1. Write a PHP script to print prime numbers between 1-50.

```
<?php
$number=2;
while($number<50)
 {
    $div_count=0;
    for($i=1;$i<=$number;$i++)
     {
        if($number%$i==0)
         {
           $div_count++;
         }
     }
     if($div_count<3)
      {
        echo $number.",";
      }
      $number++;
 }

?>
```

OUTPUT:

2 , 3 , 5 , 7 , 11 , 13 , 17 , 19 , 23 , 29 , 31 , 37 , 41 , 43 , 47 ,

2. PHP script to a. Find the length of a string. b. Count no of words in a string. c. Reverse a string. d. Search for a specific string.

a. Find the length of a string

```
<?php
echo strlen("Hello");
?>
```

OUTPUT:

5

b. Count no of words in a string

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

OUTPUT:

2

c.Reverse a string

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

OUTPUT:

!dlroW olleH

d.Search for a specific string

```
<?php
echo strpos("Hello world!", "world");
?>
```

OUTPUT:

6

3. Write a PHP script to merge two arrays and sort them as numbers, in descending order.

```
<?php
   $a1=array(1,3,15,7,5);
   $a2=array(4,3,20,1,6);
   $num=array_merge($a1,$a2);
   array_multisort($num,SORT_DESC,SORT_NUMERIC);
   print_r($num);
?>
```

OUTPUT:

Array ([0] => 20 [1] => 15 [2] => 7 [3] => 6 [4] => 5 [5] => 4 [6] => 3 [7] => 3 [8] => 1 [9] => 1)

4. Write a PHP script that reads data from one file and write into another file.

```
<?php
$filename1="C:\\xampp\\htdocs\\jahnavi\\h1.txt";
$fp1=fopen($filename1,"r");
$contents=fread($fp1,filesize($filename1));
echo "<pre>$contents</pre>";
$filename2="C:\\xampp\\htdocs\\jahnavi\\h2.txt";
$fp2=fopen($filename2,"w");
```

```
fwrite($fp2,$contents);
fclose($fp1);
fclose($fp2);
?>
```

OUTPUT:

H1.txt

```
My name is Jahnavi. I like books.
```

H2.txt

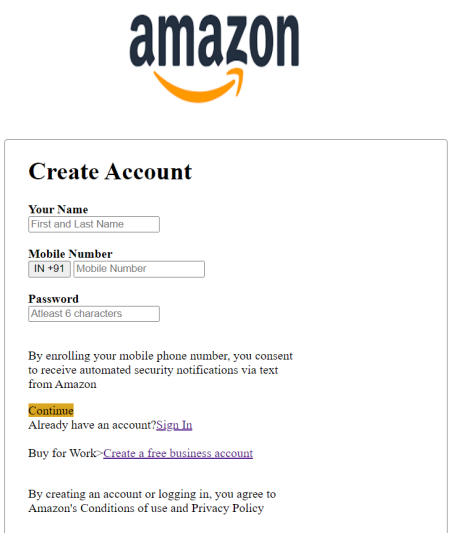
```
My name is Jahnavi. I like books.
```

5. Amazon Registration Page using HTML

```
<!DOCTYPE html>
<head>
    <title>Amazon Registration Page</title>
</head>
<body>
    <center><img src="amazonlogo.png" alt="amazon"
width="275px" height="200px"></center>
    <form
style="margin-left:480px;margin-right:450px;margin-top:0px;bor
der:solid
#808080;border-width:0.7px;border-radius:4px;padding-left:30px
;padding-bottom:30px;">
        <h1><b>Create Account</b></h1>
        <b>Your Name</b><br>
        <input type=""text placeholder="First and Last Name"
style width="330px"><br><br>
        <b>Mobile Number</b><br>
        <button type="submit"> IN +91</button>
        <input type=""text placeholder="Mobile Number" style
width="250px"><br><br>
        <b> Password</b><br>
        <input type=""text placeholder="Atleast 6 characters"
style width="250px"><br><br>
```

```
        <p> By enrolling your mobile phone number, you
consent<br> to receive automated security notifications via
text<br> from Amazon</p>
        <submit type="continue"
style="width:250px;background-color:
goldenrod;">Continue</submit><br>
        Already have an account?<a href="">Sign In</a><br><br>
        Buy for Work><a href="">Create a free business
account</a><br><br>
       <p> By creating an account or logging in, you agree
to<br> Amazon's Conditions of use and Privacy Policy</p></p>
       </form>
       </body>
       </html>
```

OUTPUT:



