Licence Informatique S4 – Prog Web - TP n°4

Exercice 1 : Créez une page contenant un formulaire de saisie des deux codes couleur préférés du visiteur du site pour la couleur de fond et le texte de la page. Les enregistrez dans un cookie valable deux mois. À la prochaine ouverture de la page, récupérez les valeurs qui y ont été enregistrées et créez un style utilisant ces données. Le fichier traitant les données renseignées dans le formulaire (i.e., valeur de l'attribut *action*) doit être le même que celui permettant d'afficher le formulaire. Dans le cas où un utilisateur clique sur le bouton *submit* du formulaire sans avoir renseigné la couleur de fond (resp. la couleur du texte), on utilisera, si cette dernière existe, le précédente qui a été renseignée : sinon on utilisera le blanc (resp. le noir).

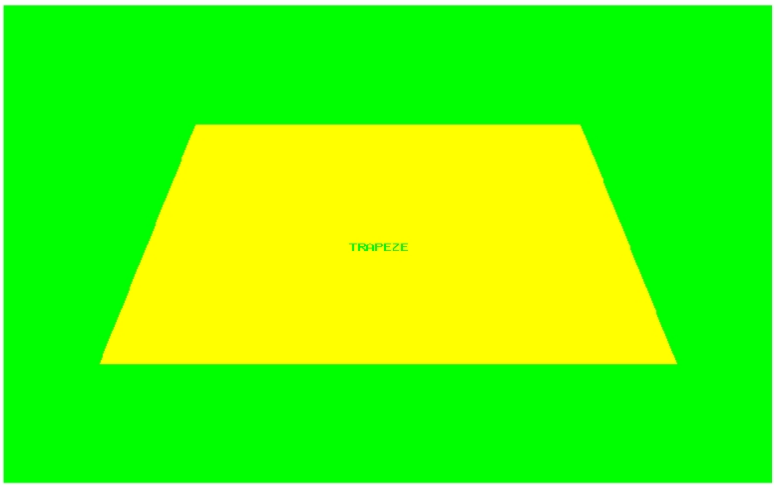
Exercice 2 : Reprendre l'exercice 1 en enregistrant les préférences du visiteur dans des variables de session.

En particulier, modifiez votre page en y ajoutant un lien vers une autre page quelconque que l'on appellera X. La page X doit contenir du texte et un lien pour revenir vers le formulaire. Toujours en utilisant uniquement les variables de sessions, l'utilisateur doit pouvoir switcher entre ces 2 pages, tout en conservant un affichage selon les dernières préférences qu'il a renseignées.

Exercice 3 : Créez un livre d'or qui n'affiche que les cinq derniers avis donnés par les visiteurs du site et qui invite un utilisateur à y écrire via un formulaire. Le livre d'or devra être stocké sous la forme d'un fichier et devra gérer les accès concurrentiels (e.g., verrouillage en lecture et en écriture pendant que le script est en train d'écrire dans le fichier).

Exercice 4 :

Ecrire un script permettant de créer une image de 800*600 pixels avec une couleur de fond verte. Sur ce fond devra apparaître un trapèze isocèle rempli de jaune dans lequel le mot « trapèze » sera écrit au centre. Le rendu visuel aura donc l'aspect suivant :



Exercice 5 :

Le fichier php en fin d'énoncé demande à l'utilisateur de remplir un bon de commande en complétant un formulaire comme dépeint dans la figure ci-dessous.

Ce formulaire ne permet la saisie que d'un article à la fois. Ce dernier comprend trois zones de saisie ainsi que 4 boutons Submit, chacun correspondant à une action particulière : un bouton Ajouter qui permet d'ajouter un article à la commande, un bouton Vérifier qui permet d'afficher l'ensemble de la commande et le montant total (comme illustré dans la figure ci-dessous), un bouton Enregistrer qui permet de stocker la commande dans un fichier texte (hors du cadre de l'exercice, il serait préférable de stocker les informations dans une base de données) et enfin un bouton Logout qui permet de fermer la session.

Cependant dans le script, il manque les parties permettant d'ajouter un article et d'enregistrer la commande dans un fichier. Complétez le script afin de remédier à cet oubli.

