**Tasks:**

1. See the table in the image below.

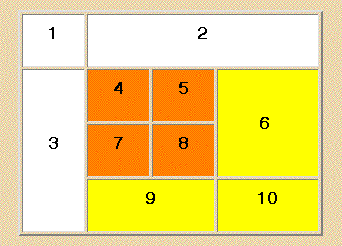
Create the table in two different ways:

- First as a single table

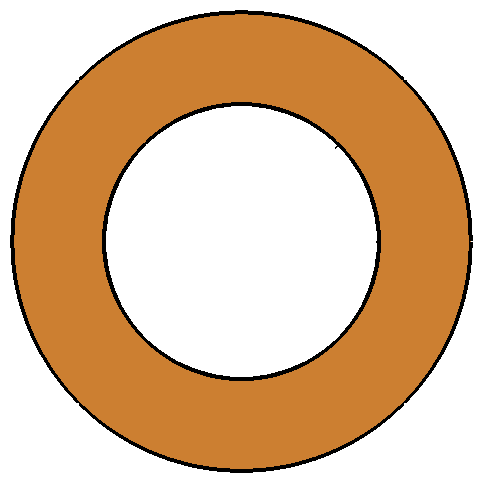
- Second as nested tables

The table is five units wide and four units tall. Feel free to use whichever units you want (for example 1 unit = 50 pixels etc).

Ignore the colored border around the table - that is not required.



2. See the attached image of a ring. Create an image map such that only the colored part of the ring links to a site (say [google.com](http://google.com)) by opening a new window.

Nothing happens if user clicks anywhere else outside the colored ring.

3. Create an image map that assigns four links to shapes within the image you downloaded. The links should be made from the four buildings in the map that are numbered. The links and their associated alternate text are as follows:

1. Link = <http://www.google.com/ncr>

alt text = "Click to go to Google Search Page."

2. Link = <http://www.samachar.com/>

alt text = "Click to go to Indian News Portal."

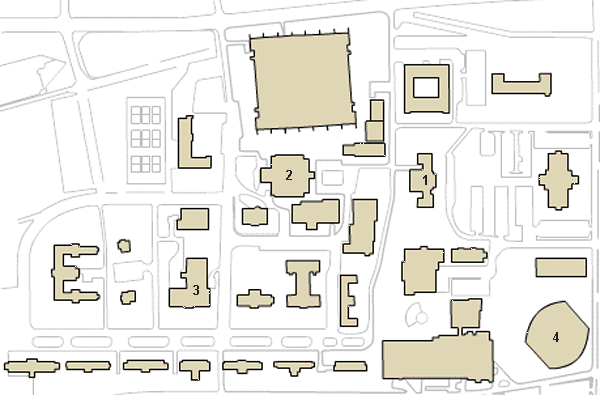
3. Link = <http://www.php.net/>

alt text = "Click to go to PHP Home Page."

4. Link = <http://www.drupal.org/>

alt text = "Click to go to Drupal Home Page."

Be sure to turn off the image's border. As with using an entire image as a link, a browser will place a border around an image used for an image map.





**4. Task Description**  
  
A polite number is a positive integer which can be written as the sum of two or more consecutive positive integers. Other positive  
integers are impolite.  
  
For example the numbers 3, 6, 20, and 11 are all polite. On the other hand, the numbers 16 and 128 are impolite.  
  
**Program Input**:   
The input is a single positive integer value.  
Ex: 4  
  
**Program Output**:   
The program must print the input value, followed by a message that says if the number is polite or impolite.  
Ex: 4 is impolite

**Program:**

<?php  
if(isset($\_POST['submit']))   
{   
    $num = $number = $\_POST['number'];  
    echo "User has submitted the number : <b> $number </b>";  
    while ($num%2 == 0) {  
      $num = $num/2;  
    }  
    if ($num == 1) {echo "<br><b>$number</b> is IMPOLITE.<br>";}  
    else {echo "<br><b>$number</b> is POLITE.<br>";}  
    echo "<br>You can use the following form again to enter a new number.";   
}  
?>  
  
<HTML>  
<HEAD><title>Polite or Impolite</title></HEAD>  
<BODY>   
<FORM method="post" action="<?php echo $\_SERVER['PHP\_SELF']; ?>">  
   <input type="text" name="number"><br>  
   <input type="submit" name="submit" value="Polite or Impolite?"><br>  
</FORM>  
</BODY>  
</HTML>

5. Happy number:

Check given number is a happy number or an unhappy number.

