Exercises

1. **Days in Month:** Write a function `daysInMonth` that takes a month (between 1 and 12) as a parameter and returns the number of days in that month in a non-leap year. For example a call to `daysInMonth(6)` should return 30, because June has 30 days.

Month	Jan	Feb	Mar	Apr	May	Jun	Jul	Aug	Sep	Oct	Nov	Dec
Days	31	28	31	30	31	30	31	31	30	31	30	31

2. **Vowel Count:** Write a function called `vowelCount` which accepts a string as a parameter, and returns the number of vowels in the in string. Vowels include a, e, i, o, and u. You may assume the string will be all lowercase.

3. **1337 5p34k:** Write a function leetspeak() which accepts a string as a parameter and returns a "leetspeak" version of the string. "Leetspeak" is an internet slang in which various English letters are replaced with other ASCII characters similar in appearance. The following table shows some common leetspeak translations.

English	A	E	G	L	O	S	T
Leetspeak	4	3	9	1	0	5	7

   You should replace at least English five characters with their leetspeak equivalent. For more information about leetspeak, visit http://en.wikipedia.org/wiki/Leet.

4. **Palindromes:** Write a function `isPalindrome` that accepts a string as a parameter and returns true if the string is a palindrome and false otherwise. A string is considered a palindrome if it has the same sequence of letters when reversed (for example, `"radar"`, `"toot"`, `"mom"`, `"a"`, `""`). Your function should be case-insensitive; for example, "Mom" and "RAdar" should be considered palindromes.

5. **Drunken Capitalizer:** Write a PHP function called `drunkenCapitalizer` that accepts a String as a parameter and returns the same String with randomly capitalized characters. For example, `drunkenCapitalizer("php rules")` could return `"PHP RULES"`, `"pHp rULes"`, `"Php RULEs"` or `"PHP rules"`.

   Switching cases is not needed; you are only required to randomly capitalize letters. So calling `drunkenCapitalizer("PHP RULES")` would only return `"PHP RULES"`.

6. **Fix Spacing:** Write a function called `fixSpacing` which accepts a filename as a parameter. The function should reduce all multiple spaces or tabs to a single space, replacing the old contents of the file. Fox example, an input file could contain the following text:

2 Exercises

```
four          score   and
seven      years ago        our

fathers brought           forth
        on this        continent
a   new

        nation
```

After calling `fixSpacing`, the contents of the file should be as follows:

```
four score and

seven years ago our

fathers brought forth
on this continent
a new

nation
```

Each word is to appear on the same line in new file as it appears in the original file. Notice that lines can be blank.

7. **Search Terms:** Write a function searchTerms() that takes a search string as a parameter and returns an array of all search terms in the string. Every token in the string is considered a search term, except for tokens wrapped in double quotation marks: these phrases should be treated as a single search term. You may assume that all tokens in the string are separated by exactly one space and that no search string has an odd number of double quotation marks – that is, all beginning quotation marks have a matching set of ending quotation marks.

   For example, `searchTerms("curious george")` returns {"curious", "george"} and `searchTerms('"The Man with the Yellow Hat"')` returns {`"The Man with the Yellow Hat"`}.

8. **Line sums:** Write a PHP function called `lineSum` that accepts one integer and a filename as a parameter and returns the sum of the integers on that line number. For example, you may have a file named `sums.txt` that contains the following data:

```
5
15 10
20 25
50
200
50 60
75 100
```

   A call to `lineSum( "sums.txt", 2)` returns 25, and a call to `lineSum("sums.txt", 5)` returns 200.

9. **GCD and Euclid's Algorithm:** Write a function named `gcd` that accepts two integers as parameters and returns the greatest common divisor (GCD) of the two numbers. The greatest common divisor of two integers a and b is the largest integer that is a factor of both a and

b. The GCD of any number and 1 is 1, and the GCD of any number and 0 is that number.

One efficient way to compute the GCD is to use Euclid's algorithm, which states the following:
gcd(a, b) = gcd(b, a % b)
gcd(a, 0) = Absolute value of a

For example:
`gcd(24, 84)` returns 12,
`gcd(105, 45)` returns 15, and
`gcd(0, 8)` returns 8.

10. **Inside Out:** Write a PHP function called insideOut() that will accept an array and return it with its innermost elements swapped with its outer most elements. For example, calling insideOut(array(1, 2, 2, 1)) would return the array: (2, 1, 1, 2).

    Similarly, the method call insideOut(array("Everyone", "says", "Kelly", "is", "REALLY", "awesome")) would result in: ("Kelly", "says", "Everyone", "awesome", "REALLY", "is").

    The array should only be altered if it has an even number of elements. Return the array passed in if this is not the case.

11. **Emailidate:** Create a function called emailidate() that validates email addresses. The function takes as a parameter a string of email addresses delimited by commas. It should return an array of valid email addresses. An email address is considered valid if it is of the form a@b.c, where a, b, and c are non-empty strings. Invalid email addresses are ignored. If none of the email addresses are valid, the function should return FALSE.

12. **University Info:** Write a function universityInfo() which accepts a university name as a parameter and returns an array containing information about the given university. The array should have a "location" key, whose value is the university's location, a "founded" key, whose value is the founding date of the university, and a "name" key containing the name of the university. The information about the university is stored in a text file called unidata.txt and has the following format:

    University Name
    University Location
    Date Founded
    University Name
    University Location
    Date Founded
    …

    You may assume any university name passed to universityInfo() can be found in the text file and that all universities in the text file have all required information. This function should be case insensitive, i.e. universityInfo("UnIveRsItY OF washington") should return the same results as universityInfo("University of Washington").