Concernant l'enregistrement on ne s'occupera pas du problème de l'accès concurrentiel. Par contre, en reprenant l'exemple décrit par la figure ci-dessus, l'enregistrement dans le fichier devra avoir la forme suivante :

526987 ; crayon ; 2.00
8796541 ; gomme ; 1.00
9874123 ; cahier ; 4.00

```
<?php
session_start();
//AJOUTER
…..
//VERIFIER
if($_POST["envoi"]=="VERIFIER")
{
echo "<table border=\"1\" >";
echo "<tr><td colspan=\"3\"><b>Récapitulatif de votre
commande</b></td>";
echo "<tr><th> code </th><th> article </
th><th> prix </th>";
$total=0;
$tab_code=explode("//",$_SESSION['code']);
$tab_article=explode("//",$_SESSION['article']);
$tab_prix=explode("//",$_SESSION['prix']);
for($i=1;$i<count($tab_code);$i++)
{
echo "<tr> <td>{$tab_code[$i]}</td> <td>{$tab_article[$i]}
</td><td>".sprintf("%01.2f", $tab_prix[$i])."</td>";
$prixtotal+=$tab_prix[$i];
}
echo "<tr>   <td colspan=2> PRIX TOTAL </td> <td>". sprintf("%01.2f",
$prixtotal)."</td>";
echo "</table>";
```

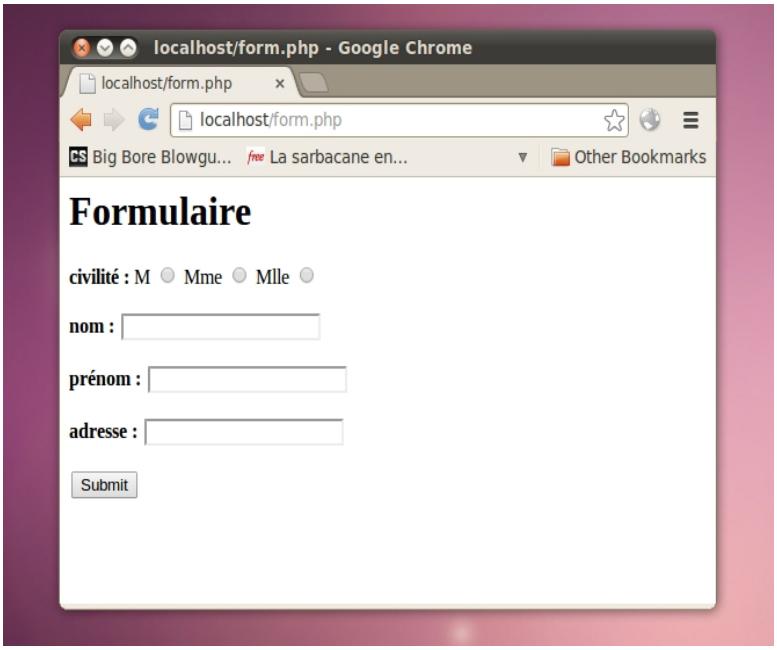
```
}
//ENREGISTRER
…..
//LOGOUT
if($_POST["envoi"]=="LOGOUT")
{
     session_unset();
     session_destroy();
     echo "<h3>La session est terminée</h3>";
}
$_POST["envoi"]="";
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.1//EN"
    "http://www.w3.org/TR/xhtml11/DTD/xhtml11.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="fr">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-
1">
<title>Gestion de panier</title>
</head>
<body>
<form action="<?php $_SERVER['PHP_SELF'] ?>" method="post"
enctype="application/x-www-form-urlencoded">
<fieldset>
<legend><b>Saisies d'articles</b></legend>
<table>
<tbody>
<tr>
<th>code : </th>
<td> <input type="text" name="code" /></td>
</tr>
<tr>
<th>article : </th>
<td><input type="text" name="article" /></td>
</tr>
<tr>
<th>prix :</th>
<td><input type="text" name="prix" /></td>
</tr>
<tr>
<td colspan="3">
<input type="submit" name="envoi" value="AJOUTER" />
<input type="submit" name="envoi" value="VERIFIER" />
<input type="submit" name="envoi" value="ENREGISTRER" />
<input type="submit" name="envoi" value="LOGOUT" />
</td>
</tr>
</tbody>
</table>
</fieldset>
</form>
</body>
</html>
```

Exercice 6 :