A ***happy number*** is defined by the following process.

1. Starting with any positive integer,

2. Replacing the number by the sum of the squares of its digits,

3. Repeating the process until the number reach 1.

4. If the number reaches 1 at the end of the process, the number is said to be Happy, else, unhappy or sad.

**Example:**

Let us Consider,

Positive Integer = 103

Sum of Digits of 103 = 1² + 0² + 3² = 1 + 0 + 9 = 10

10 = 1² + 0² = 1 + 0 = 1

Solution, 103 is a happy number

**Program:**

<?php // Happy number

$num\_arr = array(86, 32, 4565, 42, 5555, 69);

foreach($num\_arr as $key => $val) {

print $val." is ";

isHappy($val);

}

function isHappy($num) {

$len = strlen($num);

$sum = 0;

for ($i = 0; $i < $len; $i++) {

$char = substr($num, $i, 1);

$sum = $sum + $char\*$char;

}

if(strlen($sum) == 1) {

if($sum == 1) {

print "a happy number <br />";

} else {

print "an unhappy number <br />";

}

} else {

isHappy($sum);

}

}

?>

6. Description:

      1) Form

              Write a form that lets user enter his/her name and an type in an essay (text area).

                      A user's name should contain only alphabetics, white space and dots

                      For essay (All characters are allowed)

                      Minimum number of words allowed = 100

                      Maximum number of words  allowed = 1000

        2. The form action

              Form action should verify that the user name and essay are valid(server side validations)

              Print the number of lines, words and characters in the essay along with the essay.

[Hidden] Objectives

         1) Learn about client side validations (remember that client side validations are optional, users can always by-pass client side checks)(demo firefox tamperdata).

         2) Learn about how php handles GET/POST data

         3) Learn about server side validations.

         4) Learn about cross site scripting.

Description : File Uploads  
            1) Form  
                       Field - 1: text field to enter name of the user  
                       Field - 2: a file field (to upload a file)  
                       Maximum allowed file size 8 MB  
                       File types allowed are .doc and .pdf  
             2) PHP Environment for file uploads  
                      1. Maximum allowed size of POST = 10 MB  
                      2. Maximum Input time 300 sec  
              3) On successful upload. The page should display  
                      i)  Name of the user who uploaded the file  
                     ii)  Name of the file  
                     iii) Size of the file in friendly format (KB or MB based on size)  
                     iv) Date uploaded  
                      v) File MIME type  
              4) In a separate script  write php code that scans the uploads folder( in which all uploads are stored) and displays its contents like a file system tree.  
  
            Objectives:  
                  1.  Editing php runtime configuration  
                  2.  File uploads  
                  3.  Extracting file information

**7. Task Description**

In modern times, paper is abundant and inexpensive, but this hasn't always been true. In ancient times, writing materials such as papyrus and parchment were difficult and time-consuming to produce. To conserve materials, ancient writers often wrote in scriptio continua, meaning that there were no gaps between words and almost no punctuation. The first step in deciphering these ancient documents is to find the divisions between words.

 Write a program that can use a supplied dictionary to parse scriptio continua text in the not-so-ancient language of English.

**Input**

The input file will contain two sections: the dictionary of valid words and the manuscript to be parsed. The section names appear in square brackets. Multiple dictionary words may be listed in lines up to eighty characters in length. Lines in the manuscript section will have no more than eighty letters each. The manuscript input ends with a single period.

