EXERCICE 1
RÉDACTION DE FONCTIONS UTILITAIRES

1. Pour commencer

1. En *Windows*, copiez le dossier **exercice01** entreposé sur P dans votre dossier **L:\420-4W5**.
2. Ouvrez le dossier pour constater la présence des fichiers **en-tete.php**, **index.css** et **pied-page.php** en plus du fichier **index.php**.
3. Copiez maintenant la dernière version des fichiers **librairie-date.php** et **librairie-generale.php** dans ledit dossier.
4. Démarrez l'application **NetBeans IDE**, puis créez un projet.
5. Exécutez l'application **index.php** dans le navigateur. Notez qu'il est normal que l'application plante (ligne 21) puisque la fonction *aujourdhui()* n'existe pas encore.

2. Travail à effectuer

On vous demande de créer une série de fonctions utilitaires, de les tester de façon exhaustive en utilisant le code source fourni dans *index.php*, puis de les copier dans le fichier **librairie-exercice01.php**.

N'hésitez pas à vous inspirer du code source des fonctions *JavaScript* que vous avez créé en 420-2W5 à moins que des contraintes ne soient imposées.

Exécutez le démonstrateur pour avoir une bonne idée du travail à effectuer :

http://424w.cgodin.qc.ca/lmbrousseau/exercice01/index.php

Signalez toute erreur qui aurait pu se glisser dans l'énoncé ou le démonstrateur.

3. Pour terminer

1. Compressez le dossier **exercice01**, puis envoyez-le via LÉA.

FIN DE L'EXERCICE…

Fonctions à créer, tester, puis à copier dans le fichier *librairie-exercice01.php*

1. aujourdhui

Prototype : **`function aujourdhui($binAAAAMMJJ=true)`**

But : Retourne la date courante en format *aaaa-mm-jj* (par défaut) ou *jj-mm-aaaa*.

Exemples :
`aujourdhui()` retourne **`2018-01-31`**
`aujourdhui(true)` retourne **`2018-01-31`**
`aujourdhui(false)` retourne **`31-01-2018`**

2. bissextile

Prototype : **`function bissextile($intAnnee)`**

But : Retourne **`true`** si l'année passée en argument est bissextile; autrement **`false`**.

Contrainte : Vous devez utiliser la fonction `date()`.

Exemples :
`bissextile(2016)` retourne **`true`**
`bissextile(2018)` retourne **`false`**

3. nombreJoursAnnee

Prototype : **`function nombreJoursAnnee($intAnnee)`**

But : Retourne le nombre de jours de l'année saisie en argument (365 ou 366).

Contrainte : Vous devez utiliser la fonction `bissextile()`. (Vous aurez compris qu'il ne sert à rien de créer des fonctions si vous ne les utilisez pas !)

Exemples :
`nombreJoursAnnee(2016)` retourne **366**
`nombreJoursAnnee(2018)` retourne **365**

4. nombreJoursMois

Prototype : **`function nombreJoursMois($intMois, $intAnnee)`**

But : Retourne le nombre de jours du mois/année saisi en argument (28 à 31).

Contrainte : Vous devez utiliser la fonction `date()`.

Exemples :
`nombreJoursMois(2, 2016)` retourne **29**
`nombreJoursMois(2, 2018)` retourne **28**
`nombreJoursMois(4, 2018)` retourne **30**
`nombreJoursMois(5, 2018)` retourne **31**

5. nombreJoursEntreDeuxDates

Prototype : **`function nombreJoursEntreDeuxDates($strDate1, $strDate2)`**

But : Retourne le nombre de jours entre **`$strDate1`** et **`$strDate2`**. Une date ne peut être antérieure au 14 décembre 1901.

Contrainte : Vous devez utiliser la fonction `strtotime()`.

Exemples :
```
nombreJoursEntreDeuxDates("14-12-1901", aujourdhui(false))
nombreJoursEntreDeuxDates("1901-12-14", aujourdhui())
```
retourne **`42419`** si la date du jour est le *2 février 2018*.

6. extraitJSJJMMAAAAv2

Prototype : **`function extraitJSJJMMAAAAv2(&$intJourSemaine, &$intJour, &$intMois, &$intAnnee)`**

But : Mise à niveau de la fonction `extraitJSJJMMAAAA` (Réf. : Laboratoire 2). Dans cette version améliorée, la date passée en argument peut avoir le format *jj-mm-aaaa* ou *aaaa-mm-jj*.

Exemples :
```
extraitJSJJMMAAAAv2($intJS, $intJ, $intM, $intA);
extraitJSJJMMAAAAv2($intJS, $intJ, $intM, $intA, "01-03-2018");
extraitJSJJMMAAAAv2($intJS, $intJ, $intM, $intA, "2018-12-31");
```

7. dateValide

Prototype : **`function dateValide($strDate)`**

But : Retourne **`true`** si la date passée en argument est valide; autrement **`false`**. La date peut être saisie avec le format *jj-mm-aaaa* ou *aaaa-mm-jj*.

