
$result == 0
$result == 1
```

**RESULT:**

Thus the program has been successfully completed & executed

**EX.NO: 10**

**NAME:**

**DATE:**

**REG NO:**

## **CREATING USER DEFINED FUNCTIONS**

### **AIM:**

To create a php program using user defined functions

### **PROGRAM:**

*//Calling function within a function or inner function*

```
function add($a,$b)
{
    return $a+$b;
}

function sub($a,$b

    return $a-$b;
}

function math($first, $second)
{
    $res = add($first, $second)/sub($first, $second);
    return (int)$res;
}

echo math(200,100);
```

### **OUTPUT:**

Results: 3

### **RESULT:**

Thus the program has been successfully completed & executed

**EX.NO: 11**

**NAME:**

**DATE:**

**REG NO:**

**CREATING SIMPLE APPLICATIONS USING USING PHP**

**AIM:**

To create a simple application using php.

**PROGRAM:**

**Signup.html**

```
<html>
<style>
h1
{
text-align:center;
color:#00f5f5;
}
#space
{
height:400px;
width:300px;
}
#hi
{
left:500px;
top:200px;
height:400px;
width:350px;
border:1px #ff00f0 solid;
position:relative;
display:block;
float:left;
}
</style>
<body>
```

```

<pre>
<h1>Sign up</h1>
<div id="hi">
<form method='post' action='insert.php'>
<pre>
User Name      : <input type="text" name="name"/>
Password       : <input type="text" name="pass"/>
Confirm Password : <input type="text" name="pass2"/>
Gender         : <input type="radio" name="gender" value="male"/>Male <input type="radio"
name="gender" value="female"/>Female
<input type="submit" name="add"/>
</pre>
</form>
</div>
</pre>
</body>
</html>

```

### **insert.php**

```

<?php
if(isset($_POST['add']))
{
$con=mysql_connect('localhost','root','');//connect to localhost
if(!$con)
{
die("failed".mysql_error());
}
$name=$_POST['name'];
$pass=$_POST['pass'];//getting password from webpage
$gender=$_POST['gender'];
if(!$gender)
{
$msg="enter gender";
echo "<script>alert('$msg');</script>";
}
mysql_select_db('test');

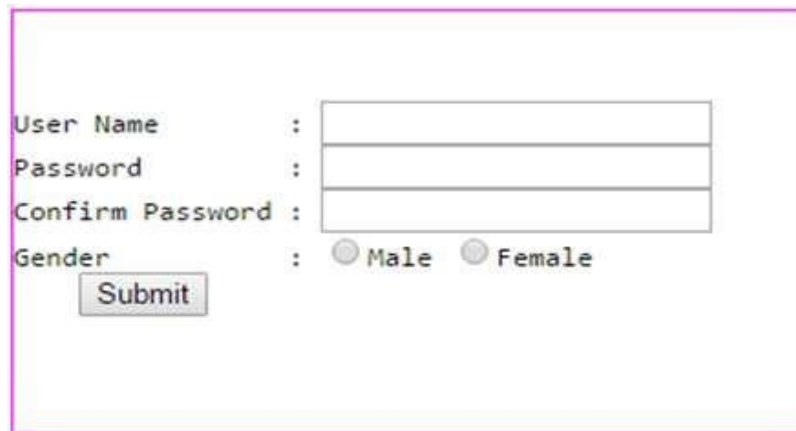
```

```

if($_POST['pass']==$_POST['pass2'])
{
$sql="INSERT INTO
userpass"."(user,pass,gender)". "VALUES"."('$name','$pass',$_POST[gender])";
$retval = mysql_query($sql,$con);
if(!$retval)
{
die('Could not enter data: ' . mysql_error());
}
echo "Entered data successfully";
header("location:password.php");
}
else
{
echo "PASSWORDS MISMATCH";
}
mysql_close($con);
}
?>

```

### **OUTPUT:**



User Name :

Password :

Confirm Password :

Gender : ☐ Male ☐ Female

PASSWORDS MISMATCH

### **RESULT:**

Thus the program has been successfully completed & executed

**EX.NO: 12**

**NAME:**

**DATE:**

**REG NO:**

**CREATING SIMPLE TABLE WITH CONSTRAINTS**

**AIM:**

To create a simple table using constraints

**PROGRAM:**

