Exercice 1:

Écrire un programme en C qui lit un ensemble de lettres minuscules et génère tous les sous-ensembles possibles, puis les affiche.

- L'ensemble en entrée est lu caractère par caractère, commençant par '{', suivie de lettres minuscules consécutives à partir de 'a' et séparées par des virgules, se terminant par '}'.
- Pour simplifier, les lettres sont supposées se suivre de manière consécutive.
- L'utilisation de tableaux n'est pas autorisée, mais les opérateurs bit à bit peuvent être utilisés pour mémoriser et manipuler les données.
- Nous supposons que l'utilisateur respecte bien le format en entrée

Exemple :
Si l'utilisateur entre l'ensemble `{a, b, c}`, le programme devra afficher les sous-ensembles suivants (l'ordre d'affichage des sous-ensembles n'est pas pertinent) :

Sous-ensembles
{}
{a}
{b}
{c}
{a, b}
{a, c}
{b, c}
{a, b, c}

Remarque : Penser à décomposer votre programme (et votre problème).

Exercice 2:

En utilisant les opérations bit à bit, écrire une fonction récursive en C qui prend en paramètre un entier positif *nb* et retourne le nombre de bits égaux à 1 dans sa représentation binaire. Par exemple, pour l'entier passé en paramètre 14, qui s'écrit 1110 en binaire, la fonction doit retourner 3.

Exercice 3:

- Dites dans quel ordre les instructions suivantes, associées à une boucle for, sont exécutées :

```
#include <stdio.h>
int main (void)
{....
    for (instructions 1; instructions 2; instructions 3)
    {
    instuctions 4
    }
    return(0);
}
```

- Est-il possible de modifier le programme ci-dessous afin de confirmer l'ordre d'exécution des instructions de la boucle "for".

```
#include <stdio.h>
int main (void)
{
  int i;
    for (i=0; i!=5; i++)
    {
```

```
        printf ("La valeur de i est %d :\n", i);
        }
        return(0);
}
```

Indications:

- Le caractère "," (virgule) peut-être utilisé pour écrire une suite d'instructions (qui sera terminée par le ";").
- Les trois champs peuvent contenir toute suite d'instructions.
- La suite d'instructions, donnée dans le champs "instructions 2", peut elle aussi être composée de plusieurs instructions élémentaires séparées par une virgule ",". Dans ce cas là, c'est l'évaluation de la dernière instruction élémentaire qui sera utilisée pour déterminer la condition d'arrêt de la boucle "for".

Exercice 4:

Ecrire une fonction C qui prend en paramètre un entier positif n et retourne le nombre de diviseurs stricts (inférieurs à n) du nombre n. Par exemple,

- si le nombre n est égal à 12, alors la fonction retourne 5 (car le nombre 12 admet 5 diviseurs stricts 1, 2, 3, 4, 6).

Exercice 5:

1. Ecrire une fonction itérative (non récursive) qui calcule a^b où a et b sont des entiers.
2. Ecrire une fonction récursive qui calcule a^b où a et b sont des entiers en utilisant les propriété suivantes :
   - Si $b = 0$ alors $a^b = 1$
   - Si b est pair alors $a^b = a^{(b/2)} * a^{(b/2)}$
   - Si b est impair alors $a^b = a * a^{(b-1)}$

Exercice 6:

La suite, notée F(n), des nombres de Fibonacci est définie par les équations de récurrences suivantes :

$$\begin{array}{lcl} \text{F(0)} & = & 1 \\ \text{F(1)} & = & 1 \\ \text{F(n)} & = & \text{F(n-1) + F(n-2)} \quad \text{pour n} > 2 \end{array}$$

- Écrire une fonction récursive, appelé fib_recursive(n), qui calcule $F(n)$.
- Dans le F(n) on réalise que les F(i), avec i<n, sont calculés plusieurs fois. En réalité, il suffit de connaître deux valeurs consécutives de la suite pour calculer la suivante. Écrire une fonction itérative, appelée fib_iterative(n), qui calcule efficacement F(n).
- (trop difficile) Proposer une fonction, appelée fib_efficace(n), plus efficace que les deux précédentes, pour le calcul de F(n).