Algorithmique et programmation

1. Ecrire un algorithme **Test** qui permet de lire deux entiers x et y et qui affiche -1 si x < y, 0 si x = y et 1 si x > y.
2. Les années **bissextiles** sont divisibles par 4, sauf celles divisibles par 100, avec toutefois une exception pour celles qui sont divisibles par 400. Par exemple :
   — 2012 est une année bissextile car 2012 est divisible par 4 ;
   — 2010 n'est pas une année bissextile puisque 2010 n'est pas divisible par 4 ;
   — 2100 n'est pas bissextile puisque divisible par 100 ;
   — 2000 une année bissextile puisque divisible par 400.
   Ecrire un algorithme algorithme **annee_bissextile** qui teste si une année est bissextile.
3. Ecrire un algorithme **Arrondi_pair** qui prmet de lire un nombre entier n et qui affiche le nombre pair immédiatement inférieur ou égal à n. Par exemple, si n=5 l'algorithme affiche 4, si n = 6 l'algorithme affiche 6.
4. Ecrire un algorithme **Seuil** qui prmet de lire un entier M et qui affiche le plus petit entier naturel n tel que n! > M.
5. Ecrire un algorithme **Divisible** qui prmet de lire un entier n et affiche 'divisible' si n! est divisible par n + 1 et 'nonDivisible' sinon.
6. Ecrire un algorithme **Premier** qui prmet de lire un entier n et affiche un message qui indique si cet entier est premier ou non.
7. Ecrire un algorithme **Premiers** qui affiche et calcule la somme de tout les nombres premiers dans un intervalle [A,B] (A et B lus par le clavier).
8. Ecrire un algorithme **Chiffre** qui prmet de lire un entier n et qui calcule son chiffre des unités (qui est donc compris entre 0 et 9).
   Même question avec le chiffre des centaines.
9. Un nombre naturel est **parfait** s'il est égal à la somme de ses diviseurs propres (c'est à dire tous les nombres entiers qui le divisent sauf lui-même).
   Exemple : 9 n'est pas parfait car 9 ≠ 1+3 mais 6 est parfait car 6 = 1 + 2 + 3.
   Ecrire un algorithme **Nbrparfait** qui teste si un nombre est parfait.
10. Un nombre d'**Armstrong** est un entier qui est égal à la somme des cubes de ces chiffres.
    Exp : 153 est un nombre d'Armstrong car $1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$.
    Ecrire un algorithme **NbrArmstrong** qui permet de vérifier si un entier N est un nombre d'Armstrong ou non.
11. Ecrire un algorithme **Division_euclidienne** qui prmet de lire deux entiers n ≥ 0 et d > 0 et qui affiche le quotient et le reste de la division euclidienne de n par d.
12. Ecrire un algorithme **PPCM** qui affiche le PPCM de deux entiers n et m.
13. Ecrire un algorithme **PGCD** qui affiche le PGCD de deux entiers n et m.
14. **Approximation de π par la formule de Leibniz**
    On peut obtenir des valeurs approchées de π en mettant en œuvre la formule (1) :
    $$\frac{\pi}{4} = \sum_{i=0}^{\infty} \frac{(-1)^i}{2i+1} = \frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \frac{1}{11} \cdots \quad (1)$$
    Ecrire un algorithme **Leibniz** qui permet de calculer et afficher une valeur approchée de π.
    Remarque :
    Répéter les calculs jusqu'à la précision voulue $\left(\left|\frac{\pi_{i+1}}{4} - \frac{\pi_i}{4}\right| < 0.00001, \text{avec } i \in \mathbb{N}\right)$.
15. Ecrire un algorithme **BIN** qui demande un entier n puis affiche son code **binaire**.