Exercice 1:

Compiler et exécuter le programme suivant :

```
/****
Mon premier programme C
 ***/

#include <stdio.h>
int main (void)
{
  printf ("Voici mon premier programme C. \n");
  printf ("Il est réalisé pendant les séances de TP. \n");
  return(0);
}
```

Solution 1:

Le programme affiche :

```
Voici mon premier programme C.
Il est réalisé pendant les séances de TP.
```

Exercice 2:

Le but de cet exercice est d'écrire la séquence d'instructions qui permet d'afficher le message suivant (la réalisation du jeu proprement dit se fera dans un TP ultérieur) :

```
**********************************************************
**          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                                 **
**********************************************************
```

Solution 2:

```
#include <stdio.h>
int main (void)
{
    printf ("**********************************************************\n");
    printf ("**          Bienvenu au jeu de Mastermind               **\n");
    printf ("**          Les caractères autorisés sont :             **\n");
    printf ("**          b : blanc, j : jaune, r : rouge,            **\n");
    printf ("**          v : vert, n : noir et g : gris.             **\n");
    printf ("**     Chaque combinaison doit avoir 4  caractères      **\n");
    printf ("**     Exemple : rbjg                                   **\n");
    printf ("**     Vous avez droit à 10 essais.                     **\n");
    printf ("**          Bon courage                                 **\n");
    printf ("**********************************************************\n\n\n\n\n");
  return(0);
}
```

Exercice 3:

Compiler et exécuter le programme suivant :

```
#include <stdio.h>
int main (void)
{
    int a, b;
    a=16;
    b=016;
    printf ("Pourquoi la variable a (=%d) a une valeur différente de
            la variable b (=%d) ? \n", a, b);
    return(0);
}
```

Commentez le résultat de l'exécution de ce programme.

Solution 3:

L'exécution de ce programme donne :

Pourquoi la variable a (=16) a une valeur différente de la variable b (=14) ?

En effet, lors de l'affectation la variable b=016 est précédé d'un "0" qui signifie que la constante "16" est écrite en octal, ce qui donne 14 en décimal.

Exercice 4:

- Ecrire un programme C qui lit un entier (de type unsigned short int), qui représente un code ASCII en décimal, et affiche le caractère associé,
  - Si l'utilisateur rentre le nombre 90, votre programme affichera le caractère 'Z'.
- Ecrire un programme C qui lit un caractère (de type char) et affiche le code ASCII associé en décimal, en octal et en hexadécimal (minuscule et majuscule).
  - Si l'utilisateur rentre le caractère 'Y', votre programme affichera :
    Le code ASCII associé au caractère Y est : 89 (en décimal), 131 (en octal), 59 (en hexadécimal minuscule) et 59 (en hexadécimal majuscule).

Solution 4:

```
1  #include <stdio.h>
2  int main (void)
3  {
4      unsigned short j;
5      unsigned char c;
6      printf ("Merci d'introduire un code ASCII : ");
7      scanf ("%hu", &j);
8      printf ("Le caractère associé au code ASCII %hu est : %c.", j, j);
9      while(getchar() !='\n');
10     printf ("\nMerci d'introduire maintenant un caractère :");
11     c=getchar();
12     printf ("\nLe code ASCII associé au caractère %c est : %d (en décimal), %o (en octal),
13         %x (en hexadécimal minuscule) et %X (en hexadécimal majuscule). \n", c, c, c, c, c);
14     return(0);
15 }
```

Exercice 5:

Ecrire un programme C qui, sans utiliser les codes ASCII associés aux caractères,

- lit un caractère minuscule et
- le transforme en un caractère majuscule.

Nous rappelons que :

- les codes ASCII des lettres majuscules (respectivement minuscules) se suivent.
- Par exemple,
  - le code ASCII de la lettre 'B' est égal à celui de la lettre 'A' plus 1.
  - Le code ASCII de la lettre 'C' est égal à celui de la lettre 'B' plus 1 et
  - ainsi de suite.

Solution 5:

Brève réponse:

```
1  #include <stdio.h>
2  int main (void)
3  {
4      char c;
5      printf ("Merci d'introduire un caractère miniscule : ");
6      c=getchar();
7      /* ou scanf ("%c", &c);
8       (c-'a') donne la position de c par rapport à 'a' */
9      printf ("Le caractère majuscule associé est %c \n", (c-'a')+'A');
10       return(0);
11 }
```

Solution détaillée:

Les deux principes utilisés pour traiter cet exercice sont :

- Les codes ASCII des lettres majuscules (respectivement minuscules) se suivent. Par exemple, le code ASCII de la lettre 'B' est égal à celui de la lettre 'A' plus 1. Le code ASCII de la lettre 'C' est égal à celui de la lettre 'B' plus 1 et ainsi de suite.
- Maintenant, si une lettre minuscule c se trouve à une certaine distance x de la lettre 'a' alors la lettre majuscule associée à c se trouve à la même distance de la lettre 'A'. Cette distance x est simplement le résultat de l'opération c-'a'. Et pour trouver la lettre majuscule associée il suffit de rajouter cette distance à la lettre 'A'.

