```
<!DOCTYPE html>
<html>
<head>
    <title>Assignment 4 Recursive Descent Parser</title>
</head>
<body>
<body bgcolor = "lightgreen">

<h1 style="text-align:center;"><font color = "black">Assignment PL4 Recursive
Descent Recognizer</font></h1>
<p><b> My Recursive Descent Recognizer</b><br>

<p><b>BNF Grammar:

<p><b>EXP ::= EXP + TERM | EXP - TERM | TERM<p>
<p>TERM ::= TERM * FACTOR | TERM / FACTOR | FACTOR<p>
<p>FACTOR ::= ( EXP ) | DIGIT<p>
<p>DIGIT ::= 0 | 1 | 2 | 3</p>


<p style = "color:red;">  Example of Valid String: 1+2$ , 0-3*2$, 2/(3+1)$</p>
<p style = "color: red;"> Example of Invalid String: 1+2, 1*4$, 1*a$</p>

<p><center><form action="rdr.php" method="post">

<p>Enter string using the numbers 0 to 3, and the
symbols +, -, *, /, (, and ).</p>
<p>Your end of string variable will be dollar sign ($). The default message is
"Input String is Invalid" since there is no input.</p>
<p>Once the user enters a string and clicks on 'Submit', the RDR will check whether
entered string is valid or not for a given grammar.</p>

Input String: <input type="text" name="String" />
    <input type="submit" value="Submit" />
    </form></center><p>




<?php
$input = $_POST["String"];
$parser = new RDR($input);

abstract class Grammar {
         protected $inputString;
         protected $pointerInString;
         protected $resultString;
         protected $endOfString;
abstract protected function exp();
function __construct($input, $delimiter = '$')
{
         $this->inputString = $input;
         $this->pointerInString = 0;
         $this->resultString = true;
         $this->endOfString = $delimiter;
         $this->exp();
         if(!$this->endOfInput())
         $this->resultString = false;
}

function isresultString()
{
         return $this->resultString;
}

protected function endOfInput()
{
      $isDone = ($this->pointerInString >= strlen($this->inputString)) ||
(strlen($this->inputString) == 0);
      if($this->pointerInString == (strlen($this->inputString) - 1))
      if($this->inputString[$this->pointerInString] == $this->endOfString)

                           $isDone = true;
                           return $isDone;

}

protected function match($myToken)
{
       if(($this->pointerInString < strlen($this->inputString)) &&
($this->inputString[$this->pointerInString] == $myToken))
{
       $this->pointerInString += 1;
       return true;
}
       else
       return false;
}
}
/* Grammar for RDR is:
* EXP ::= EXP + TERM | EXP - TERM | TERM
* TERM ::= TERM * FACTOR | TERM / FACTOR | FACTOR
* FACTOR ::= (EXP) | DIGIT
* DIGIT ::= 0 | 1 | 2 | 3
* Assume the input ends with '$'.
```

```
*/

class RDR extends Grammar {
   function exp()
  {
      if($this->resultString)
      $this->term();
      $done = $this->endOfInput();
          while($this->resultString && !$done)
               {
          if(($this->inputString[$this->pointerInString] == '+') ||
($this->inputString[$this->pointerInString] == '-'))
                 {
                    $this->match($this->inputString[$this->pointerInString]);
                    $this->term();
                 }
          elseif(($this->inputString[$this->pointerInString] == ')') ||
($this->inputString[$this->pointerInString] == $this->endOfString))
                   {
                   $done = true;
                   }
          else
                    $this->resultString = false;
       }
     }


   function term()
  {
      if($this->resultString)
      $this->factor();
      $done = $this->endOfInput();
          while($this->resultString && !$done)
               {
         if(($this->inputString[$this->pointerInString] == '*') ||
($this->inputString[$this->pointerInString] == '/'))
              {
                   $this->match($this->inputString[$this->pointerInString]);
                   $this->factor();
              }
         elseif(($this->inputString[$this->pointerInString] == '+') ||
($this->inputString[$this->pointerInString] == '-') ||
($this->inputString[$this->pointerInString] == ')') ||
($this->inputString[$this->pointerInString] == $this->endOfString))
               {
                $done = true;
               }
         else
```

```
                $this->resultString = false;
       }
    }


       function factor()
    {
         if($this->endOfInput())
         $this->resultString = false;
         if($this->resultString)
           {
              if($this->inputString[$this->pointerInString] == '(')
                   {
                        $this->match($this->inputString[$this->pointerInString]);
                        $this->exp();
                        $this->resultString = $this->resultString &&
$this->match(')');
                   }
              else
                {
                    $this->digit();
                }
            }
    }


       function digit()
           {
             $digitArray = array('0', '1', '2', '3');
             if($this->endOfInput())
             $this->resultString = false;
                  elseif(array_search($this->inputString[$this->pointerInString],
$digitArray) !== False)
                        {
                              $this->resultString = $this->resultString &&
$this->match($this->inputString[$this->pointerInString]);
                        }
                  else
                               $this->resultString = false;

           }
}
?>


<?php
         if($parser->isresultString())
               {
```

```
?>
         <div><p><font color="red">Input String <?php echo $input; ?> is 
<strong>Valid</strong>.</font></p></div>
<?php
}
         else
{
?>
         <div><p><font color="red">Input String <?php echo $input; ?> is 
<strong>Invalid</strong>.</font></p></div>
<?php
}
?>



<h2 style="text-align:center;"><font color = "black">RDR report</font></h1>
<p> (1) I chose PHP as the language for my Web-based Recursive Descent Recognizer. I 
never learn or done PHP before, so it took a long time for
me to implement my code. The difficult part was to learn how to pass input string to 
php function using html.
<p> (2) The useful resource that I used for this assignment is <p>W3schools: <a 
href= "https://www.w3schools.com/php7/default.asp">W3 PHP</a><p>
I also used the resources provided under CSC 135's module section on Canvas to help 
me start with the assignment.
<p> (3) I would encourage students to start the assignment early so that they can 
have more time to do the testing. For me, the hardest part is testing because if 
there are some errors, I have to go back and trace the errors.
In this assignment, I wrote my PHP code on Notepad, so it was a little bit difficult 
and time-consuming to go back and fix my code.
</body>
</html>
```