Exercice 1 : chiffres et nombres

Rapellons que $a\%b$ est le reste de la division entière de a par b.

1. Ecrire la fonction `int unites(int n)` retournant le chiffre des unités du nombre n.
2. Ecrire la fonction `int dizaines(int n)` retournant le chiffre des dizaines du nombre n.
3. Ecrire la fonction `int extrait(int n, int p)` retourant le p-ème chiffre de représentation décimale de n en partant des unités.
4. Ecrire la fonction `int nbChiffres(int n)` retournant le nombre de chiffres que comporte la représentation décimale de n.
5. Ecrire la fonction `int sommeChiffres(int n)` retournant la somme des chiffres de n.
6. Ecrire la fonction `void convertir(int n,char s[])` convertissant le nombre n en chaîne de caractères.

Exercice 2 : Nombre d'occurences

Ecrivez une fonction `int nb_occur(char chaine[], char lettre_cherchee)` qui renvoie le nombre d'occurences de la lettre `lettre_cherchee` dans le tableau de caractères `chaine`. Testez votre fonction sur la chaine suivante :

```
chn[13]={'a','b','r','a','c','a','d','a','b','r','a','\0'};
```

Exercice 3 : Première occurence

Ecrivez une fonction qui prend en paramètre une chaine de caractères et qui ne garde dans la chaîne que la première occurence de chaque lettre. Par exemple pour la chaine précédente, le résultat sera la chaîne :

```
{'a','b','r','c','d','\0','\0','\0','\0','\0','\0','\0','\0'}
```

Exercice 4 : Calcul de puissance

Créer un programme qui renvoie le nombre x^n. Dans un premier temps, on suppose que x et n sont donnés "en dur" dans la fonction `main`.

1. Créer une fonction entière `power` qui calcule itérativement x^n par n multiplications successives
2. Modifier la fonction `power` pour calculer récursivement x^n en utilisant récursivement la propriété que :

$$\begin{cases} x^1 &= x \\ x^n &= \left(x^{\frac{n}{2}}\right)^2 & \text{si } n \text{ est pair} \\ x^n &= \left(x^{\frac{n-1}{2}}\right)^2 * x & \text{si } n \text{ est impair.} \end{cases}$$

3. Comparer les deux solutions en utilisant la commande `time`.

Exercice 5 : Fonction `main`

Ecrire, compiler et exécuter un programme pour réaliser une commande `calculer` dont le comportement est le le suivant :

```
$>./calculer -sum 1 2 3 4
10
$>./calculer -prod 1 2 3 4
24
$>./calculer -abc 1 2 3
Commande non valide
Usage : ./calculer -[sum|prod] n1 n2 ...
$>./calculer -prod
Commande non valide
Usage : ./calculer -[sum|prod] n1 n2 ...
```