```
CREATE TABLE Persons
```

```
(  
  PersonID int,  
  LastName varchar(255),  
  FirstName varchar(255),  
  Address varchar(255),  
  City varchar(255)  
);
```

**SOL NOT NULL Constraint**

```
CREATE TABLE PersonsNotNull
```

```
(  
  P_Id int NOT NULL,  
  LastName varchar(255) NOT NULL,  
  FirstName varchar(255),  
  Address varchar(255),  
  City varchar(255)  
);
```

**SOL UNIQUE Constraint on CREATE TABLE**

```
CREATE TABLE Persons
```

```
(  
  P_Id int NOT NULL,  
  LastName varchar(255) NOT NULL,  
  FirstName varchar(255),  
  Address varchar(255),  
  City varchar(255),  
  UNIQUE (P_Id)  
);
```

### **SOL PRIMARY KEY Constraint on CREATE TABLE**

```
CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
PRIMARY KEY (P_Id)
);
```

### **SOL FOREIGN KEY Constraint**

```
CREATE TABLE Orders
(
O_Id int NOT NULL,
OrderNo int NOT NULL,
P_Id int,
PRIMARY KEY (O_Id),
FOREIGN KEY (P_Id) REFERENCES Persons(P_Id)
);
```

### **SOL CHECK Constraint on CREATE TABLE**

```
CREATE TABLE Persons
(
P_Id int NOT NULL,
LastName varchar(255) NOT NULL,
FirstName varchar(255),
Address varchar(255),
City varchar(255),
CHECK (P_Id>0)
);
```



### **SOL DEFAULT Constraint on CREATE TABLE**

```
CREATE TABLE Persons  
(  
P_Id int NOT NULL,  
LastName varchar(255) NOT NULL,  
FirstName varchar(255),  
Address varchar(255),  
City varchar(255) DEFAULT 'Sandnes'  
);
```

### **OUTPUT:**

Table created.

### **RESULT:**

Thus the program has been successfully completed & executed

**EX.NO: 13**

**NAME:**

**DATE:**

**REG NO:**

## **INSERTION, UPDATION AND DELETION OF ROWS IN MYSQL TABLES**

### **AIM:**

To insert, update and delete rows using MYSQL

### **(A) INSERTING DATA**

The INSERT statement is used to insert data into tables.

We will create a new table, where we will do our examples.

```
mysql> CREATE TABLE Books(Id INTEGER PRIMARY KEY, Title VARCHAR(100),  
Author VARCHAR(60));
```

We create a new table Books, with Id, Title and Author columns.

```
mysql> INSERT INTO Books(Id, Title, Author) VALUES(1, 'War and Peace', 'Leo Tolstoy');
```

This is the classic INSERT SQL statement. We have specified all column names after the table name and all values after the VALUES keyword.

We add our first row into the table.

```
mysql> SELECT * FROM Books;  
+----+-----+-----+  
| Id | Title      | Author  |  
+----+-----+-----+  
| 1  | War and Peace | Leo Tolstoy |  
+----+-----+-----+
```

We have inserted our first row into the Books table.

```
mysql> INSERT INTO Books(Title, Author) VALUES ('The Brothers Karamazov',  
-> 'Fyodor Dostoyevsky');
```

We add a new title into the Books table. We have omitted the Id column. The Id column has AUTO\_INCREMENT attribute. This means that MySQL will increase the Id column automatically. The value by which the AUTO\_INCREMENT column is increased is controlled by auto\_increment\_increment system variable. By default it is 1.

