LIFAPI – TP4 : Fonctions et Procédures … suite

1. Ecrivez une **fonction** `saisie_positive` qui saisit et retourne une valeur strictement positive. On recommencera la saisie tant que la valeur proposée n'est pas strictement positive.

2. Ecrivez un sous-programme qui affiche le triangle de Pascal jusqu'à la ligne n en effectuant les étapes suivantes :
   a. Ecrivez la fonction `int combinaison(int n, int p)` (qui utilise la fonction *factorielle* que vous devrez recopier ou réécrire).

      Rappel : $C_n^p = \frac{n!}{p!(n-p)!}$

   b. Testez ce premier sous-programme avec différentes valeurs. Que se passe-t-il si p>n ?
   c. Ecrivez la procédure `void trianglePascal(int n)` qui utilise la fonction combinaison pour afficher le triangle de rang n.

      Exemple : trianglePascal(5); ➔
      ```
      1
      1  1
      1  2  1
      1  3  3  1
      1  4  6  4  1
      1  5 10 10  5  1
      ```
   d. Ecrivez le programme principal qui teste les différents sous-programmes. On utilisera la fonction `saisie_positive` écrite dans l'exercice 1 pour s'assurer que la hauteur donnée est bien strictement positive.

3. Écrivez un sous-programme `renverse` qui demande un entier `n` et affiche `n` en inversant l'ordre des chiffres. Par exemple, si l'utilisateur fournit n=123, le programme affichera n=321. On ne demande pas ici de stocker le résultat dans une nouvelle variable mais juste d'afficher.

Indication : n modulo 10 (*n%10* en C) donne le dernier chiffre (celui de droite) ; la partie entière de n/10 donne tous les chiffres de gauche (sauf le dernier). Il suffit alors d'itérer. En C, si vous divisez deux entiers entre eux, vous obtenez la partie entière de la division.

Par exemple :

```
int n,i;

n=125;

i = n/10; /* i prend pour valeur 12 */
```

4. En mathématiques, un nombre est dit **triparfait** si la somme de ses diviseurs (y compris lui-même) est égale à trois fois ce nombre.
   Exemple : 120 a pour diviseurs : 1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 20, 24, 30, 40, 60, 120.
   La somme de ces diviseurs est 360.
   Or, 360 ÷ 3 = 120 → donc 120 est **triparfait**.
   a. Écrivez un sous-programme `somme_diviseur` qui calcule et retourne la somme des diviseur d'un nombre n strictement positif passé en paramètre.
   b. Écrivez une fonction booléenne `estTriparfait` qui retourne `true` si `n` est triparfait, `false` sinon.
   c. Écrivez le programme principal qui affiche tous les nombres triparfait entre 1 et 100000.