6. Amazon Sign In page using HTML

```
<!DOCTYPE html>
<head>
    <title>Amazon Sign in page</title>
    <style>
       #link{
            text-decoration:none;
        }
        </style>
        </head>
        <body>
```

```
            <center><img src="amazonlogo.png" alt="amazon"
width="275px" height="200px"></center>
            <form
style="margin-left:480px;margin-right:450px;margin-top:0px;border:so
lid
#808080;border-width:0.7px;border-radius:4px;padding-left:150px;padd
ing-bottom:30px;">
                <h1>Sign In</h1>
                <p>Email or Mobile Phone Number</p>
                <input type="text" style width="250px"><br><br>
                <button type="continue"
style="width:200px;background-color:goldenrod;">Continue</button>
                <pre><p> By continuing, you agree to Amazon's<br><a
href=" ">Conditions of Use</a> and <a href=" ">Privacy
Notice</a></p>
                    <ul type="triangle"><li><a href=" ">Need
help?</a></li></ul>
                </pre>
            </form>
            <form style="margin-left:500px;">
                ----------------------------------New to
Amazon----------------------------------------<br>
                <button type="submit"
style="width:200px;background-color:#808080;">
                    <a id="link" href="C:\Users\Jahnavi
Bandaru\Desktop\Amreg.html"> Create Amazon account</a></button>
                </form>
            </body>
            </html>
```

OUTPUT:

6. Create a servlet to print hi

MyServlet.java

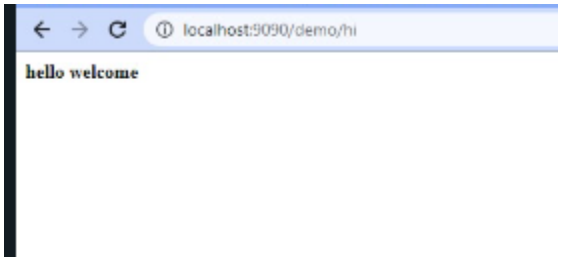
```
 import java.io.*;
import javax.servlet.*;
public class MyServlet extends GenericServlet {
public void service(ServletRequest req , ServletResponse res)
throws IOException,ServletException {
res.setContentType("text/html");
PrintWriter pw = res.getWriter();
pw.print("<html><body>");
pw.print("<b> hello welcome </b>");
pw.print("</body> </html>");
pw.close();
}
}
```

Web.xml

web.xml

```
<web-app>
<servlet>
<servlet-name>First</servlet-name>
<servlet-class>MyServlet</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>First</servlet-name>
<url-pattern>/hi</url-pattern>
</servlet-mapping>
</web-app>
```

OUTPUT:



7. Create servlets for reading initialization parameters

Login.java

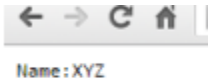
```
 import java.io.*;
 import java.util.Enumeration;
 import javax.servlet.*;
 import javax.servlet.http.*;

 public class Login extends HttpServlet{
 public void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException,IOException
 {

 PrintWriter pw=res.getWriter();
 ServletConfig config=getServletConfig();
 Enumeration e=config.getInitParameterNames();
 while(e.hasMoreElements())
 {
 String name=(String)e.nextElement();
 String value=config.getInitParameter(name);
 pw.print(name+":"+value);
 }
 pw.close();
 }
 }
```

Web.xml

```
<web-app>
 <servlet>
 <servlet-name>ABC</servlet-name>
 <servlet-class>Login</servlet-class>
 <init-param>
 <param-name>Name</param-name>
 <param-value>XYZ</param-value>
 </init-param>
```

```
</servlet>
<servlet-mapping>
<servlet-name>ABC</servlet-name>
<url-pattern>/log</url-pattern>
</servlet-mapping>
</web-app>
```

OUTPUT:



8. Create Servlet to read Servlet Parameters

PostParametersServlet.java

```
import java.io.*;
 import java.util.*;
 import javax.servlet.*;
 public class PostParametersServlet extends GenericServlet
 {
 public void service(ServletRequest request,ServletResponse
response)throws
ServletException, IOException
 {// Get print writer.
 PrintWriter pw = response.getWriter();
 // Get enumeration of parameter names.
 Enumeration e = request.getParameterNames();
 // Display parameter names and values.
 while(e.hasMoreElements())
 {
 String pname = (String)e.nextElement();
 pw.print(pname + " = ");
 String pvalue = request.getParameter(pname);
 pw.println(pvalue);
 }
 pw.close();
 }
 }
```