```
mysql> SELECT * FROM Books;  
+----+-----+-----+  
| Id | Title          | Author          |  
+----+-----+-----+  
| 1  | War and Peace  | Leo Tolstoy     |  
| 2  | The Brothers Karamazov | Fyodor Dostoyevsky |  
+----+-----+-----+
```

Here is what we have in the Books table.

```
mysql> INSERT INTO Books VALUES(3, 'Crime and Punishment',  
-> 'Fyodor Dostoyevsky');
```

In this SQL statement, we did not specify any column names after the table name. In such a case, we have to supply all values.

```
mysql> REPLACE INTO Books VALUES(3, 'Paradise Lost', 'John Milton');  
Query OK, 2 rows affected (0.00 sec)
```

The REPLACE statement is a MySQL extension to the SQL standard. It inserts a new row or replaces the old row if it collides with an existing row. In our table, there is a row with Id=3. So our previous statement replaces it with a new row. There is a message that two rows were affected. One row was deleted and one was inserted.

```
mysql> SELECT * FROM Books WHERE Id=3;  
+----+-----+-----+
```

Id	Title	Author	
+.....+		+.....+	
3	Paradise Lost	John Milton	
+.....+		+.....+	

This is what we have now in the third column.

We can use the INSERT and SELECT statements together in one statement.

```
mysql> CREATE TABLE Books2(Id INTEGER PRIMARY KEY AUTO_INCREMENT,
-> Title VARCHAR(100), Author VARCHAR(60)) type=MEMORY;
```

First, we create a temporary table called Books2 in memory.

```
mysql> INSERT INTO Books2 SELECT * FROM Books;
Query OK, 3 rows affected (0.00 sec)
Records: 3 Duplicates: 0 Warnings: 0
```

Here we insert all data into the Books2 that we select from the Books table.

```
mysql> SELECT * FROM Books2;
```

Id	Title	Author	
+.....+		+.....+	
1	War and Peace	Leo Tolstoy	
2	The Brothers Karamazov	Fyodor Dostoyevsky	
3	Paradise Lost	John Milton	
+.....+		+.....+	

We verify it. All OK.

```
mysql> INSERT INTO Books(Title, Author) VALUES ('The Insulted and Humiliated', 'Fyodor Dostoyevsky'), ('Cousin Bette', 'Honore de Balzac');
Query OK, 2 rows affected (0.00 sec)
Records: 2 Duplicates: 0 Warnings: 0
```

## **(B)**

## **DELETING DATA**

In MySQL, we can delete data using the DELETE and TRUNCATE statements.

The TRUNCATE statement is a MySQL extension to the SQL specification. First, we are going to delete one row from a table. We will use the Books2 table that we have created previously.

```
mysql> DELETE FROM Books2 WHERE Id=1;
```

We delete a row with Id=1.

```
mysql> SELECT * FROM Books2;
+----+-----+-----+
| Id | Title           | Author           |
+----+-----+-----+
| 2  | The Brothers Karamazov | Fyodor Dostoyevsky |
| 3  | Paradise Lost       | John Milton       |
+----+-----+-----+
```

We verify the data.

```
mysql> DELETE FROM Books2;
mysql> TRUNCATE Books2;
```

These two SQL statements delete all data in the table.

**(C)**

## **UPDATING DATA**

The UPDATE statement is used to change the value of columns in selected rows of a table.

```
mysql> SELECT * FROM Books;
```

+	+	+	+
Id	Title	Author	
+	+	+	+
1	War and Peace	Leo Tolstoy	
2	The Brothers Karamazov	Fyodor Dostoyevsky	
3	Paradise Lost	John Milton	
4	The Insulted and Humiliated	Fyodor Dostoyevsky	
5	Cousin Bette	Honore de Balzac	
+	+	+	+

We recreate the table Books. These are the rows.

Say we wanted to change 'Leo Tolstoy' to 'Lev Nikolayevich Tolstoy' table. The following statement shows, how to accomplish this.

```
mysql> UPDATE Books SET Author='Lev Nikolayevich Tolstoy'  
-> WHERE Id=1;
```

The SQL statement sets the author column to 'Lev Nikolayevich Tolstoy' for the column with Id=1.