    Example call:
    $info = universityInfo("University of Washington");

```
echo $info[“name”]; //outputs "University of Washington"
echo $info[“location”]; //outputs "Seattle, Washinton"
echo $info[“founded”]; //outputs "1861"
```

13. **Folder Crawler:** Write a PHP function called folderCrawler() that takes a full path to a folder as a parameter and outputs the HTML for an unordered list of all the files in that folder. The filenames should link to the files themselves, and before every filename you should display an appropriate icon based on the file extension. All icon images are in a folder called icons located in the root directory of the website and each icon is saved as a gif file named “<extension>.gif”; for example, the icon for an HTML file is called “html.gif.” You may assume that every file extension has an icon.

14. **Welcome Message:** The following webpage has a form prompting the user for his or her name. Edit the HTML and use embedded PHP to modify the following webpage to display a welcome message of the form, “Hello, <name>!” if the user types his or her name into the text box and presses “Enter.” The form should send the submission as a POST request and the welcome message should be displayed in as a level one heading above the form.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
   <title>Welcome to my page!</title>
</head>
<body>
   <p>
      Enter your name: <br />
      <input type="text" />
      <input type="submit" value="Enter" />
   </p>
</body>
</html>
```

15. **Welcome Message, revisited:** Modify your solution to the **Welcome Message** exercise to remember the last person who visited the webpage. Save the person’s submitted name in a flat file called “visitor.txt and read the first line of this text file to retrieve the name when the page loads. You should overwrite this name every time someone submits a name to the form. You can assume there is always a name in the text file and you can assume the submitted name is not blank.

    You should add the following paragraph after the prompt for the user’s name. Replace “[NAME]” with the name of the last visitor.

```
<p>
   The last person who visited my website was [NAME].
</p>
```

16. **Intro to Phishing:** The following HTML is a snippet of a website trying to masquerade as Google. It asks for the user’s name and credit card number in addition to the search query.

Write a PHP program called phish.php that processes the form submission. This program should save the name and credit card number to a text file called suckers.txt. (The information can be saved in any readable format of your choosing.) The program should then redirect the user to the actual Google search results page for the search query they typed in. You may find the PHP header() function to be useful in this task.

Hint: As described in the beginning of Chapter 6, the URL for a Google search results page is of the form "`http://www.google.com/search?q=<search value>".`

```
<h1>Google.com!</h1>
<p>
   Welcome to Google.com <br />
   Google now requires your full name and credit card number before
   you are allowed to make a search.
</p>
<form action="phish.php" method="post">
   <fieldset>
      Name: <input type="text" name="name" /><br />
      Credit Card Number: <input type="text" name="ccn" /><br />
      Search Query: <input type="text" name="query" /> <br />
      <input type="submit" value="I'm pheeling lucky" />
   </fieldset>
</form>
```

Disclaimer: We do not in any way condone or encourage phishing attacks! This is just a silly exercise written in jest.

17. **Image Gallery Search:** The following HTML snippet is the skeleton of a simple search page for an image gallery. The gallery stores all of its images in a directory called images. Write a PHP program called search.php to implement the searching feature. An image is considered a "match" to the search string if the name of the image contains the entirety of the search string. For example, a query of "tea" might match "tea.jpg" or "steamboat.jpg."

    The search should be case-insensitive and you should eliminate the whitespace surrounding a query before processing it. You should output an unordered list of links to all matches. A blank query should return no results. You may assume there are only image files in the images directory.

```
<h1>Image Gallery Search<h1>
<form action="search.php" method="get">
   <fieldset>
      Type a query: <input type="text" name="query" /> <br />
      <input type="submit" value="Search" />
   </fieldset>
</form>
```

18. **Image Gallery Search, revisited:** Modify your solution to the **Image Gallery Search** exercise to account for a smarter search algorithm and a more appealing results page:

- Instead of matching only image names that contain the *exact* search string, an image is considered a match if it contains *any* of the terms in the search string. You should treat phrases wrapped in quotation marks as a single term. You may find it useful to use the searchTerm() function written in the **Search Terms** exercise.
- Instead of outputting a list of matched image names, output thumbnails of the matched images that link to the full-sized image. The thumbnails should be 100px in height.

19. **Contact Form:** Create a valid web page called contact.html that contains a form to let users send you a simple message via email. The form should contain a text field for the "from" email address, a text field for the subject text of the email, and a textarea for the body of the message. You should also write a page called send.php to process the email submission.

    The code that you will need to actually send the message is PHP's mail() function:
    bool mail($to, $subject, $message, $additional_headers);
    The additional_headers parameter can be used to specify the "from" address, and it should be of the form "From: <email>"

    You may assume that the email address entered is a valid email address.

20. **Contact Form, revisited:** Modify your solution to contact.html to include the following features:
    - Use the function from the **Emailidate** exercise to validate the "from" address entered by the user.
    - If the email address is invalid, display a message saying such and do not send the message.
    - If the email address is valid, send the message and display a confirmation message that the email was sent. You should also send a confirmation email back to the sender of the message, repeating the message that was sent.