PostParameters.html

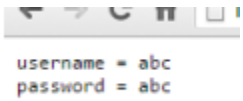
```
<html>
 <body>
```

```
<center>
<form
name="Form1"method="post"action="http://localhost:8088/damp/hi">
<B>Name:</B>
<input type=textbox name="username" ><br><br>
<B>Password</B>
<input type=textbox name="password" ><br><br>
<input type=submit value="Submit">
</body>
</html>
```

Web.xml

```
<web-app>
 <servlet>
 <servlet-name>ABC</servlet-name>
 <servlet-class>PostParametersServlet</servlet-class>
 </servlet>
 <servlet-mapping>
 <servlet-name>ABC</servlet-name>
 <url-pattern>/hi</url-pattern>
 </servlet-mapping>
 </web-app>
```

OUTPUT:



9. Registration using servlet

Registration.html

```
<body>

<form action="http://localhost:9090/demo/Regist" method="post">


Name:<input type="text" name="userName"/><br/><br/>

Password:<input type="password" name="userPass"/><br/><br/>

Email Id:<input type="text" name="userEmail"/><br/><br/>

Country:
```

```
<select name="userCountry">
<option>India</option>
<option>Pakistan</option>
<option>other</option>
</select>
 <br/><br/>
<input type="submit" value="register"/>
 </form>
</body>
</html>
```

Registration.java

```
import java.io.*;
import java.sql.*;
import javax.servlet.ServletException;
import javax.servlet.http.*;
  public class Register extends HttpServlet {
public void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
  response.setContentType("text/html");
PrintWriter out = response.getWriter();
 String n=request.getParameter("userName");
String p=request.getParameter("userPass");
String e=request.getParameter("userEmail");
String c=request.getParameter("userCountry");
```

```
try{
Class.forName("com.mysql.cj.jdbc.Driver");
Connection con=DriverManager.getConnection(
"jdbc:mysql://localhost:3306/shahenaaz","root","");
PreparedStatement ps=con.prepareStatement( "insert into registeruser values(?,?,?,?)");
ps.setString(1,n);
ps.setString(2,p);
ps.setString(3,e);
ps.setString(4,c);
int i=ps.executeUpdate();
if(i>0)
out.print("You are successfully registered...");
}catch (Exception e2) {System.out.println(e2);
}
out.close();
}
 }
```

Web.xml

```
<web-app>
<servlet>
<servlet-name>Register</servlet-name>
<servlet-class>Register</servlet-class>
</servlet>

<servlet-mapping>
<servlet-name>Register</servlet-name>
```

```
<url-pattern>/Regist</url-pattern>
</servlet-mapping>

</web-app>
```

CD PROGRAMS:

1. **Write a C program to design a lexical analyzer that recognizes identifiers and keywords of flow control statements of C language**

```
#include<stdio.h>

#include<ctype.h>

#include<conio.h>

#include<string.h>

void main()

{

int i,flag;

char str[50];

clrscr();

printf("enter string\n");

scanf("%s",str);

if((strcmp(str,"if")==0)||(strcmp(str,"else")==0)||(strcmp(str,"do")==0)||(strcmp(str,"for")==0)||(strc
mp(str,"break”)==0)||(strcmp(str,"while")==0)||(strcmp(str,"switch")==0)||
(strcmp(str,"case")==0)||(strcmp(str,"default")==0))

printf("keyword of control flow statements");

else

if(isalpha(str[0])&&strlen(str)<32)

{
```

```
for(i=1;i<strlen(str);i++)
if(isalnum(str[i])||str[i]=='_')
flag=0;
else
flag=1;
}
if(flag==0)
printf("identifier");
else
printf("not a keyword or identifier");
getch();
}
```

Output 1:

enter string

abcd

identifier

Output 2:

enter string

for

keyword of control flow statements

Output 3:

enter string

2dfdgfh

not a keyword or identifier

2. Write a C program to construct Recursive Descent parser for the following grammar

E->TR

R->+TR/e

T->FP

P->*FP/e

F->a/(E)

```
#include<stdio.h>
#include<conio.h>
void E();
void R();
void P();
void T();
void F();
void error();
char str[20];
int ip=0;
void main()
{
clrscr();
printf("enter string  in a's and ending with $\n");
scanf("%s",str);
E();
```

```
if(str[ip]=='$')
printf("parsing is success");
}
void E()
{
T();
R();
}
void R()
{
if(str[ip]=='+')
{
ip++;
T();
R();
}
else {}
}
void T()
{
F();
P();
}
void P()
```

```
{
if(str[ip]=='*')
{
ip++;
F();
P();
}
else {}
}
void F()
{
if(str[ip]=='(')
{
ip++;
E();
if(str[ip]==')')
ip++;
else
{
printf("I value is required\n");
exit();
}
}
else if(str[ip]=='a')
```

```
ip++;
else error();
}
void error()
{
printf("error occured during parsing");
exit();
}
```