```
mysql> SELECT * FROM Books WHERE Id=1;
```

+	+	+	+
Id	Title	Author	
+	+	+	+
1	War and Peace	Lev Nikolayevich Tolstoy	
+	+	+	+

The row is correctly updated.

### **RESULT:**

Thus the queries has been successfully completed & executed

**EX.NO: 14**

**NAME:**

**DATE:**

**REG NO:**

## **SEARCHING OF DATA BY DIFFERENT CRITERIA**

### **AIM:**

To search data by different criteria using MYSQL

### **RETRIEVING INFORMATION FROM A TABLE**

The SELECT statement is used to pull information from a table. The general form of the statement is:

```
SELECT what_to_select
FROM which_table
WHERE conditions_to_satisfy;
```

#### **(A) SELECTING ALL DATA**

The simplest form of SELECT retrieves everything from a table:

```
mysql> SELECT * FROM pet;
```

UPDATE statement:

```
mysql> UPDATE pet SET birth = '1989-08-31' WHERE name = 'Bowser';
```

if you want to verify the change that you made to Bowser's birth date, select Bowser's record like this:

```
mysql> SELECT * FROM pet WHERE name = 'Bowser';
```

if you want to know which animals were born during or after 1998, test the birth column:

```
mysql> SELECT * FROM pet WHERE birth >= '1998-1-1';
```

## **(B) SELECTING PARTICULAR COLUMNS**

For example, if you want to know when your animals were born, select the name and birth columns:

```
mysql> SELECT name, birth FROM pet;
```

To find out who owns pets, use this query:

```
mysql> SELECT owner FROM pet;
```

To minimize the output, retrieve each unique output record just once by adding the keyword DISTINCT:

```
mysql> SELECT DISTINCT owner FROM pet;
```

For example, to get birth dates for dogs and cats only, use this query:

```
mysql> SELECT name, species, birth FROM pet  
-> WHERE species = 'dog' OR species = 'cat';
```

## **(C) SORTING ROWS**

To sort a result, use an ORDER BY clause.

Here are animal birthdays, sorted by date:

```
mysql> SELECT name, birth FROM pet ORDER BY birth;
```

To sort in reverse (descending) order, add the DESC keyword to the name of the column you are sorting by:

```
mysql> SELECT name, birth FROM pet ORDER BY birth DESC;
```

For example, to sort by type of animal in ascending order, then by birth date within animal type in descending order (youngest animals first), use the following query:

```
mysql> SELECT name, species, birth FROM pet  
-> ORDER BY species, birth DESC;
```



## (D) WORKING WITH NULL VALUES

The NULL value can be surprising until you get used to it. Conceptually, NULL means “a missing unknown value” and it is treated somewhat differently from other values.

To test for NULL, use the IS NULL and IS NOT NULL operators, as shown here:

```
mysql> SELECT 1 IS NULL, 1 IS NOT NULL;
```

## (E) PATTERN MATCHING

MySQL provides standard SQL pattern matching as well as a form of pattern matching based on extended regular expressions similar to those used by Unix utilities such as **vi**, **grep**, and **sed**.

You do not use = or <> when you use SQL patterns;

use the LIKE or NOT LIKE comparison operators instead.

To find names beginning with “b”:

```
mysql> SELECT * FROM pet WHERE name LIKE 'b%';
+-----+-----+-----+-----+-----+-----+
| name  | owner | species | sex | birth   | death   |
+-----+-----+-----+-----+-----+-----+
| Buffy | Harold | dog     | f   | 1989-05-13 | NULL    |
| Bowser | Diane | dog     | m   | 1989-08-31 | 1995-07-29 |
+-----+-----+-----+-----+-----+-----+
```

To find names ending with “fy”:

