Licence Informatique — **UE INF101**
Licence Mathématiques — **Initiation la programmation**
Université Bordeaux 1 — **Travaux Dirigés**
2006-2007 — **feuille 2**

Exercice 1. Écrire une fonction `int factorielle(int n)` qui calcule et retourne `n!`.

Exercice 2. On considère la suite définie par $u_0 = 1$ et $\forall n > 0$ $u_n = 3 * u_{n-1}^2 + 2$.
Écrire une fonction `int suite(int n)` qui calcule et retourne le terme u_n de cette suite.

Exercice 3.
Écrire une fonction `void affiche_carrés_parfaits(int n)` qui affiche tous les nombres entiers inférieurs à `n!` qui sont des carrés parfaits.

Exercice 4. Écrire une fonction `int somme_chiffres(int n)` qui retourne la somme des chiffres du nombre entier `n`.
Rappels : Dans une base donnée `b`
- le reste de la division de n par `b` est égal à la valeur du dernier chiffre de n (celui situé le plus à droite).
- le quotient de la division entière de n par `b` est égal à la valeur repésentée par le nombre `n` privé de son dernier chiffre

Exercice 5.

1. Écrire une fonction qui retourne la somme des chiffres de rang impair d'un nombre `n` passé en paramètre (le chiffre le plus à droite est considéré comme étant de rang 0).
   exemple : la valeur retournée pour le nombre 1254687998 est 9 + 7 + 6 + 5 + 1 = 28
2. Un nombre entier `n` est divisible par 11 si la somme de ses chiffres de rang impair est congrue à la somme de ses chiffres de rang pair modulo 11.
   Utiliser cette propriété pour écrire une fonction booléenne `bool multiple_de_onze(int n)` qui retourne `true` si n est un multiple de 11 et false sinon.

Exercice 6. Quelle est la valeur de l'expression `n / x > x && x != 0` dans les cas suivants :
- n = 10.0, x= 5.0
- n= 10.0, x = 3.0
- n = -10.0, x = -3
- n = 10, x = 0

Corriger cette expression afin qu'elle ne produise jamais d'erreur.

Exercice 7. Écrire une fonction `double moyenne()` qui calcule et retourne la moyenne des entiers lus à l'aide de la fonction `lire_entier`. La saisie 0 signifiera la fin de la saisie.

Exercice 8. Quelle est la valeur de l'expression `x%2 == 0 || x < 10` dans les cas suivants :
- `x` vaut 5 ?
- `x` vaut 12 ?
- `x` vaut 23 ?

Exercice 9. Ecrire une fonction `table_mult` qui affiche une table de multiplication de taille n. Par exemple pour n = 6, on obtient l'affichage :

1	2	3	4	5	6
2	4	6	8	10	12
3	6	9	12	15	18
4	8	12	16	20	24
5	10	15	20	25	30
6	12	18	24	30	36

Exercice 10. Suite de Fibonacci

On considère la suite définie par :

$$\begin{cases} u_0 = u_1 = 1 \\ u_n = u_{n-1} + u_{n-2} \end{cases}$$

Écrire une fonction `int fibonacci(int n)` qui calcule et retourne le terme u_n de cette suite.

Exercice Complémentaire 1. Suite de Syracuse

On considère la suite définie pour une valeur u_1 positive fixée par :

$$\begin{cases} u_n = 3u_{n-1} + 1 \ si \ u_{n-1} \ est \ impair \\ u_n = \frac{u_{n-1}}{2} \ si \ u_{n-1} \ pair \end{cases}$$

1. Écrire une fonction une fonction `suivant` qui, étant donné en paramètre un terme de la suite, calcule et retourne le terme suivant de cette suite.
2. Écrire une fonction `syracuse1` qui calcule pour une valeur de u_1 le énième terme de la suite.
3. Ecrire une fonction `syracuse2` qui affiche pour une valeur u_1 donnée les n premiers termes de la suite.
4. La conjecture suivante est appelée conjecture de Syracuse :
   La suite définie ci-dessus converge vers le cycle (1 ;4 ;2)
   Cette conjecture n'a jamais été démontrée, mais on a montré par le calcul que ce résultat était vrai pour tout u_1 inférieur à $5.5 * 10^{14}$.

   Ecrire une fonction `syracuse3` qui pour une valeur u_1 affiche la liste des valeurs prises par la suite entre u_1 et 1 (bornes comprises).
   Ecrire une fonction `max_syracuse` qui pour une valeur u_1 calcule le maximum des valeurs prises par la suite entre u_1 et 1 (bornes comprises).