Output 1:

```
enter string  in a's and ending with $
a+a$
parsing is success
```

Output 2:

```
enter string  in a's and ending with $
a+*a$
error occured during parsing
```

3. Write a C program to construct predictive parser for the following grammar

E->TR

R->+TR/e

T->FP

P->FP/e

F->a/(E)

```
#include<stdio.h>
```

```
#include<process.h>
#include<conio.h>
char stack[20];
int top=-1;
void push(char);
char pop();
void error();
void main()
{
char str[10],ch;
int ip=0;
clrscr();
printf("enter string ending with $");
scanf("%s",str);
push('$');
push('E');
while((ch=pop())!='$')
{
switch(ch)
{
case 'E':{
        if(str[ip]=='a'||str[ip]=='(')
        {
        push('R');
```

```
        push('T');
        }
        else error();
        }
        break;
case 'R':{
        if(str[ip]=='+')
        {
        push('R');
        push('T');
        push('+');
        }
        else if(str[ip]==')'||str[ip]=='$')
        {
        }
        else error();
        }
        break;
case 'T':{
        if(str[ip]=='a'||str[ip]=='(')
        {
        push('P');
        push('F');
        }
```

```
        else error();
        }
        break;
case 'P':{
        if(str[ip]=='*')
        {
        push('P');
        push('F');
        push('*');
        }
        else if(str[ip]=='+'||str[ip]==')'||str[ip]=='$')
        {
        }
        else error();
        }
        break;
case 'F':{
        if(str[ip]=='a')
        push('a');
        else if(str[ip]=='(')
        {
        push(')');
        push('E');
        push('(');
```

```
            }
            else error();
            }
            break;
case '+':if(str[ip]=='+')
            ip++;
            break;
case '*':if(str[ip]=='*')
            ip++;
            break;
case '(':if(str[ip]=='(')
            ip++;
            break;
case ')':if(str[ip]==')')
            ip++;
            break;
case 'a':if(str[ip]=='a')
            ip++;
            break;
}
}
if(stack[top+1]=='$')
printf("parsing is succesful");
}
```

```
void error()
{
printf("parsing is not successful");
exit(0);
}
void push(char c)
{
top++;
stack[top]=c;
}
char pop()
{
top--;
return(stack[top+1]);
}
```

Output 1:

enter string in a's and ending with $

a+a$

parsing is successful

Output 2:

enter string in a's and ending with $

a+*a$

parsing is not successful

4. Write a Lex specification to recognize +ve integers,reals and -ve integers,reals.

```
%{
#include<stdio.h>
%}
%%
        "+"?[0-9]+                  {printf("%s:positive integers",yytext);}
        -[0-9]+                     {printf("%s:negative integers",yytext);}
        -[0-9]+\.[0-9]+                     {printf("%s:negative real numbers",yytext);}
        "+"?[0-9]+\.[0-9]+      {printf("%s:positive real numbers",yytext);}
%%
main()
{
 yylex();
}
```

Compilation: lex noformat.l

```
cc lex.yy.c -ll
./a.out
24
positive integer
+24.12
positive real number
-24
negative integer
```

-24.12

negative real number

5. **Write a Lex specification for converting real numbers to integers.**

```
%{
        int i,j;
        #include<stdio.h>


%}
%%
[0-9]*\.[0-9]+  {
                for(i=0;i<10;i++)
                        {
                                if(yytext[i]=='.')
                                for(j=0;j<=i-1;j++)
                                {
                            printf("%c",yytext[j]);
                          }
                        }
                        exit(0);
                        }
%%
main()
{
```

```
 yylex();
}
```

Compilation: lex realtoint.l

cc lex.yy.c -ll

./a.out

24.12

12

6. Write a Lex specification to print the number of days in a month using a procedure

```
%{
        #include<stdio.h>
        Int year;
%}
%%
jan|mar|may|july|aug|oct|dec  {printf("31 days");}
april|june|sep|nov       {printf("30 days");}
feb      {leap();}
[a-zA-Z]*         {printf("invalid");}
%%
main()
{
 yylex();
}
```

```
leap()
{
        printf(“enter year”);
        scanf(“%d”,&year);
        if(year%4==0)
        printf("29 days");
        else printf("28 days");
}
```

