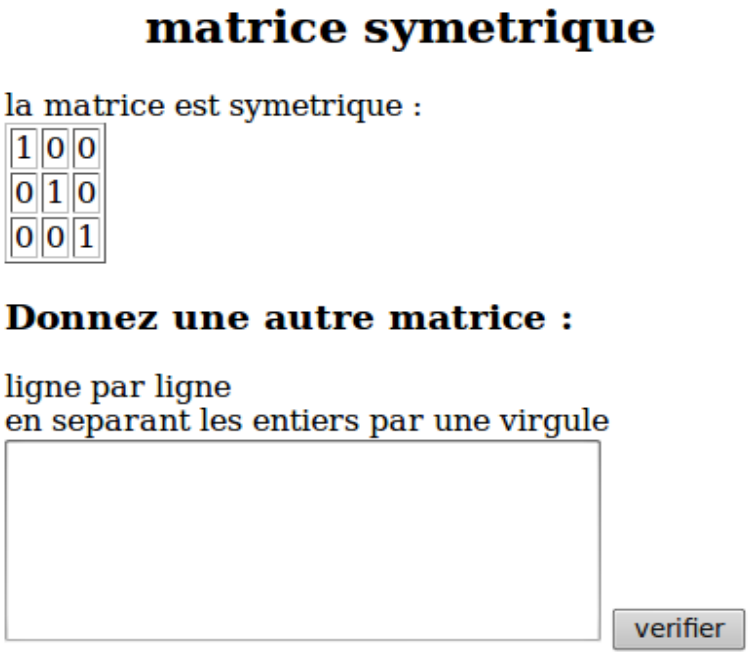
Licence Informatique S4 – Prog. Web - TP n°2

Exercice 1 : Créez un tableau d'entiers variant de 0 à 63, puis à partir de celui ci un autre tableau de nombres variant de 0 à 6.3. Créez ensuite un tableau associatif dont les clés X varient de 0 à 6.3 et dont les valeurs sont sin(X). Affichez le tableau de valeurs dans un tableau HTML.

Exercice. 2 : Complétez le script PHP symetrie.php ci-dessous qui affiche une matrice sous forme d'un tableau HTML si celle-ci est carrée. Si la matrice n'est pas carrée, le script devra l'indiquer sans avoir à afficher la matrice. Par ailleurs, dans le cas d'une matrice carrée, le script doit aussi indiquer si elle est symétrique ou non.

```
<html>
  <body>
    <h2 align=center> matrice symetrique </h2>
<?php
  if(isset($_POST['lignes'])) {
    $tab_lignes=explode("\n", $_POST['lignes']);
    $dim=count($tab_lignes);
    for ($i=0; $i<$dim; ++$i)
      $matrice[$i]=explode(",", trim($tab_lignes[$i]));
    .....
else echo 'la matrice n\'est pas carrée<br>' ;
  }
?>
<h3> Donnez une autre matrice : </h3>
    <form action="symetrie.php" method="POST">
      ligne par ligne <br>
      en separant les entiers par une virgule<br>
      <textarea name="lignes" cols="40" rows="6"></textarea>
      <input type="submit" value="verifier" >
    </form>
  </body>
</html>
```



Vous disposez déjà du script PHP verif_matrice.php :

```
<?php
  function isCarre($mat) {
    $carre=TRUE;
    $dim=count($mat);
    for ($i=0; ($i<$dim) && $carre; ++$i)
      if (isset($mat[$i]) && (count($mat[$i])==$dim))
        for ($j=0; ($j<$dim) && $carre; ++$j) {
          if (!isset($mat[$i][$j]))
            $carre=FALSE;
        }
      else
        $carre=FALSE;
    return $carre;
  }
  function isSymetrique($mat) {
    // ne marche que si elle est carrée
    $sym=TRUE;
    $dim=count($mat);
    for ($i=0; ($i<$dim) && $sym; ++$i)
      for ($j=0; ($j<$i) && $sym; ++$j)
        if ($mat[$i][$j]!=$mat[$j][$i])
          $sym=FALSE;
    return $sym;
  }
?>
```

Exercice 3 : Créez une fonction de validation d'une adresse HTTP, puis la tester par des entrées données par le biais d'un formulaire.

Le modèle d'une adresse devra répondre à la définition suivante :

1. Commencer par « www. »
2. Suivi par des lettres puis éventuellement un point ou un tiret suivis d'un deuxième groupe de lettres
3. Se terminer par un point suivi de l'extension qui peut avoir de 2 à 4 caractères.

