

## PHP – Ynov Ingesup B1

### TP1





#### Exercice 1

Générez une page Web qui affiche le jour et l'heure du serveur.

Pour cela, utilisez la fonction `date`. Une description complète de cette fonction est proposée sur le site [php.net](http://php.net). Essayez les différents formats d'affichage proposés par cette fonction.

#### Exercice 2

Nous souhaitons écrire une fonction `genererGalerie($informations)` qui génère automatiquement une galerie de portraits à partir d'informations présentes dans la variable `$informations` :

Prénom	Nom	Photo
Victor	Hugo	
Jean	De La Fontaine	
Pierre	Corneille	
Jean	Racine	

Le paramètre `$informations` contient un tableau de tableaux associatifs. Le nombre de tableaux associatifs présents dans le tableau `$informations` est égal aux nombres de personnages dans la galerie.

Chaque tableau associatif `$t` contenu dans le tableau `$informations` contient le nom, le prénom et un lien vers la photo du personnage. Ces trois données sont respectivement associées aux clés `"nom"`, `"prenom"` et `"photo"`.

Par exemple, le code suivant doit générer le tableau précédent :

```
$p1 = array("prenom"=>"Victor", "nom"=>"Hugo",  
"photo"=>'http://upload.wikimedia.org/wikipedia/commons/5/5a/Bonnat_Hugo001z.jpg');  
$p2 = array("prenom"=>"Jean", "nom"=>"de La Fontaine",  
"photo"=>'http://upload.wikimedia.org/wikipedia/commons/e/e1/La_Fontaine_par_Rigaud.jpg');  
$p3 = array("prenom"=>"Pierre", "nom"=>"Corneille",  
"photo"=>'http://upload.wikimedia.org/wikipedia/commons/2/2a/Pierre_Corneille_2.jpg');  
$p4 = array("prenom"=>"Jean", "nom"=>"Racine",  
"photo"=>'http://upload.wikimedia.org/wikipedia/commons/d/d5/Jean_racine.jpg');  
$informations = array($p1, $p2, $p3, $p4);  
genererGalerie($informations);
```

#### Exercice 3

Écrivez une fonction PHP *analyser(\$strA, \$strB)* qui prend en entrée :

1. une chaîne de caractères *\$strA* représentant un ensemble de règles de la forme *a->b* (séparées par des virgules). Chaque règle *a->b* associe une clé *a* à un mot *b*;
2. une chaîne de caractères *\$strB* représentant un ensemble de clés (séparées par des virgules).

La fonction doit construire la liste des mots associés par les règles définies dans *\$strA* aux clés définies dans *\$strB*. Cette liste doit être retournée par la fonction triée par ordre alphabétique et ne doit pas contenir de doublon.

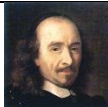



Par exemple, l'appel *analyser("a->b, c->d, f->g", "a, f")* doit retourner la chaîne *"b, g"*.

#### Exercice 4

Vous remarquez que le tableau généré par la fonction *genererGalerie* de l'exercice 2 affiche les personnages en fonction de leur ordre d'apparition dans le tableau *\$informations*.

Ecrivez une fonction de comparaison personnalisée afin de trier les tableaux associatifs présents dans le tableau *\$informations* de façon à respecter l'ordre alphabétique des noms puis des prénoms lorsque les noms sont identiques.

Une fois cette fonction de comparaison écrite, modifiez la fonction *genererGalerie* de l'exercice 2 de sorte qu'elle affiche la galerie de personnages en respectant l'ordre alphabétique :

Prénom	Nom	Photo
Pierre	Corneille	
Jean	De La Fontaine	
Victor	Hugo	
Jean	Racine	

#### Exercice 5

Le but de cet exercice est de préparer des fonctions utiles à la réalisation d'un site de sondages.

Un sondage est décrit ici par un tableau associatif qui contient la question et un tableau de réponses possibles. La question a comme clé "question" et le tableau de réponses a comme clé "reponses".

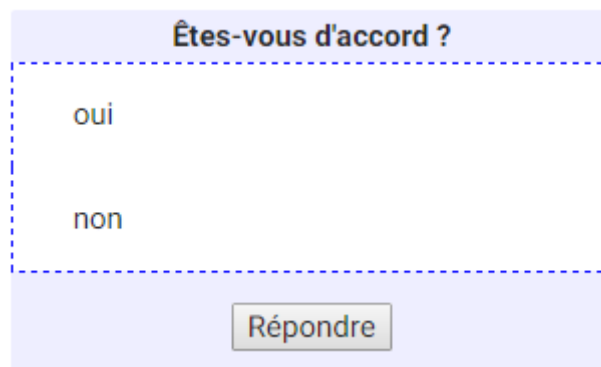
Par exemple, le sondage qui a comme question "Êtes-vous d'accord ?" et comme seule réponses possibles "oui" et "non" est décrit dans le tableau associatif

```
$sondage = array("question"=>"Êtes-vous d'accord ?", "reponse"=>array("oui", "non")).
```

Dans un premier temps, écrivez la `genererFormulaire($sondage)` qui génère à partir du sondage `$sondage` un formulaire de vote.

Le formulaire de vote contient un bouton radio pour chaque réponse possible et un bouton pour poster la réponse.

L'appel de la fonction `genererFormulaire($sondage)` avec le tableau donné plus haut doit générer le formulaire suivant :



Êtes-vous d'accord ?