**[DICTIONARY]**

**A BROWN DOG FOX JUMPED LAZY OVER QUICK THE**

**[MANUSCRIPT]**

**AQUICKBROWNFOXJUMPEDOVERTHELAZYDOG.**

**Output**

The program must display the manuscript text with spaces between words, with word wrapping applied at eighty characters. All the words in the text must appear in the dictionary; any division that does not identify dictionary words is invalid. There will be only one valid solution. For example, the dictionary will not contain the words JUSTICE, JUST, and ICE. That’s another problem for another time.

**A QUICK BROWN FOX JUMPED OVER THE LAZY DOG.**

**Program:**

<?php

function dictionary($start,$end,$total) {

$total=stristr($total,$start);

$f2=stristr($total,$end);

return substr($total,strlen($start),-strlen($f2));

}

$file="input\_file.txt";

$fp=fopen($file,"r");

if(!$fp) {

echo "File cannot be opened.";

}

$contents=fread($fp,filesize($file));

$dic=trim(dictionary("[DICTIONARY]","[MANUSCRIPT]",$contents));

echo "<br />";

$parts = explode(" ", $dic);

echo "<b>DICTIONARY:</b><br /><br />";

print\_r($parts);

echo "<br /><br />";

$manuscript=trim(dictionary("[MANUSCRIPT]",".",$contents));

echo "<b>MANUSCRIPT:</b><br /><br />";

echo $manuscript."<br /><br /><br />";

$len=strlen($manuscript);

$st\_pos=0;

$str\_len=1;

$i=1;

echo "<b>SINGLE LINE:</b><br /><br />";

while($i<=$len) {

$i++;

$word=trim(substr($manuscript,$st\_pos,$str\_len));

if (in\_array($word,$parts)) {

echo " $word";

$st\_pos=$st\_pos+$str\_len;

$str\_len=1;

} else {

$str\_len++;

}

}

if($len==$st\_pos) {

echo ".";

}

?>

**8. Minesweeper**  
  
Have you ever played Minesweeper? The goal of the game is to find where all the mines are located within a M × N field.  
  
The game shows a number in a square which tells you how many mines there are adjacent to that square. Each square has at most eight adjacent squares. The 4×4 field on the left contains two mines, each represented by a “\*” character. If we represent the same field by the hint numbers described above, we end up with the field on the right:  
  
\*...    \*100  
....    2210  
.\*..    1\*10  
....    1110  
  
**Input**  
The input will consist of an arbitrary number of fields. The first line of each field contains two integers n and m (0 < n,m < 100) separated by a space, which stand for the number of lines and columns of the field, respectively. Each of the next n lines contains exactly m characters, representing the field. Safe squares are denoted by “.” and mine squares by “\*,” both without the quotes.   
  
**Output**  
Print the message Minefield: on the first line alone. The next n lines should contain the field with the “.” characters replaced by the number of mines adjacent to that square.  
  
**Sample Input        Sample Output**  
3 5                  Minefield:  
\*\*...                \*\*100  
.....                33200  
.\*...                1\*100

**Program:**

<?php

$file="matrix.txt";

$fp=fopen($file,"r");

if(!$fp) {

echo "File cannot be opened.";

}

$contents=fread($fp,filesize($file));

$contents1=filter\_var($contents, FILTER\_SANITIZE\_STRING, FILTER\_FLAG\_STRIP\_LOW);

global $rows;

global $cols;

$rows=substr($contents1,0,1);

$cols=substr($contents1,2,1);

$matrix=substr($contents1,3);

$len=strlen($matrix);

$st\_pos=0;

$mat=array();

$res=array();

echo "<b>INPUT:</b><br />";

for($i=($rows-1);$i>=0;$i--) {

for($j=0;$j<$cols;$j++) {

$mat[$i][$j]=substr($matrix,$st\_pos,1);

echo $mat[$i][$j]." ";

$st\_pos++;

}

echo "<br />";

}

echo "<br />";

for($i=($rows-1);$i>=0;$i--) {

for($j=0;$j<$cols;$j++) {

if($mat[$i][$j]=="\*") {

$res[$i][$j]="\*";