Plus précisément,

```
1  #include <stdio.h>
2  int main (void)
3  {
4      char c;
5      printf ("Merci d'introduire un caractère miniscule : ");
6      c=getchar();
7      // ou scanf ("%c", &c);
8      // (c-'a') donne la position de c par rapport à 'a'
9      printf ("Le caractère majuscule associé est %c \n", (c-'a')+'A');
10     while(((c = getchar()) !='\n') && (c != EOF));
11     printf ("Merci d'introduire un caractère majuscule : ");
12     c=getchar();
13     printf ("Le caractère miniscule associé est %c \n", (c-'A')+'a');
14     return(0);
15 }
```

Exercice 6:

- Exécuter le programme suivant :

```
#include <stdio.h>
int main (void)
{
unsigned char  i='A';
int j=1;
printf ("\n \t i=%d \t \"j=%d\" \n", i, j);
printf ("\n \t i=%c \t \" \n", i);
return(0);
}
```

  - Un étudiant, au moment de la saisie du programme ci-dessous, s'est rendu compte que la touche \ ne fonctionne pas. Proposer une ré-écriture du programme ci-dessous sans utiliser le caractère \.
  - **Indication :** Utiliser les codes ASCII des caractères retour à la ligne, etc.

Solution 6:

- L'exécution du programme donne :

$$i = 65"j = 1"$$

$$i = A$$

- Le même programme sans utiliser le caractère \ :

```
1
2  #include <stdio.h>
3  int main (void)
4  {
5  unsigned char  i='A';
6  int j=1;
7  printf ("%c %c i=%d %c %cj=%d%c %c", 10, 9, i, 9, 34, j,34, 10);
8  printf ("%c %c i=%c  %c", 10, 9, i, 10);
9  return(0);
10 }
```

- Ce programme est obtenu en utilisant les codes ASCII des différents caractères d'échappement "\t",\n", etc

- **Remarque** : \n est considéré comme un seul caractère, avec un code ASCII égale à 10. Par contre, le caractère \ seul a le nombre 92 comme code ASCII. De ce fait :

  - Les deux impressions suivantes sont équivalentes :

```
#include <stdio.h>
int main (void)
{
    printf ("\n Impression 1 avec deux caractères de retour à la ligne \n");
    printf ("%c Impression 2 avec les codes ASCII de retour à la ligne %c", 10, 10);
    return(0);
}
```

    et le programme affiche :

> >
> *Impression 1 avec deux caractères de retour à la ligne*
> >
> *Impression 2 avec les codes ASCII de retour à la ligne*
> >

– Mais ces deux impressions diffèrent de l'impression suivante :

```
#include <stdio.h>
int main (void)
{
    printf ("%cn Impression 3 avec les codes ASCII du caractère backslash %cn", 92, 92);
    return(0);
}
```

et le programme affiche :

> *\n Impression 3 avec les codes ASCII du caractère backslash \n*

Exercice 7:

Grâce aux différentes options de la fonction printf, un étudiant a réalisé les affichages suivants du nombre d'or (≈1.61803398875) :

```
Ecriture no.  1   :  1.618034
Ecriture no.  2   :   1.618034
Ecriture no.  3   :  01.618034
Ecriture no.  4   :    1.618034
Ecriture no.  5   :  001.618034
Ecriture no.  6   :     1.618034
Ecriture no.  7   :  0001.618034
Ecriture no.  8   :        1.618
Ecriture no.  9   :  0000001.618
```

Ecrire un programme C qui reproduit l'affichage ci-dessus (sachez que l'écriture 1 a été obtenue en utilisant uniquement le format "%f").

Solution 7:

```
#include <stdio.h>
int main (void) {
    float n_or = 1.61803398875;
    int i = 1;
    printf ("Ecriture no. %2d \t :  %f \n", i++, n_or);
    printf ("Ecriture no. %2d \t :  %9f \n", i++, n_or);
    printf ("Ecriture no. %2d \t :  %09f \n", i++, n_or);
    printf ("Ecriture no. %2d \t :  %10f \n", i++, n_or);
    printf ("Ecriture no. %2d \t :  %010f \n", i++, n_or);
    printf ("Ecriture no. %2d \t :  %11f \n", i++, n_or);
    printf ("Ecriture no. %2d \t :  %011f \n", i++, n_or);
    printf ("Ecriture no. %2d \t :  %11.3f \n", i++, n_or);
    printf ("Ecriture no. %2d \t :  %011.3f \n", i++, n_or);
    return(0);
}
```

Exercice 8:

Sachant que le code ASCII de '\a' (bip) est 7 et que le code ASCII de '0' est 48, dites qu'affiche le programme C suivant :

```
#include <stdio.h>
int main (void)
{
    printf("Impression 1 : %c. \n", 7);
    printf("Impression 2 : %c. \n", '7');
    printf("Impression 3 : %d. \n", 7);
    printf("Impression 4 : %d. \n", '7');
    return(0);
}
```

Solution 8:

Le programme affiche :

Impression 1 : . (un bruit sonore)
Impression 2 : 7.
Impression 3 : 7.
Impression 4 : 55.