Compilation: lex daysinamonth.l

```
        cc lex.yy.c -ll
        ./a.out
        jan
        31 days
        feb
        enter year
        1984
        29 days
```

7. Write a Lex specification to print a number in between 0-100 in words

```
%{
        #include<stdio.h>
%}
```

```
%%
        2/[0-9] printf("Twenty");
        3/[0-9] printf("Thirty");
        4/[0-9] printf("Forty");
        5/[0-9] printf("Fifty");
        6/[0-9] printf("Sixty");
        7/[0-9] printf("Seventy");
        8/[0-9] printf("Eighty");
        9/[0-9] printf("Ninty");
        10 printf("Ten");
        11 printf("Eleven");
        12 printf("Twelve");
        13 printf("Thirteen");
        15 printf("Fifteen");
        1[4|6|7|8|9] {
                        units();
                        printf("teen");
                       }
        [1-9] units();
        0       ;
        %%
        units()
        {
         switch(yytext[yyleng-1])
```

```
 {
  case'1':printf("One");
break;
  case'2':printf("Two");
          break;
  case'3':printf("Three");
          break;
  case'4':printf("Four");
            break;
  case'5':printf("Five");
            break;
  case'6':printf("Six");
          break;
  case'7':printf("Seven");
          break;
  case'8':printf("Eight");
          break;
  case'9':printf("Nine");
            break;
}
}
```

Compilation: lex twodigitno.l

cc lex.yy.c -ll

```
./a.out
6
Six
12
Twelve
```

8. Write a Lex specification to retrieve comments.

```
%{
    #include<stdio.h>
%}
%%
[/][/][a-z A-Z 0-9]*    {printf("%s",yytext);}
[a-z A-Z 0-9]*          {printf(" ");}
[/][*][a-z A-Z 0-9]*[*][/] {printf("%s",yytext);}
%%
main()
{
 yylex();
}
```

Compilation: lex comments.l

```
cc lex.yy.c -ll
./a.out
Hello //world
world
```

9. Write a Lex specification to print a number in between 0-1000 in words.

```
%{
        #include<stdio.h>
%}
%%
        1/[0-9][0-9] print(“One Hundred”);
        2/[0-9][0-9] print(“Two Hundred”);
        3/[0-9][0-9] print(“Three Hundred”);
        4/[0-9][0-9] print(“Four Hundred”);
        5/[0-9][0-9] print(“Five Hundred”);
        6/[0-9][0-9] print(“Six Hundred”);
        7/[0-9][0-9] print(“Seven Hundred”);
        8/[0-9][0-9] print(“Eight Hundred”);
        9/[0-9][0-9] print(“Nine Hundred”);
        2/[0-9] printf("Twenty");
        3/[0-9] printf("Thirty");
        4/[0-9] printf("Forty");
        5/[0-9] printf("Fifty");
        6/[0-9] printf("Sixty");
        7/[0-9] printf("Seventy");
        8/[0-9] printf("Eighty");
        9/[0-9] printf("Ninty");
        10 printf("Ten");
        11 printf("Eleven");
```

```
12 printf("Twelve");

13 printf("Thirteen");

15 printf("Fifteen");

1[4|6|7|8|9] {

                    units();

                  printf("teen");

                  }

[1-9] units();

0         ;

%%

units()

{

 switch(yytext[yyleng-1])

 {

  case'1':printf("One");

break;

  case'2':printf("Two");

          break;

  case'3':printf("Three");

          break;

  case'4':printf("Four");

            break;

  case'5':printf("Five");

            break;
```

```
    case'6':printf("Six");
            break;
    case'7':printf("Seven");
            break;
    case'8':printf("Eight");
            break;
    case'9':printf("Nine");
              break;
  }
  }
```

Compilation: lex threedigitno.l

```
cc lex.yy.c -ll
./a.out
6
Six
12
Twelve
126
One Hundred Twenty Six
```

10. Write a Lex specification to design a lexical analyzer that recognizes identifiers and keywords of flow control statements of C language

```
%{
  #include<stdio.h>
%}
```

```
%%
If|else|while|do|switch|case|break|for|default {printf("Keyword");}
IF|ELSE|WHILE|DO|SWITCH|CASE|BREAK|FOR|DEFAULT          {printf("Keyword");}
[A-Z a-z]+[a-z A-Z 0-9 _]*       {printf("identifier");}
%%
main()
{
 yylex();
}
```

Compilation: lex lexanalysis.l

```
cc lex.yy.c -ll
./a.out
If
Keyword
FOR
Keyword
Abc123_def
identifier
```