Complétez le fichier.php ci-dessous. Il doit permettre d'enregistrer le nom de la page du site préférée du visiteur dans un cookie. Le nom de cette page aura été demandée via le formulaire déjà présent dans le script. Lors de la prochaine visite, le visiteur devra être redirigé automatiquement vers sa page préférée. Pour cela on utilisera la fonction header("location:MaPage") qui permet la redirection vers la page de nom MaPage.

```
<?php
...
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=iso-
8859-1" />
<title>Choisissez votre page préférée</title>
</head>
<body>
<form method="post" action="<?php echo $_SERVER['PHP_SELF'];?>">
<fieldset>
<legend>Choisissez votre page préférée</legend>
<select name="mapage">
<option value="toto.php">Accueil</option>
<option value="tata.php">Cinéma</option>
<option value="titi.php">Voyages</option>
</select>
<br />
<input type="submit" value="Envoyer" />  
<input type="reset" value="Effacer" />
</fieldset>
</form>
</body>
</html>
```

Exercice 7 :



Créer une page contenant un formulaire correspondant à celui décrit dans la figure située ci-dessus. Ce formulaire devra avoir la particularité décrite dans les paragraphes qui suivent.

Si un utilisateur *Y* n'a jamais cliqué sur le bouton *submit*, alors lors de la visite de Y sur la page, le formulaire devra s'afficher comme sur la figure : on dit que le formulaire est vierge. Sinon, l'utilisateur *Y* a déjà cliqué sur le bouton *submit* et on dénote par *t* la date correspondant à la dernière fois où *Y* a cliqué sur *submit*. Nous distinguons 2 cas.

1er cas : A la date *t*, tous les champs du formulaire avaient été renseignés : quand *Y* visite à nouveau la page, le formulaire est vierge.

2ième cas : A la date *t*, tous les champs du formulaire n'avaient pas été renseignés. On considère alors 2 sous-cas.

1er sous-cas : Quand *Y* visite à nouveau la page la date *t* est antérieure de moins de 4 semaines : le formulaire est alors pré-rempli avec toutes les données renseignées par *Y* à la date *t* (s'il y en a).

2eme sous-cas : Quand Y visite à nouveau la page, il s'est écoulé au moins 4 semaines depuis la date *t* : le formulaire affiché doit alors être vierge.

Pour effectuer ce qui est demandé dans l'exercice, il convient, de surcroît, de prendre en compte les 2 remarques suivantes.

Remarque 1 : il faut utiliser des cookies.

Remarque 2 : un champs qui est complété par une chaîne ne contenant que des espaces ne doit pas être considéré comme un champs qui a été renseigné.

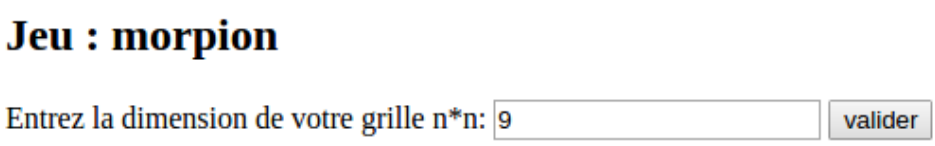
Exercice 8.

Le but de cet exercice est de programmer un mini-jeu du morpion en php.

