Remarque : Les deux derniers exercices sont des exercices bonus.

Exercice 1:

Ecrire une fonction C qui :

- lit un nombre entier positif et
- affiche les deux derniers chiffres du nombre lu.

Exercice 2:

Cet exercice illustre le comportement de la fonction scanf lorsque la donnée lue ne correspond pas au format attendu.

Ecrire un programme C qui :

- lit un premier entier et l'affiche, puis
- lit un deuxième entier et l'affiche, et
- lit un dernier entier et l'affiche.

Tester votre programme avec les valeurs :

- 3, 5 et 19.
- A, 12 et 14.

Nous verrons plus tard des conséquences plus importantes et proposerons des solutions.

Exercice 3:

Considérons le programme suivant :

```
#include <stdio.h>
int main (void)
{
   printf ("\n Impression avec deux caractères de retour à la ligne \n");
   return(0);
}
```

Un étudiant, n'ayant pas la touche \, a re-codé ce programme par le suivant :

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

- A votre avis, pourquoi l'étudiant a choisi le code ASCII 92 ?
- Est-ce que son programme fonctionne correctement ? Justifier votre réponse.

Exercice 4:

La bibliothèque <limits.h> contient une liste de constantes qui donnent les domaines des différents types de variables entières. Parmi ces constantes, on trouve :

CHAR_BIT	Nombre de bits mot
SCHAR_MIN	Valeur minimale pour signed char.
SCHAR_MAX	Valeur maximale pour signed char.
UCHAR_MAX	Valeur maximale pour unsigned char.
CHAR_MIN	Valeur minimale pour char.
CHAR_MAX	Valeur maximale pour char.
SHRT_MIN	Valeur minimale pour short int.
SHRT_MAX	Valeur maximale pour short int.
USHRT_MAX	Valeur maximale pour unsigned short int.
INT_MIN	Valeur minimale pour int.
INT_MAX	Valeur maximale pour int.
UINT_MAX	Valeur maximale pour unsigned int.
LONG_MIN	Valeur minimale pour long int.
LONG_MAX	Valeur maximale pour long int.
ULONG_MAX	Valeur maximale pour unsigned long int.

Ecrire un programme C qui affiche la valeur de ces constantes.

Exercice 5:

Le but de cet exercice est d'expliquer que l'emplacement de l'opérateur "cast" est important. Exécuter les programmes suivants :

1.
```
#include <stdio.h>
int main (void)
{
  int i=5, j=2;
  float f;
  f=4*(i/j);
  printf("La valeur de f est : %.f. \n", f);
  return(0);
}
```

2.
```
#include <stdio.h>
int main (void)
{
  int i=5, j=2;
  float f;
  f=4*(float)(i/j);
  printf("La valeur de f, après une application globale de l'opérateur cast, est : %f. \n", f);
  return(0);
}
```

3.
```
#include <stdio.h>
int main (void)
{
  int i=5, j=2;
  float f;
  f=4*((float)i/(float)j);
  printf("La valeur de f, après des applications locaux de l'opérateur cast, est : %f. \n", f);
  return(0);
}
```

Exercice 6:

Grâce à la fonction pré-définie "sizeof", écrire un programme C qui affiche la taille des différents types de variables vus en cours (utiliser aussi les séquences "\t" et "\n" pour avoir un bon affichage).

Exercice 7:

Compiler et exécuter le programme suivant :

```
1 #include <stdio.h>
2 int main (void)
3 {
4   unsigned char  i;
5   i=250;
```

```
6    i=i+6;
7    printf("La valeur de i est %d \n", i);
8    return(0);
9  }
```

Expliquer l'affichage obtenu.

Exercice 8:

Le but de cet exercice est d'illustrer qu'un nombre réel n'a qu'une représentation approchée en langage C (et en ordinateur).

- On vous demande d'écrire un programme C qui :
  - lit un nombre réel positif,
  - le stocke dans une variable de type float,
  - l'imprime sous forme standard mais avec une précision de 10 chiffres après la virgule, et
  - multiple le nombre réel lu par 10 et affiche le résultat obtenu avec une précision de 10 chiffres après la virgule.
- Reprendre les questions précédentes mais avec une variable de type double et une précision de 15 chiffres après la virgule.
- Tester vos programmes avec plusieurs valeurs réelles : 0.1, 0.212, 0.212121, etc

Exercice 9:

Ecrire un programme C qui déclare une variable i de type entier et affiche son adresse.

Exercice 10:

Soit *var* une variable de type signed char, codée sur 8bits.

- Donner le domaine de valeurs associé à *var*.
- Supposons que l'on affecte à *var* la valeur -200. Quelle sera la valeur finale associée à *var*.
- Même question si l'on affecte à *var* la valeur 188.

Ecrire un programme C qui confirme vos résultats.

Exercice 11:

Un étudiant a écrit le programme C ci-dessous. Il s'est rendu compte qu'il avait fait 8 erreurs. Proposer une correction de ces erreurs. Compiler et tester votre programme.

```
/****
 Les huit erreurs de compilation
 ***/
#include <stdia.h>
int main (void)
{
  int b==1,c;
  const int f=0;
  printf ("Merci de saisir un premier nombre. \n");
  scanf ("%c", &b);
  printf ("Merci de saisir un deuxieme nombre. \n");
  scanf ("%d", c);
  a=++(b+c);
  f=(a>1);
  if (f)
    printf ("La somme des deux nombres lus est strictement positive. \n);
  return(0)
}
```

Exercice 12:

Soit i une variable de type "unsigned int".

- Compléter le tableau suivant (toujours avec i=14):

Opérations bit à bit	valeur de i en binaire	valeur de i en décimal
i	0000 1110	14
$i << 1$	0001 1100	28
$i << 4$	1110 0000	224
$i >> 1$	0000 0111	7
$i >> 4$	0000 0000	0
i&1	1110 et 0001	0
i&4	1110 et 0100	4
$1 << 1$	0000 0010	2
$1 << 4$	0001 0000	16

- Compléter le tableau suivant, en remplaçant cette fois ci les opérations bit à bit par une suite d'instructions arithmétiques :

Opérations bit à bit	Expression arithmétique
$i << 1$	i*2^1
$i << 4$	i*2^4
$i >> 1$	i*2^-1
$i >> 4$	i*2^-4
i&1	i%2
i&4	(i%2) % i
$1 << 1$	1*2^1
$1 << 4$	1*2^4

- Ecrire un programme C qui confirme vos réponses.

Exercice 13:

Considérons l'énigme suivant :

"Le décor de la quatrième énigme, publiée par "Sud Ouest" dans le cadre de la Semaine des mathématiques, est un immeuble de *11* étages, dont l'ascenseur est étrange : il ne peut monter que deux, trois ou cinq étages à la fois et ne peut descendre que quatre ou onze étages. Le concierge, dont la loge est située au rez-de-chaussée, doit procéder à la distribution du courrier. Comment doit-il opérer pour partir de sa loge, s'arrêter une fois et une seule à chaque étage, revenir chez lui? "

On vous demande d'écrire un programme C qui permet de trouver toutes les solutions.

Exercice 14:

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).