```php
<?php
/**
* Grammar class
*
* The Grammar class is a main/top level class for parsing a string
* and matching it to a grammar.
* grammar rules will be implemented in to another class which will extend this basic class
*
*
*/
abstract class Grammar {

// User input - string.
protected $inputString;

//Pointer pointing to current position in input string
protected $pointerInString;

// boolean variable which will return true or false based on parsing result
protected $resultString;

//end of string variable '$ - in this case'.
protected $endOfString;
/**
* Recursive Descent Parser
*
* This function will get overridden by child classes
*/
abstract protected function exp();
```

```
function __construct($input, $delimiter = '$') {

$this->inputString = $input; // user input string taken from input page

$this->pointerInString = 0; // initial pointer value will be 0 - pointer pointing to first character in input
string

$this->resultString = true; // it will be set to false if program can not match string to the expected at any
point in time while execution

$this->endOfString = $delimiter;

$this->exp();  // starting point for each parsing

if(!$this->endOfInput())

$this->resultString = false; // this means the string contains some unparsable character

}

/*

*

* True if expression is resultString else False

*/

function isresultString() {

return $this->resultString;

}

/*


*/

protected function endOfInput() {

// check for end of the string

$isDone = ($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 $isDone;
}
/*
* match function basically matches character with current pointer character
* if matches, it will advance pointer to next character and return true.
*/
protected function match($myToken) {
if(($this->pointerInString < strlen($this->inputString)) &&
($this->inputString[$this->pointerInString] == $myToken))
{
$this->pointerInString += 1;
return true;
}
else
return false;
}
}
```

```
/*
*
* Grammar for RDR2:
* EXP ::= ( LIST ) | a
* LIST ::= LIST , EXP | EXP
*
* Assume the input ends with '$'.
*/

class RDR2 extends Grammar {
function exp() {
if($this->endOfInput())
$this->resultString = false;
if($this->resultString)
{
if($this->inputString[$this->pointerInString] == 'a')
{
$this->match($this->inputString[$this->pointerInString]);
}
elseif($this->inputString[$this->pointerInString] == '(')
{
$this->match($this->inputString[$this->pointerInString]);
$this->ecList();
if($this->endOfInput())
$this->resultString = false;
if($this->resultString)
{
$this->match(')');
}
}
else
$this->resultString = false;
}
}
function ecList() {
$this->exp();
$done = false;
while(!$done && $this->resultString && !$this->endOfInput())
{
if($this->inputString[$this->pointerInString] == ',')
{
$this->match($this->inputString[$this->pointerInString]);
$this->exp();
}
elseif($this->endOfInput() || $this->inputString[$this->pointerInString] == ')')
{
$done = true;
}
else
$this->resultString = false;
}
}
}
```