Par exemple, les adresses www.machin.com ou www.machintruc.uk sont valides.

Exercice 4 :

Dans cet exercice, on supposera qu'une adresse email a la forme user@nom_de_domaine, où

- user est une séquence de caractères sans @ ni blancs
- nom_de_domaine est composé de 2 groupes. Le premier groupe est constitué d'un ou plusieurs noms séparés par des ".", tandis que le second groupe correspond à un code de pays sur 2 lettres minuscules (fr,de…) ou un code d'activité sur 3 lettres minuscules (com,edu,org…). Chaque nom du premier groupe est composé de lettres minuscules, chiffres ou tirets. Le premier groupe et le second groupe sont séparés par un "."

Ecrivez un script permettant de vérifier, via un formulaire, qu'une adresse email respecte le modèle décrit ci-dessus.

Exercice 5 : Créez un script permettant d'enregistrer la date de chaque connexion dans un fichier *liste_connexion.txt*. Après chaque enregistrement ce même script devra ensuite procéder à la lecture des données du fichier en les affichant.

Exercice 6 : Ecrire un script permettant de compter les occurrences de l'adjectif « bon » ou « bons » dans une phrase renseignée par le biais d'un formulaire. Vous utiliserez obligatoirement la fonction preg_match_all.

Exercice 7 : Ecrire (puis tester via un formulaire) une fonction PHP *bizarre($arg_1)* où $arg_1 correspond à une chaîne de caractères et qui retourne en résultat une chaîne de caractère ayant la propriété suivante :

Pour chaque mot *m* composé uniquement de chiffres dans *$arg_1*, *m* est remplacé dans le résultat par la chaîne suivante (sans les guillemets) : "N.N" où N correspond à la valeur de m multiplié par 2. Exemple : si m="50" il devient alors "100.100" dans la chaîne résultat.

Exercice 8 : Ecrire (puis tester via un formulaire) une fonction php *foo* qui prend en argument une chaîne *$chaine1* et retourne une chaîne qui correspond à *$chaine1* mais dans laquelle chaque lettre des mots ne contenant que des consonnes, est remplacée par *1*.

Exemple : si $chaine1='Vive le PHP et le XML' alors *foo($chaine1)* retourne 'Vive le 111 et le 111'

Exercice 9 : Ecrire une fonction php *foo2* qui prend en argument une chaine $chaine1 et qui retourne un tableau contenant toutes les valeurs des options href situés dans un élément de type *<a href="...">* dans $chaine1 (i.e., ce qui correspond au points de suspension). On supposera qu'il n y a aucun espace inutile dans ces éléments et qu'après les guillemets fermants on trouve immédiatement le chevron fermant. Par ailleurs, les valeurs des options href ne contiennent aucun chevron ouvrant ou fermant ni de guillements.Testez la fonction en renseignant $chaine1 via une balise *textarea* et en affichant le retour de la fonction sous la forme d'un tableau HTML.

Exercice 10 : Ecrire (puis tester via un formulaire) une fonction php permettant de rendre cliquables les liens, d'un texte donné en argument, qui sont placés entre deux # (on supposera que chaque phrase comprise entre # et # est une adresse web réelle). Modifier la fonction pour rendre cliquables ces liens uniquement lorsqu'ils sont précédés de http:// ou https://. Dans tous les cas, dans le texte final, les # encadrants les liens qui deviennent cliquables doivent avoir disparu.

Exercice 11 : Ecrire une fonction PHP compute*($arg_1),* où $arg_1 est une chaîne de caractères correspondant à une suite d'opérations entre entiers naturels, qui retourne un entier correspondant au résultat de la série d'opérations représentée par $arg_1. **Le résultat doit impérativement tenir compte de l'ordre de priorité des opérateurs.**

Pour cet exercice on fait les hypothèses suivantes

- Les opérations sont uniquement des multiplications et/ou des additions.
- On ne tient pas compte des éventuels dépassements d'entiers.
- Il n'y aucun espace dans $arg_1 et aucune parenthèse.

Ainsi par exemple si $arg_1= "1+4+6*4+6+4" alors la fonction *compute* retourne 39.

Rappel : La priorité de la multiplication est supérieure à la priorité de l'addition.

Comme pour les exercices précédents, testez la fonction via un formulaire.