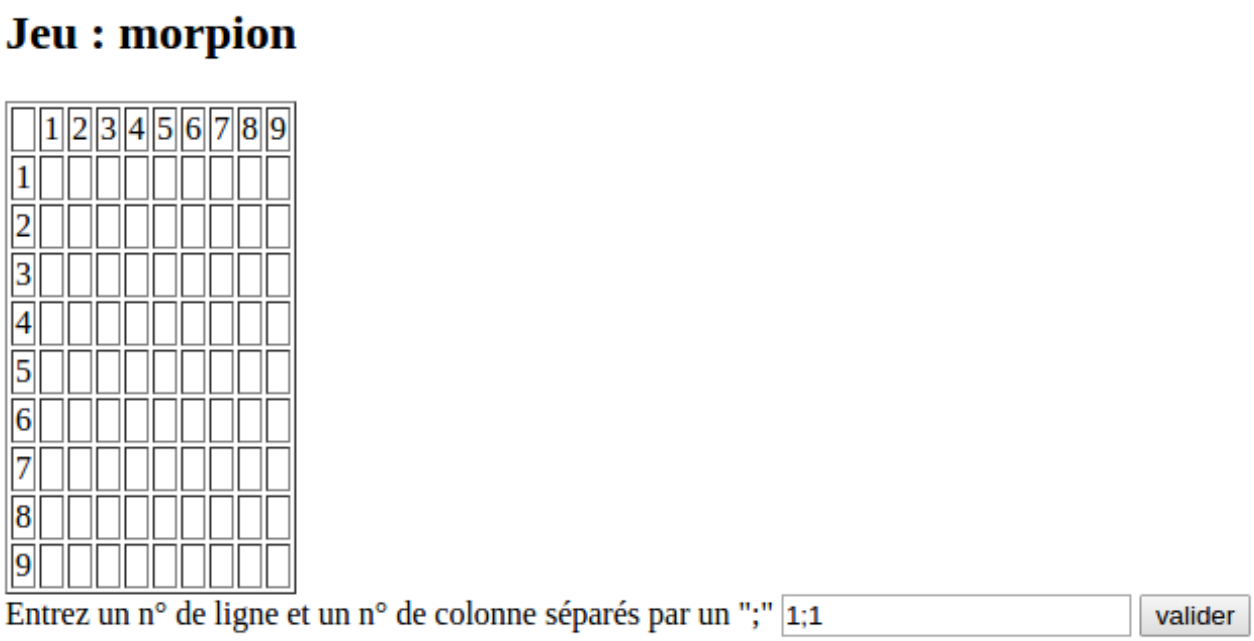
Etant donnée une grille n*n (avec n>4), 2 joueurs s'affrontent en plaçant à tour de rôle un sigle spécifique dans la grille : un joueur place des X tandis que l'autre place des O. Le but pour un joueur est de faire une ligne de 5 sigles (i.e., une ligne de X si le sigle du joueur est X) horizontalement, verticalement ou en diagonale. Votre jeu devra respecter les spécifications qui sont données dans la suite de cet énoncé.

Un des joueurs sera le client qui placera des sigles X tandis que le rôle du second joueur sera pris en charge par le serveur qui jouera des coups aléatoirement (mais néanmoins valides) en plaçant des O.

Dans un premier temps, le client devra être invité à indiquer la dimension souhaitée pour la grille comme dans la figure ci-dessous. On supposera pour cette étape que le client est « intelligent » et qu'il fournit nécessairement un nombre entier supérieur ou égal à 5.

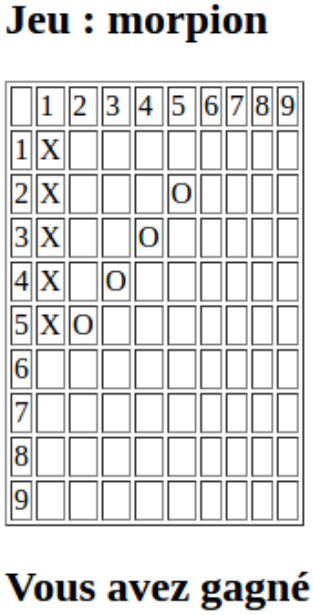


Une fois cela effectué, l'utilisateur devra voir apparaître une nouvelle page contenant une grille aux dimensions souhaitées : les lignes et les colonnes étant numérotées de 1 à n, où n est la dimension préalablement indiquée. C'est l'utilisateur qui joue toujours le premier coup.

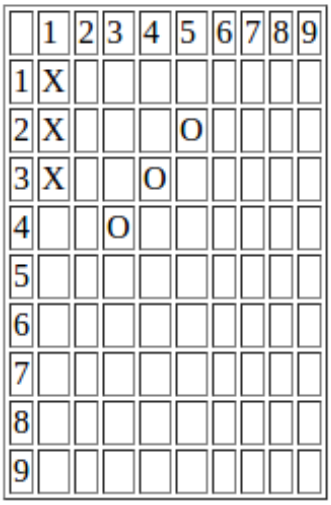


Quand il joue un coup, l'utilisateur renseigne la case où il souhaite placer un sigle X en donnant le numéro de ligne suivi du numéro de colonne dans le formulaire situé en dessous de la grille (le numéro de la ligne et de la colonne doivent être séparés par un point-virgule). Quand l'utilisateur clique sur le bouton *valider*, le serveur vérifie que le coup est valide, c'est-à-dire que la case demandée est bien une case de la grille et que celle-ci est vide. Dans le cas où le coup de l'utilisateur est non valide, le serveur doit inviter à nouveau l'utilisateur à renseigner une case en renvoyant une page dans laquelle la grille est inchangée. Sinon deux cas se présentent.