☐ oui

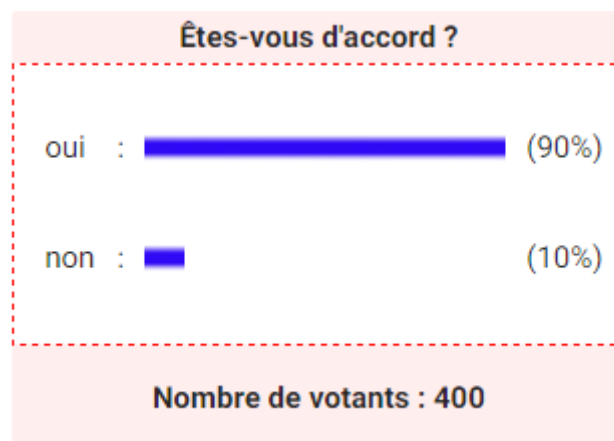
☐ non

Répondre

Dans un deuxième temps, écrivez une fonction `genererResultat($sondage, $vote)` qui génère un graphique présentant aux visiteurs les résultats du sondage.

Le tableau `$voix` contient le nombre de voix que chaque réponse du sondage a obtenues (les tableaux sont organisés de sorte que les réponses et les nombres de voix soient dans le même ordre).

L'appel `genererResultat($sondage, array(360, 40))` doit générer ce qui suit :



Exemple de réalisation d'une barre de pourcentages :

CSS

```
.graph{
    width:200px;
    border:1px solid #000;
    background-color: #ccc;
}

.bar{
    height:10px;
    background-color: #F4661B;
}
```

PHP

```
<?php
    $vote= 38;
    $totalVotes = 100;
    $pourcentVote = round(($vote/$totalVotes)*100);

    echo '<div class="graph"><div class="bar" style="width:', $pourcentVote, '%></div></div>';
?>
```

## Exercice 6

Le but de cet exercice est de réaliser un moteur de recherche de sondages.

Pour ce faire, vous écrivez une fonction *rechercheSondage(\$sondages, \$motcle)* qui prend en paramètre le tableau *\$sondages* contenant un ensemble de sondages et un mot-clé *\$motcle*. Les sondages sont décrits de la même façon que dans l'exercice précédent. La fonction doit retourner un tableau contenant tous les sondages qui ont le mot-clé dans leur question.

Le code suivant doit générer les deux formulaires donnés en dessous :

```
$s1 = array("question"=>"Êtes-vous heureux ?", "reponse"=>array("oui", "non", "peut-être"));
$s2 = array("question"=>"Êtes-vous triste ?", "reponse"=>array("oui", "non"));
$s3 = array("question"=>"Existe-t-il des gens heureux ?", "reponse"=>array("oui", "non"));
$sondages = array($s1, $s2, $s3);
$t = rechercheSondage($sondages, "heureux");
foreach ($t as $sondage) genererFormulaire($sondage);
```

Êtes-vous heureux ?

oui

non

peut-être

Répondre

Existe-t-il des gens heureux ?

oui

non

Répondre

### Exercice 7

Dans cet exercice, vous devez écrire une fonction *ChangerBase(\$n, \$baseFrom, \$baseTo)* qui prend en paramètre une chaîne de caractères *\$n* représentant un nombre écrit en base *\$baseFrom* et qui retourne une chaîne représentant ce même nombre en base *\$baseTo*.

Vous ne devez utiliser que les fonctions PHP qui permettent de manipuler les chaînes de caractères. Je vous propose de décomposer le problème et d'écrire :

1. une fonction *ConvertirEnEntier(\$n, \$baseFrom)* qui convertit le nombre *\$n* écrit en base *\$baseForm* en entier.

```
ConvertirEnEntier("10", 2) doit retourner l'entier 2.  
ConvertirEnEntier("23", 7) doit retourner l'entier 17.  
ConvertirEnEntier("FF", 16) doit retourner l'entier 255.  
ConvertirEnEntier("123", 10) doit retourner l'entier 123.  
ConvertirEnEntier("A", 11) doit retourner l'entier 10.
```

2. une fonction *ConvertirEntierVersBase(\$entier, \$baseTo)* qui retourne une chaîne représentant l'entier *\$entier* en base *\$baseTo*.

ConvertirEntierVersBase(2, 2) doit retourner la chaîne "10". ConvertirEntierVersBase(17, 7) doit retourner la chaîne "23". ConvertirEntierVersBase(255, 16) doit retourner la chaîne "FF". ConvertirEntierVersBase(123, 10) doit retourner la chaîne "123". ConvertirEntierVersBase(10, 11) doit retourner la chaîne "A".
---

Combinez ensuite les deux fonctions précédentes pour obtenir la fonction *ChangerBase*.

### Exercice 8

Nous voulons générer des pyramides de chiffres de la forme suivante :

1 11 21 1211 111221 312211 13112221
---

La première chaîne est toujours égale à "1". Pour obtenir la chaîne suivante, on lit les chiffres de la gauche vers la droite : on compte le nombre de chiffres identiques consécutifs, et on écrit ce nombre suivi du chiffre dans la chaîne suivante. Par exemple, après la chaîne "111221", on trouve la chaîne "312211" car, dans la première chaîne, on a trois "1" consécutifs, deux "2" consécutifs, puis un "1".

Pour générer les pyramides, écrivez les fonctions :

1. *nextSuite(\$str)* qui, à partir d'une chaîne contenu dans la variable *\$str*, calcule la chaîne suivante;
2. *printSuite(\$i)* qui génère les *\$i* premières chaînes (en partant de la chaîne "1").