```
mysql> SELECT * FROM pet WHERE name LIKE '%fy';
+-----+-----+-----+-----+-----+-----+
| name  | owner | species | sex | birth   | death   |
+-----+-----+-----+-----+-----+-----+
| Fluffy | Harold | cat     | f   | 1993-02-04 | NULL    |
| Buffy | Harold | dog     | f   | 1989-05-13 | NULL    |
+-----+-----+-----+-----+-----+-----+
```

To find names containing a “w”:

```
mysql> SELECT * FROM pet WHERE name LIKE '%w%';
```

name	owner	species	sex	birth	death
Claws	Gwen	cat	m	1994-03-17	NULL
Bowser	Diane	dog	m	1989-08-31	1995-07-29
Whistler	Gwen	bird	NULL	1997-12-09	NULL

To find names containing exactly five characters, use five instances of the “\_” pattern character:

```
mysql> SELECT * FROM pet WHERE name LIKE '_____';
```

name	owner	species	sex	birth	death
Claws	Gwen	cat	m	1994-03-17	NULL
Buffy	Harold	dog	f	1989-05-13	NULL

## (F) COUNTING ROWS

Databases are often used to answer the question, “How often does a certain type of data occur in a table?” For example, you might want to know how many pets you have, or how many pets each owner has, or you might want to perform various kinds of census operations on your animals.

Counting the total number of animals you have is the same question as “How many rows are in the pet table?” because there is one record per pet. COUNT(\*) counts the number of rows, so the query to count your animals looks like this:

```
mysql> SELECT COUNT(*) FROM pet;
```

Earlier, you retrieved the names of the people who owned pets. You can use COUNT() if you want to find out how many pets each owner has:

```
mysql> SELECT owner, COUNT(*) FROM pet GROUP BY owner;
```

Number of animals per species:

```
mysql> SELECT species, COUNT(*) FROM pet GROUP BY species;
```

### **(G) USING MORE THAN ONE TABLE**

the CREATE TABLE statement for the event table might look like this:

```
mysql> CREATE TABLE event (name VARCHAR(20), date DATE,  
-> type VARCHAR(15), remark VARCHAR(255));
```

We saw earlier how to calculate ages from two dates. The litter date of the mother is in the event table, but to calculate her age on that date you need her birth date, which is stored in the pet table. This means the query requires both tables:

```
mysql> SELECT pet.name,  
-> TIMESTAMPDIFF(YEAR,birth,date) AS age,  
-> remark  
-> FROM pet INNER JOIN event  
-> ON pet.name = event.name  
-> WHERE event.type = 'litter';
```

### **RESULT:**

Thus the program has been successfully completed & executed.

**EX.NO: 15**

**NAME:**

**DATE:**

**REG NO:**

## **SORTING OF DATA**

### **AIM:**

To achieve sorting data using MYSQL

### **PROGRAM:**

```
root@host# mysql -u root -p password;
Enter password:*****
mysql> use TUTORIALS;
Database changed
mysql> SELECT * from tutorials_tbl ORDER BY tutorial_author ASC
```

### **TO VERIFY:**

```
<?php
$dbhost = 'localhost:3036';
$dbuser = 'root';
$dbpass = 'rootpassword';
$conn = mysql_connect($dbhost, $dbuser, $dbpass);
if(! $conn )
{
    die('Could not connect: ' . mysql_error());
}
$sql = 'SELECT tutorial_id, tutorial_title,
        tutorial_author, submission_date
        FROM tutorials_tbl
        ORDER BY tutorial_author DESC';

mysql_select_db('TUTORIALS');
$retval = mysql_query( $sql, $conn );
if(! $retval )
{
```

```

    die('Could not get data: ' . mysql_error());
}
while($row = mysql_fetch_array($retval, MYSQL_ASSOC))
{
    echo "Tutorial ID :{$row['tutorial_id']} <br> ".
        "Title: {$row['tutorial_title']} <br> ".
        "Author: {$row['tutorial_author']} <br> ".
        "Submission Date : {$row['submission_date']} <br> ".
        ".....<br>";
}
echo "Fetched data successfully\n";
mysql_close($conn);
?>

```

### **OUTPUT:**