Cas 1 : L'utilisateur gagne grâce au coup valide qu'il vient d'effectuer. Le serveur renvoie une page comme sur la figure ci-dessous.



Jeu : morpion



Entrez un n° de ligne et un n° de colonne séparés par un ";" 1;1 valider

Cas 2 : Le coup qui a été joué par l'utilisateur ne lui permet pas de gagner. Le serveur retourne alors une page contenant la grille mise à jour avec le dernier coup du joueur ainsi que le dernier coup du serveur : si ce dernier coup est gagnant pour le serveur alors l'utilisateur devra voir afficher, en plus de la grille, le message “Vous avez perdu”. Sinon, l'utilisateur est à nouveau invité à renseigner une case comme ci-dessous.

Question 1.

Ecrivez une fonction IsValidLineOrColumn($mat,$sig,$k) qui prend en entrée une matrice carrée $mat, un sigle $sig et un entier $k : la fonction retourne vraie si la matrice $mat contient $k sigles consécutifs sur une même ligne ou sur une même colonne (i.e. horizontalement ou verticalement).

Question 2.

Ecrivez une fonction IsValidDiagonal($mat,$sig,$k) qui prend en entrée une matrice carrée $mat, un sigle $sig et un entier $k : la fonction retourne vraie si la matrice $mat contient $k sigles consécutifs en diagonale.

Question 3.

Utilisez les fonctions IsValidDiagonal($mat,$sig,$k) et IsValidLineOrColumn($mat,$sig,$k) pour écrire une fonction IsValid($mat,$sig,$k) qui prend en entrée une matrice carrée $mat, un sigle $sig et un entier $k : la fonction retourne vraie si la matrice $mat contient $k sigles consécutifs horizontalement ou verticalement ou en diagonale.

Question 4.

Ecrire une fonction Insert($mat,$n,$sig,$l,$c) qui prend en entrée une matrice $mat de

dimension $n, un sigle $sig, un entier $l (correspondant à un numéro de ligne) et un entier c (correspondant à un numéro de colonne). La fonction doit renvoyer faux dans les cas suivants :
- $l>=$n ou $c>=$n ou $l<0 ou $c<0
- $mat[$l][$c] est différent de NULL

Dans tous les autres cas, la fonction retourne une matrice $mat2 correspondant à la matrice $mat dans laquelle $mat[$l][$c]==$sig.

Question 5.

Ecrivez une fonction AfficheMorpion($mat,$n) qui prend en entrée une matrice carrée $mat et sa dimension $n. La fonction doit retourner une chaîne qui contient le code d'un tableau HTML correspondant au contenu de la matrice $mat. En particulier, ce code HTML devra correspondre uniquement à tout ce qui est entre les balises <table border= '1'> et </table> (en incluant ces 2 balises). Dans ce code HTML les lignes (resp. colonnes) devront être numérotées de 1 à $n.

Par exemple, si $mat[0][0]=="X" et $mat[0][1]=="O", tandis que toutes les autres cases de $mat valent NULL alors la fonction AfficheMorpion($mat,5) retournera le code ci-dessous (attention il y a un décalage de 1 entre les numéros des cases du tableau $mat et les numéros des cases du tableau HTML).

```
<table border='1'>
<tr>
<td></td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td>
</tr>
<tr>
<td>1</td><td>X</td><td>0</td><td></td><td></td><td></td>
</tr>
<tr>
<td>2</td><td></td><td></td><td></td><td></td><td></td><td></td>
</tr>
<tr>
<td>3</td><td></td><td></td><td></td><td></td><td></td><td></td>
</tr>
<tr>
<td>4</td><td></td><td></td><td></td><td></td><td></td><td</td>
</tr>
<tr>
<td>5</td><td></td><td></td><td></td><td></td><td></td><td></td><td>/td>
</tr>
</table>
```

Question 6.

En utilisant au maximum les fonctions écrites lors des 5 questions précédentes, proposez une solution en PHP/HTML permettant d'effectuer ce mini-jeu du morpion (en respectant les conditions indiquées au début de l'exercice). Si nécessaire, en plus des fonctions des questions précédentes, vous pouvez écrire une ou plusieurs autres fonctions afin de faciliter la présentation du code final.