}

else {

// echo $i." ".$j."<br />";

$res[$i][$j]=count\_stars($mat,$i,$j);

}

}

}

echo "<br /><br /><b>RESULT:</b><br /><br />";

for($i=($rows-1);$i>=0;$i--) {

for($j=0;$j<$cols;$j++) {

echo $res[$i][$j]." ";

}

echo "<br />";

}

function count\_stars($check\_mat,$x,$y){

$count=0;

for($i=($x-1);$i<=($x+1);$i++) {

for($j=($y-1);$j<=($y+1);$j++) {

if(isset($check\_mat[$i][$j])){

if($check\_mat[$i][$j]=="\*") {

$count++;

}

}

}

}

return $count;

}

?>

**9. Jolly Jumpers**  
  
A sequence of n > 0 integers is called a jolly jumper if the absolute values of the differences between successive elements take on all possible values 1 through n-1.   
  
For example, "1 4 2 3" is a jolly jumper, because the absolute differences are 3, 2, and 1, respectively.   
  
The definition implies that any sequence of a single integer is a jolly jumper. Write a program to determine whether each of a number of sequences is a jolly jumper.  
  
**Input - a simple text file (ex: jolly.txt)**  
Each line of input contains an integer n < 999 followed by n integers representing the sequence.  
  
**Output - display on browser**  
Print the n integers sequence and say “Jolly” or “Not jolly”.  
  
**Sample Input**  
4 1 4 2 3  
5 1 4 2 -1 6  
  
**Sample Output**  
"1 4 2 3" is Jolly  
"1 4 2 -1 6" is Not jolly



**Program:**

<?php

$file = fopen("jolly\_input.txt", "r"); //opening the text file using fopen

//$jolly\_file = file("jolly.txt");

$c = 0;

echo "Input is:<br>";

while (!feof($file)) {

$line[$c] = trim(fgets($file));

echo $line[$c]."<br>"; //displaying input to the browser

$c++;

}

//echo count($line);

//$j=0;

for ($i = 0 ;$i < count($line);$i++) {

//exploding with spaces and storing into an array

$line\_1[$i] = explode(" ", $line[$i]);

}

echo "<br>output is <br>";

//logic to tell whether the number is jolly or not

for ($i=0; $i< count($line);$i++) {

for ($j=1;$j < count($line\_1[$i])-1;$j++) {

//finding the absolute difference

$res[$i][$j] = abs($line\_1[$i][$j]-$line\_1[$i][$j+1]);

$res1[$i][$j] = $j;

}

echo "<pre>";

//print\_r($res1);

echo "</pre>";

if (check\_jolly($res[$i],$res1[$i])) {

for ($j=0;$j < count($line\_1[$i]);$j++) {

echo $line\_1[$i][$j]." ";

}

echo "jolly <br>";

}

else {

for ($j=0;$j < count($line\_1[$i]);$j++) {

echo $line\_1[$i][$j]." ";

}

echo " not jolly <br>";

}

}

function check\_jolly($array\_1,$array\_2) {

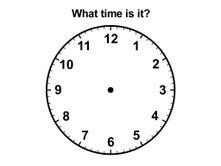
return (array\_diff($array\_1,$array\_2) === array\_diff($array\_2,$array\_1));

}

?>

10. "**Clockhands**" is a fun exercise in the use of php graphics.

A clockface image is attached with this email, but feel free to create and use your own version if you want.  
You are expected to draw the hour and minute hands on it that show the correct time as demonstrated in the link below.  
  
To make it even more interesting you can add a seconds hand and auto-refresh the page every second so that you get real time movement of the hands.



**11. Crypt Kicker**  
  
A common but insecure method of encrypting text is to permute the letters of the alphabet. In other words, each letter of the alphabet is consistently replaced in the text by some other letter. To ensure that the encryption is reversible, no two letters are replaced by the same letter.  
  
Your task is to decrypt several encoded lines of text, assuming that each line uses a different set of replacements, and that all words in the decrypted text are from a dictionary of known words.  
  