Contrainte : Vous devez utiliser les fonctions `extraitJSJJMMAAAAv2()` et `checkdate()`.

Exemples : `dateValide("29-02-2016")` et `dateValide("2016-02-29")` retournent **`true`**
`dateValide("29-02-2018")` et `dateValide("2018-02-29")` retournent **`false`**

8. dateEnLitteral

Prototype : **`function dateEnLitteral()`**

But : Retourne la date passée en paramètre sous forme littérale.

Contrainte : Vous devez utiliser les fonctions créées préalablement. Comme indiqué précédemment, rien ne sert de créer une librairie de fonctions si vous ne l'utilisez pas !

Scénarios :	Sortie attendue :
`dateEnLitteral()`	Date du jour sans le jour de la semaine
`dateEnLitteral("date")`	Date passée en argument sans le jour de la semaine
`dateEnLitteral("C")`	Date du jour avec le jour de la semaine
`dateEnLitteral("date", "C")`	Date passée en argument avec le jour de la semaine
`dateEnLitteral("C", "date")`	Date passée en argument avec le jour de la semaine

Remarques :

- La "**`date`**" peut être en format "`jj-mm-aaaa`" ou "`aaaa-mm-jj`"
- Minuscule et majuscule traitées indifféremment

Exemples (En supposant que la date du jour est le 31 janvier 2018)

`/* Date SANS le jour de la semaine */`	
`dateEnLitteral()`	**31 janvier 2018**
`dateEnLitteral("01-01-2019")`	**1er janvier 2019**
`dateEnLitteral("2019-01-01")`	**1er janvier 2019**
`/* Date AVEC le jour de la semaine */`	
`dateEnLitteral("C")`	**Mercredi 31 janvier 2018**
`dateEnLitteral("c")`	**Mercredi 31 janvier 2018**
`dateEnLitteral("31-12-2018", "C")`	**Lundi 31 décembre 2018**
`dateEnLitteral("2019-02-05", "c")`	**Mardi 5 février 2019**
`dateEnLitteral("C", "06-03-2019")`	**Mercredi 6 mars 2019**
`dateEnLitteral("c", "2019-04-11")`	**Jeudi 11 avril 2019**

9. AAAAMMJJ

Prototype : **`function AAAAMMJJ()`**

Scénarios : `function AAAAMMJJ($strDate) où $strDate = "jj-mm-aa[aa]"`
`function AAAAMMJJ($intJour, $intMois, $intAnnee)`

But : Retourne la date en format *aaaa-mm-jj*. (Inspirez-vous de la fonction `JJMMAAAA`).

Exemples :
```
AAAAMMJJ("31-12-2018")
AAAAMMJJ("31-12-18")
AAAAMMJJ(31, 12, 2018)
AAAAMMJJ(31, 12, 18)    retournent 2018-12-31
```

a. get

Prototype : **`function get($strNomParametre)`**

But : Retourne la valeur du « paramètre » **`$strParametre`**; autrement la valeur **`null`**.

Exemples : `application.php?tbNom=Legendre&tbPrenom=Pierre`

`get("tbNom")` retourne « **`Legendre`** »
`get("tbprenom")` retourne « **`null`** »

b. post

Prototype : **`function post($strNomParametre)`**

But : Retourne la valeur du « paramètre » **`$strParametre`**; autrement la valeur **`null`**.

c. input

Prototype : **`function input($strID, $strCLASS, $strMAXLENGTH, $strVALUE, $binECHO=false)`**

But : Genère une balise `<INPUT type="text">`.

Exemples :

```
$strDateDuJour = aujourdhui();
echo input("tbDateDuJour", "sDate", "10", $strDateDuJour);
echo input("tbDateDuJour", "sDate", "10", $strDateDuJour, false);
input("tbDateDuJour", "sDate", "10", $strDateDuJour, true);
```

génère la balise…

```
<input id="tbDateDuJour" name="tbDateDuJour" class="sDate"
   type="text" maxlength="10" value="2018-02-06" />
```

d. annee, mois, jour (ajouts de dernière minute !)

Prototypes :
```
function annee($strDate)
function mois($strDate)
function jour($strDate)
```

But : Retourne respectivement l'année, le mois et le jour (sous forme d'entiers) de la date passée en paramètre selon le format *aaaa-mm-jj* ou *jj-mm-aaaa*.

Exemples :
```
annee("2018-02-09")   retourne 2018
mois("2018-02-09")    retourne 2
jour("2018-02-09")    retourne 9

annee("31-12-2019")   retourne 2019
mois("31-12-2019")    retourne 12
jour("31-12-2019")    retourne 31
```