Exercice 1:

Ecrire une fonction qui prend en paramètre un entier positif n et retourne 1 si le nombre n est uniforme. La fonction retourne 0 sinon. Un nombre est dit uniforme s'il est une répétition d'un même chiffre. Par exemple les nombres 111, 33333, 7777 sont des nombres uniformes alors que les nombres 123, 3332, 8879179 ne le sont pas.

Exercice 2:

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 3:

- Déclarer une constante k (par exemple k est égale à 5) grâce à la directive #define.
- Écrire une fonction, appelée fusion, qui prend en paramètres deux tableaux (de même taille k) triés (de manière croissante) et fusionne ces deux tableaux dans un troisième tableau passé aussi en paramètre. Le troisième tableau résultat (de taille 2xk) doit-être également trié.
- Tester votre fonction depuis le programme principal main.

Exercice 4:

Écrire un programme en C qui :

- lit un entier n,
- lit n entiers positifs ou nuls, puis les stocke en mémoire grâce à une allocation dynamique,
- affiche d'abord le nombre d'entiers nuls,
- affiche ensuite les nombres pairs,
- et enfin, affiche les nombres impairs.

Exercice 5:

Écrire une fonction qui prend en paramètre un entier positif a et un indice n, et qui inverse le n-ième bit de cet entier (change 0 en 1 ou 1 en 0). L'indice n est compris entre 0 et (sizeof(a)*8)-1. La fonction doit retourner le nouvel entier après l'inversion du bit.

Exemple :

Si l'entier est $a = 10$ (en binaire 1010) et que l'on souhaite inverser le 2ème bit (en partant de 0), alors la fonction retournera 14 (en binaire 1110).

Exercice 6 :

Les types de base (int, long int) ne permettent pas de représenter de très grands nombres. Dans cet exercice, on suppose qu'un grand nombre est codé par un tableau de taille TAILLE où chaque case contient un entier de 5 chiffres. La case d'indice (TAILLE-1) contient les 5 chiffres de poids faibles. Par exemple, le nombre suivant :

$$12345154524542545245425425425454524542542542$$

sera codé par le tableau suivant (avec TAILLE=10) :

valeur	00000	01234	51545	24542	54524	54254	25425	45452	45425	42542
indice	0	1	2	3	4	5	6	7	8	9

- Ecrire une fonction qui pend en entrée deux grands nombres A et B de taille TAILLE et retourne 1 si A est strictement plus grand B et retourne 0 sinon.

- Ecrire une fonction qui
  - prend en paramètres trois grands nombres (e.g., tableaux) A, B et C de taille TAILLE, et
  - calcule l'opération de la soustraction $A - B$ (c'est-à-dire calcule le résultat de la soustraction de deux tableaux où chaque élément contient un nombre de 5 chiffres).
  - Le résultat est stocké dans le tableau C. On suppose que A est plus que grand que B.
- Tester votre fonction depuis la fonction principale main.

Exercice 7:

Le but de cet exercice est d'illustrer qu'un problème peut avoir des solutions différentes

- Ecrire une fonction C qui prend en paramètre un entier positif b et retourne un nombre positif a tel que b=a*a (on suppose qu'un tel nombre a existe). Les variables a et b sont déclarés comme unsigned long int.
  Tester votre programme avec b=64 et b=766680721.
- Ecrire une fonction C qui prend en paramètre deux nombres flottant x et y et retourne 1 si $x-\epsilon \leq y \leq x+\epsilon$. La fonction retourne 0 sinon. La constante ϵ est définie avec la directive #define.
- Ecrire une fonction C qui prend en paramètre un entier réel positif b et retourne un nombre positif a tel que $(a*a)-\epsilon \leq b \leq (a*a)+\epsilon$ (on suppose qu'un tel nombre a existe). Dans cette question, les variables a et b sont déclaré comme des "double" ou "float".
  Tester votre programme avec b=98.71 et $\epsilon = 0.001$.

Exercice 8:

Le but de cet exercice est de réaliser le jeu de Mastermind. Voici le principe du jeu :

- L'ordinateur choisit une combinaison ordonnée de n couleurs différentes (typiquement, n est égal à 4), stockée dans des cases numérotées de "0" à "n-1".
- Le joueur tente de deviner cette combinaison.
- A chaque fois que le joueur propose une combinaison ordonnée de n couleurs l'ordinateur indique :
  - le nombre de cases bien placées,
  - le nombre de cases mal placées, et
- Le jeu s'arrête lorsque la combinaison a été trouvée ou bien le nombre d'essai maximal (typiquement 10) est dépassé.

Nous supposons que :

- Chaque couleur est représentée par un caractère :
  b : blanc, j : jaune, r : rouge, v : vert, n : noir et g : gris.
  Aucune autre couleur n'est autorisée.
- On utilise les tableaux pour enregistrer une combinaison des couleurs.
- Le tableau, appelé secret; contient la combinaison choisie par l'ordinateur.
- Le tableau, appelé joueur, contient une combinaison proposée par le joueur.

On vous demande :

- de déclarer les deux constantes n et *max_ essais* grâce à la directive #define.
- d'écrire une fonction afficher (qui n'admet pas paramètres et ne retourne pas de valeurs) qui affiche le message suivant :

```
**********************************************************
**          Bienvenu au jeu de Mastermind               **
**          Les caractères autorisés sont :             **
**          b : blanc, j : jaune, r : rouge,            **
**          v : vert, n : noir et g : gris.             **
**     Chaque combinaison doit avoir 4 caractères       **
**     Exemple : rbjg                                   **
**     Vous avez droit à 10 essais.                     **
**          Bon courage                                 **
**********************************************************
```

- d'écrire une fonction, appelée case_exacte, qui :
  - prend en paramètres
    - deux tableaux de taille n, et
    - un entier, appelé *indice*
  - retourne la valeur 1 si les deux tableaux partage la même valeur de la case *indice* et retourne la valeur 0 sinon.
- d'écrire une fonction, appelée existe_mais_malplacee, qui :
  - prend en paramètres
    - Le tableau secret,
    - un caractère c (de type unsigned char) et
    - un entier, appelé *indice*
  - retourne la valeur 1 si le caractère c existe dans une position différente de *indice* et retourne la valeur 0 sinon.
- d'écrire une fonction, appelée combinaison_valide, qui :
  - prend en paramètres
    - Le tableau tab de n caractères
  - retourne la valeur 1 si la combinaison, représentée par tab, est valide et retourne la valeur 0 sinon. Une combinaison est dite valide si elle ne contient pas de doublons et elle ne contient que des caractères (ou couleurs) autorisés.
- d'écrire un programme C qui réalise le jeu de Mastermind.