**Input**  
The input text file consists of a line containing an integer n, followed by n lowercase words, one per line, in alphabetical order. These n words compose the dictionary of words which may appear in the decrypted text. Following the dictionary are several lines of input. Each line is encrypted as described above.  
  
There are no more than 1,000 words in the dictionary. No word exceeds 16 letters. The encrypted lines contain only lower case letters and spaces and do not exceed 80 characters in length.  
  
**Output**  
Decrypt each line and print it to standard output (browser screen). If there are multiple solutions, any one will do. If there is no solution, replace every letter of the alphabet by an asterisk.   
  
**Sample Input**  
6  
and  
dick  
jane  
puff  
spot  
yertle  
bjvg xsb hxsn xsb qymm xsb rqat xsb pnetfn  
xxxx yyy zzzz www yyyy aaa bbbb ccc dddddd  
  
**Sample Output**  
dick and jane and puff and spot and yertle  
\*\*\*\* \*\*\* \*\*\*\* \*\*\* \*\*\*\* \*\*\* \*\*\*\* \*\*\* \*\*\*\*\*\*

**Program:**

<?php

$input\_file=fopen("input.txt","r"); // reading the input.txt file.

$count=fgets($input\_file); // taking the number of given dictionary words.

$var=0;

for($i=0;$i<$count;$i++){ // taking the given dictionary words into '$dic' array.

$dic[$i]=trim(fgets($input\_file));

}

$alpha=range("a","z"); // taking an array with values from 'a' to 'z'.

$starsArray=array\_fill(0,26,'\*'); // taking a 26 size array with all the values as '\*'s.

while(!feof($input\_file)){

$input\_read=trim(fgets($input\_file)); // taking the given encoded line.

if(feof($input\_file))

break;

$input=explode(" ",$input\_read); // splitting into words and taking them into an input array.

for($i=0;$i<count($input);$i++){ // taking one input string and checking for the same string length in the

for($j=0;$j<count($dic);$j++){ // given dictionary words.

if(strlen($dic[$j])==strlen($input[$i])){ // calling the 'assign\_values' function if the two string

if(assign\_values($dic[$j],$input[$i])){// lengths are equal.

$inindex=$i;

$dicindex=$j;

break;

}

elseif($j==count($dic)-1){

delete\_values($dic[$dicindex]);

$i=$inindex;

$j=$dicindex+1;

}

else{ continue; }

}

}

}

for($i=0;$i<count($input);$i++){ // decrypting the code using the generated format.

for($j=0;$j<strlen($input[$i]);$j++){

$output[$i][$j]=$starsArray[array\_search($input[$i][$j],$alpha)];

}

$outArray[$i]=implode("",$output[$i]);

}

echo "<div style='font-family:courier;'>";

for($i=0;$i<count($input);$i++){

if(in\_array($outArray[$i],$dic)) // checking for the obtained decrypted word in the given dictionary.

echo $outArray[$i]." ";

else{

$stars=array\_fill(0,strlen($outArray[$i]),"\*"); // writing the stars in the output\_file if the word

$stars=implode("",$stars);

echo $stars." ";

}

}

echo "<br />";

}

echo "</div>";

function assign\_values($dicWord,$inputWord){

global $alpha; // globalising the '$alpha' array

global $starsArray; // globalising the '$starsArray' array

for($i=0;$i<strlen($inputWord);$i++){ // generating the format for decrypting the encoded lines.

if(!array\_search($dicWord[$i],$starsArray)){

$starsArray[array\_search($inputWord[$i],$alpha)]=$dicWord[$i];

}

else if(array\_search($dicWord[$i],$starsArray)==array\_search($inputWord[$i],$alpha)){

//$starsArray[array\_search($inputWord[$i],$alpha)]=$dicWord[$i];

}

else{

// delete\_values($starsArray);

return false;

}

} return true;

}

function delete\_values($dicWord){

global $starsArray;

for($i=0;$i<strlen($dicWord);$i++){

$starsArray[array\_search($dicWord[$i],$starsArray)]="\*";

