Exercice 1:

Compléter la ligne 8 du programme suivant (en remplissant les ....). Ce programme convertit une lettre majuscule (resp. minuscule) en une lettre minuscule (resp. majuscule). On suppose que l'utilisateur saisit soit une lettre majuscule soit une lettre minuscule.

```
1  #include <stdio.h>
2  int main (void)
3  {
4      char car;
5      printf ("\nMerci d'introduire une lettre majuscule ou minuscule: ");
6      scanf ("%c", &car);
7      printf ("\nla conversion (majuscule vs minuscule) de %c
8        donne %c.\n ", ..., ... + (....?  'a' -'A' : ...));
9      return(0);
10 }
```

Exercice 2:

1. Ecrire un programme C

   - qui lit un entier positif n et
   - qui affiche la somme des chiffres qui le composent.

   Par exemple, si $n = 2095$ le programme affichera le nombre 16. On rappelle que :

   - $a\%b$: donne le reste de la division de a par b.
   - a / b : donne le résultat de la division entière de a par b lorsque a et b sont des variables de type entier.

2. Considérons l'énigme suivant : Le 24 février 2014, Vincent a fêté son anniversaire. En présence de ses amis réunis pour l'occasion, il a fait la remarque suivante :

   > Cette année, mon âge est égal à la somme des chiffres de mon année de naissance.

   Ecrire un programme C qui calcule l'âge (ainsi que l'année de naissance) possible de Vincent en 2014. Il est demandé d'afficher toutes les solutions possibles de l'énigme.

Exercice 3:

- Écrire un programme en langage C qui lit un entier positif `n` et affiche sa forme binaire en utilisant uniquement des opérations bit à bit.

  Quelques remarques :

  – Dans un premier temps, il est demandé d'afficher la représentation binaire de `n` de manière inversée. Par exemple, si l'utilisateur entre `14`, sa représentation binaire est `1110`, mais le programme doit afficher `0111`. Il n'est pas demandé d'afficher les zéros non significatifs.

  – Si l'utilisateur ne saisit pas un entier valide, le programme doit afficher un message d'erreur et s'arrêter en renvoyant un code d'erreur `return 1`.

  – On gardera en tête que `n` est un entier non signé. Chaque fois qu'une constante entière est utilisée dans une expression impliquant la variable `n` (par exemple `0` ou `1`), elle devra être écrite avec le suffixe `u` pour indiquer qu'il s'agit d'une constante de type `unsigned int`.

  – L'affichage des bits doit se faire exclusivement avec l'instruction `putchar()`. L'utilisation de `printf()` n'est pas autorisée pour l'impression des bits (sauf éventuellement pour le message d'erreur ou pour la demande de saisie d'un entier positif). Une fois tous les bits affichés, le programme doit ajouter un retour à la ligne également via `putchar`.

- Reprendre l'exercice en procédant cette fois-ci à un affichage normal. Si l'utilisateur entre `14`, le programme doit afficher `1110`. Pour cela, nous vous encourageons à écrire une fonction qui affiche la longueur d'un entier positif en terme de bits significatifs (c'est-à-dire sans compter les zéros non significatifs). Par exemple, si l'utilisateur entre `14`, la fonction retournera la valeur 4.
- Reprendre l'exercice en affichant cette fois tous les zéros non significatifs.

Exercice 4:

- 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 5:

Dans cet exercice, on suppose que nous avons n individus, identifiés simplement par les nombres $\{0, ..., n-1\}$. On s'intéresse à determiner s'il existe un agent secret parmi ces individus. Un agent secret est défini comme quelqu'un qui n'est connu par personne (sauf par lui même bien sûr) mais qui connaît tout le monde. Supposons que vous disposez d'une fonction booléenne, appelée connaît(i,j), qui retourne vraie si la personne i connaît la personne j. Supposons qu'il existe au plus un agent secret parmi les n individus.

- Ecrire une fonction (avec deux boucles imbriquées) qui retourne l'identifiant (le numéro) de l'individu agent secret (s'il existe).
- Proposer une version efficace de votre fonction en n'utilisant que des boucles simples (non-imbriquées).