```

+.....+.....+.....+.....+
| tutorial_id | tutorial_title | tutorial_author | submission_date |
+.....+.....+.....+.....+
|      2 | Learn MySQL   | Abdul S      | 2007-05-24   |
|      1 | Learn PHP     | John Poul    | 2007-05-24   |
|      3 | JAVA Tutorial | Sanjay       | 2007-05-06   |
+.....+.....+.....+.....+
3 rows in set (0.42 sec)

```

### **RESULT:**

Thus the program has been successfully completed & executed

**EX.NO: 16**

**NAME:**

**DATE:**

**REG NO:**

**WORKING WITH STRING AND DATE FUNCTIONS**

**AIM:**

To work with string and date functions using php

**(A) STRING LENGTH**

**PROGRAM:**

```
<!DOCTYPE html>
<html> <body>
<?php
echo strlen("Hello world!");
?>
</body> </html>
```

**Output:**

12

**(B) STRING COUNT**

```
<!DOCTYPE html>
<html> <body>
<?php
echo str_word_count("Hello world!")."<br>";
echo str_word_count("THIS IS MY COUNTRY");
?>
</body> </html>
```

**Output:**

2

4

### (C) STRING REVERSE

```
<!DOCTYPE html>
<html>
<body>

<?php
echo strrev("Hello world!");
?>

</body>
</html>
```

#### **Output:**

!dlrow olleH

### (D) STRING REPLACE

```
<!DOCTYPE html>
<html>
<body>

<?php
echo str_replace("world", "Dolly", "Hello world!");
?>

</body>
</html>
```

#### **Output:**

Hello Dolly!

### (E) DATE FUNCTION

```
<!DOCTYPE html>
<html> <body>

<?php
// Prints the day
echo date("l") . "<br>";

// Prints the day, date, month, year, time, AM or PM
echo date("l jS \of F Y h:i:s A") . "<br>";

// Prints October 3, 1975 was on a Friday
echo "Oct 3,1975 was on a ".date("l", mktime(0,0,0,10,3,1975)) . "<br>";

// Use a constant in the format parameter
echo date(DATE_RFC822) . "<br>";

// prints something like: 1975-10-03T00:00:00+00:00
echo date(DATE_ATOM,mktime(0,0,0,10,3,1975));
?>

</body> </html>
```

#### **Output:**

```
Wednesday
Wednesday 5th of October 2016 12:22:27 AM
Oct 3,1975 was on a Friday
Wed, 05 Oct 16 00:22:27 -0400
1975-10-03T00:00:00-04:00.
```

#### **RESULT:**

Thus the program has been successfully completed & executed.



**EX.NO: 17**

**NAME:**

**DATE:**

**REG NO:**

**DATABASE CONNECTIVITY IN PHP WITH MYSQL**

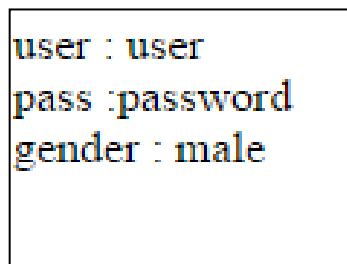
**AIM:**

To achieve database connectivity in php using MYSQL

**PROGRAM:**

```
<?php
$con=mysql_connect('localhost','root','');
mysql_select_db('test',$con);
$u="user";
$sql="select * from userpass where user='$u'";
$ret=mysql_query($sql,$con);
while($row=mysql_fetch_array($ret))
{
echo "user : ".$row['user']."<br>pass : ".$row['pass']."<br>gender : ".$row['gender'];
}
mysql_close();
?>
```

**OUTPUT:**

A screenshot of a web browser displaying the output of the PHP script. The output is displayed in a monospaced font with syntax highlighting. It shows three lines of text: 'user : user', 'pass : password', and 'gender : male', each on a new line.

```
user : user
pass : password
gender : male
```

**RESULT:**

Thus the program has been successfully completed & executed.