Laboratoire 06

Les fonctions

6.1) Écrivez une fonction **`Accumulator()`** qui renvoie la somme de cinq nombres entrés par vous et passés en paramètres individuellement dans la fonction (ne PAS utiliser de tableaux pour ce numéro).

N.B. Votre fonction `accumulator()` peut être appelée dans une fonction nommée `E01()`

```
La somme est de 15
```

6.2) Écrire une fonction **`Sum()`** ayant comme paramètres un tableau de nombre et qui retourne la somme de tous les éléments du tableau. Le tableau contiendra 5 nombres.

N.B. Votre fonction peut être appelée dans une fonction nommée `E02()`

```
Entrer nombre #1 : 1.25
Entrer nombre #2 : 50
Entrer nombre #3 : 5.75
Entrer nombre #4 : 3
Entrer nombre #5 : 1
La somme de ces 5 nombres est de 61
```

6.3) Écrire une fonction **`Distance()`** ayant comme paramètres 4 nombres `x1`, `y1` et `x2`, `y2` qui représentent les coordonnées de deux points 1 et 2 et qui renvoie la *distance entre les points*.

N.B. Votre fonction peut être appelée dans une fonction nommée `E03()`

```
Entrer x1 : 1
Entrer y1 : 1
Entrer x2 : 5
Entrer y2 : 5
La distance entre les points est de 5.66
```

6.4) Écrire une fonction **`RandomNumbers()`** ayant en paramètre une quantité de nombre à entrer et qui retourne un tableau remplis de nombre aléatoires en 0 et 100 qui sera ensuite affichés à l'écran à partir de la fonction `E04();`. Il faut que l'algorithme ne génère l'objet `Random()` qu'une seule fois en mémoire .

Défi : Permettre à l'utilisateur de remplacer le nombre aléatoire maximum par le nombre de son choix. Cela doit également fonctionner s'il n'entre absolument rien. Indice : Lire sur les *try… catch* !

N.B. Votre fonction peut être appelée dans une fonction nommée `E04()`

```
Entrer une quantité de nombre : 10
Mes nombres sont : 10 94 59 82 0 9 3 84 96 12
```

6.5) Écrivez une fonction **`Power()`** permettant de calculer la puissance de n'importe quel nombre. Exemple 2 à la 4, 10 à la 5. Vous devez bien sur tout coder vous-même, pas le droit d'utiliser de fonctions C#. L'algorithme se termine quand l'utilisateur ne veut plus faire de calcul (`N`).

N.B. Votre fonction peut être appelée dans une fonction nommée `E05()`

```
Entrer un calcul (O/N)?
O
Entrer nombre : 2
Entrer puissance : 4
Résultat = 16
Entrer un calcul (O/N)?
oui
Entrer nombre : 5
Entrer puissance : 3
Résultat = 125
Entrer un calcul (O/N)?
P
S
Entrer un calcul (O/N)?
N
```

6.6) Reprenez l'exercice précédent et, si ce n'est pas déjà le cas, transformer la demande de faire d'autres calculs en fonction **`AnotherPower()`** Qui retourne une valeur booléenne (0 ou 1).

```
//Voir résultat précédent
```

Les fonctions avancées

6.7) Lisez sur les fonctions récursives et créer un programme en C# avec une fonction **récursive** permettant de calculer la factorielle d'un nombre demandé à l'utilisateur.

```
Entrez un nombre entier : 6
La factorielle de 6 est 720
```

6.8) Écrire une fonction **`Prime()`** ayant en paramètre un nombre et qui renvoie une valeur booléenne si le nombre entré est un *nombre premier*. L'algorithme se termine lorsque l'utilisateur entre un zéro (0). Aucune des fonctions C# ne seront acceptées, vous devez bien sur coder l'algorithme du nombre premier.

```
Entrer nombre : 10
10 est un nombre divisible
Entrer nombre : 7
7 est un nombre premier
Entrer nombre : 0
```

6.9) Écrire un programme C# qui demande un nombre x et affiche les x premiers nombre de cette série [1, 1, 2, 3, 5, 8, 13, ...] (suite de Fibonacci) en mode **récursif**.

Double-défi, écrire proprement la fonction récursive en 2 ligne de code !

```
Entrer la position de Fibonacci : 22
Nombre #9 de la suite de Fibonacci = 17711
```

6.10) Inventer un algorithme qui nécessite l'utilisation d'une fonction qui fait appel à une autre fonction, toutes deux développées par vous (pas de fonctions C#).