}

}

fclose($input\_file); // closing the input\_file

?>

12. **LCD Time Display**

Display current as it is displayed in a LCD Screen

**Program:**

<?php

$s=4;

$rows=(2\*$s)+3;

$columns=$s+2;

$t=time();

$time=date("Hi",$t); // getting the time in hours and minutes

$clock=str\_split($time); // spliting the string into an array

$space1=array("&nbsp;"); // an array with an element as space

$verticalSlash=array("<b>|</b>"); // an array with an element as ("|")

$dash=array\_fill(1,$s,"<b>-</b>"); // an array with an element as ("-")

$horz=array\_merge($space1,$dash,$space1); // creating (" ----- ") using array merging

$horArray=array($horz); // converting $horz array into two dimensional array

$space2=array\_fill(1,$s,"&nbsp;"); // creating an array with elements as spaces in the keys from 1 to $s

$space3=array\_fill(1,$s+1,"&nbsp;"); // creating an array with elements as spaces in the keys from 1 to $s+1

$space4=array\_fill(0,$s+1,"&nbsp;"); // creating an array with elements as spaces in the keys from 0 to $s+1

$leftVer=array\_merge($verticalSlash,$space3); // creating ("| ") using array merging

$rightVer=array\_merge($space4,$verticalSlash); // creating (" |") using array merging

$twoVer=array\_merge($verticalSlash,$space2,$verticalSlash); // creating ("| |") using array merging

$completeSpace=array\_fill(0,$s+2,"&nbsp;"); // creating an array with complete spaces (" ")

$completeSpace2d=array($completeSpace); // converting $space array into two dimensional array

$twoVers=array\_fill(1,$s,$twoVer);// creating $s ("| |") blocks and converting into two dimensional array

$leftVers=array\_fill(1,$s,$leftVer);// creating $s ("| ") blocks and converting into two dimensional array

$rightVers=array\_fill(1,$s,$rightVer);// creating $s(" |")blocks and converting into two dimensional array

$output=array(); // creating an ouput variable and declaring it as an array

// Evaluating each value based on the current time and pushing those values into output array

for($i=0;$i<count($clock);$i++){

switch($clock[$i]){

case 0: $zero=array\_merge($horArray,$twoVers,$completeSpace2d,$twoVers,$horArray);

array\_push($output,$zero);

break;

case 1: $one=array\_merge($completeSpace2d,$rightVers,$completeSpace2d,$rightVers,$completeSpace2d);

array\_push($output,$one);

break;

case 2: $two=array\_merge($horArray,$rightVers,$horArray,$leftVers,$horArray);

array\_push($output,$two);

break;

case 3: $three=array\_merge($horArray,$rightVers,$horArray,$rightVers,$horArray);

array\_push($output,$three);

break;

case 4: $four=array\_merge($completeSpace2d,$twoVers,$horArray,$rightVers,$completeSpace2d);

array\_push($output,$four);

break;

case 5: $five=array\_merge($horArray,$leftVers,$horArray,$rightVers,$horArray);

array\_push($output,$five);

break;

case 6: $six=array\_merge($horArray,$leftVers,$horArray,$twoVers,$horArray);

array\_push($output,$six);

break;

case 7: $seven=array\_merge($horArray,$rightVers,$completeSpace2d,$rightVers,$completeSpace2d);

array\_push($output,$seven);

break;

case 8: $eight=array\_merge($horArray,$twoVers,$horArray,$twoVers,$horArray);

array\_push($output,$eight);

break;

case 9: $nine=array\_merge($horArray,$twoVers,$horArray,$rightVers,$completeSpace2d);

array\_push($output,$nine);

break;

default: break;

}

}

echo "<samp>"; // displaying the output in sample code format

for($i=0;$i<$rows;$i++){ // printing the three dimensional output

for($j=0;$j<count($output);$j++){

for($k=0;$k<$columns;$k++){

echo $output[$j][$i][$k];

} echo "&nbsp;";

} echo "</br>";

}

echo "</samp>";

?>