Algorithmique et programmation
« Les fonctions »

Exercice N° 1 :

1. Ecrivez une fonction « signe(x) » ayant un seul argument qui renvoie 1 si cet argument est strictement positif, -1 s'il est strictement négatif, 0 s'il est nul.
2. A l'aide de de la fonction « **signe(x) »**, écrivez une fonction valAbs(x) qui renvoie la valeur absolue de son argument.
3. Ecrire la fonction « **pgcd(a,b) »** qui calcule et retourne le PGCD de a et b.
4. Ecrire la fonction « **ppmc(a, b)** » qui retourne le plus petit multiple commun de a et b.
5. Ecrire un programme qui demande à l'utilisateur de saisir deux nombres et qui calcule et affiche leur pgcd et ppmc.

Exercice N° 2 :

1. Écrire une fonction « **f(x)** » qui retourne f (x) = $2x^3 + x - 5$.
2. Écrire une procédure « **tabuler(borneInf, borneSup, pas) »** avec 3 paramètres : **borneInf**, **borneSup** et pas. Cette procédure affiche les valeurs de la fonction f, de **borneInf** à **borneSup**, avec un avancement de valeur **pas**.
3. Tester cette procédure par un appel dans le programme principal après avoir saisi les deux bornes.

Exercice N° 3 :

Un nombre entier est parfait s'il est égal à la somme de ses diviseurs (sauf lui-même).
Ex : 6 = 1 + 2 + 3 est parfait.

1. Ecrire une fonction « **def somme_div(n)** » qui retourne la somme de tous les diviseurs d'un nombre passé en paramètre.
2. Ecrire une fonction « **def parfait (m)** qui teste si un nombre passé en paramètre **m** est parfait et qui retourne **True** s'il l'est et **False** sinon.
3. Ecrire une fonction « **def premier (x)** qui teste si un nombre passé en paramètre **x** est premier et qui retourne **True** s'il l'est et **False** sinon.

Deux nombres M et N sont appelés nombres amis si la somme des diviseurs strictes de M est égale à N et la somme des diviseurs strctes de N est égale à M.

4. Ecrire une fonction « **def amis(x, y) »** qui renvoie **True** si x et y sont amis ou **False** si non.
5. Vérifier si 28 et 496 sont des nombres parfaits et que 220 et 284 sont amis

Exercice N° 4 :

1. Ecrire une fonction **fact(n)** qui calcule et retourne la factorielle de n.
2. Utiliser la fonction **fact** pour écrire une fonction **Comb(n,p)** qui renvoie les combinaisons de p dans n.
3. Ecrire une procédure **pascale(nl)** qui affiche un triangle de **nl** lignes.
4. Afficher un triangle de pascal de 10 lignes

Exercice N° 5 :

Un nombre narcissique (ou nombre d'Armstrong de première espèce) est un entier naturel n non nul qui est égal à la somme des puissances p-ièmes de ses chiffres en base dix, où p désigne le nombre de chiffres de n :

Exemples de nombres narcissiques :
$153 = 1^3 + 5^3 + 3^3$
$548834 = 5^6 + 4^6 + 8^6 + 8^6 + 3^6 + 4^6$

1. Ecrire en une fonction **nbre_chiffres(n)** qui prend en argument un nombre entier naturel n et renvoie **le nombre de chiffre de n**.
   Exemple : **nbre_chiffres(153) renvoie 3**
2. Ecrire en une fonction **est_armstrong(n)** qui prend en argument un nombre entier naturel n et renvoie **True** si n est un nombre d'Armstrong ou **False** si non.

Exercice N° 6 :

1. Ecrire une fonction **conversionHMS(heure, minute, seconde)** qui donne le nombre de secondes équivalente à un horaire fourni avec son heure, ses minutes, et ses secondes.
2. On suppose implémenter une fonction « **extrait(chaine, numeroDebut, numeroFin) »** qui renvoie l'extrait de la chaine, commençant au caractère à la position **numeroDebut** et finissant au caractère à la position **numeroFin**.
   Ecrire une fonction « **conversionChaine(horaire)** » qui donne le nombre de secondes équivalente à un horaire fourni en paramètre. L'horaire sera une chaine de caractères respectant le format "HH:MM:SS" où HH est le nombre d'heures sur 2 caractères, MM de minutes et SS de secondes. Aucune vérification du format ne sera prévue.
3. Un radar-tronçon mesure la vitesse moyenne d'un véhicule entre deux points de contrôle (le tronçon) séparés d'une certaine distance, en mémorisant l'horaire H0 auquel le véhicule a débuté le tronçon et l'horaire H1 auquel il l'a fini.
   Ecrire une fonction « **radar(H0, H1, distance, limite)** » qui prend en paramètres les deux horaires H0 et H1, la distance du tronçon et la limitation de vitesse (donnée en km/h) sur le tronçon. La fonction renverra un booléen qui indique par True si le véhicule est en excès de vitesse et False si le véhicule ne l'est pas. Les horaires H0 et H1 auront le format d'une chaine de caractère "HH:MM:SS"'.
4. Un véhicule est parti du péage autoroutier de Casablanca-Lissasfa à 11h30 et a atteint le péage autoroutier d'El Jadida à 12h00 ; la distance séparant les deux péages est de 92 km. Ecrire un programme qui affiche à l'écran si l'utilisateur est en excès de vitesse ou non.

Exercice N° 7 :

1. Ecrire une fonction **de( )** sans paramètre qui demande à l'utilisateur de choisir un nombre entier entre 1 et 6 (en vérifiant la saisie) et renvoie la valeur saisie.
2. Ecrire un programme qui calcule la somme de deux saisies de l'utilisateur et affiche le résultat à l'écran.
3. L'utilisateur souhaite savoir si le dé est pipé ou non : il doit donc vérifier que les fréquences d'apparition de chaque face sont égales à 1/6. Pour rappel, pour la face 1, la fréquence d'apparition du nombre 1 peut se calculer en tirant N fois le dé et en comptant le nombre de 1 **n1** obtenu à l'issu de ces N ; elle vaut alors n1/N. Une marge d'erreur de 5% entre la valeur estimée et la valeur théorique 1/6 sera considérée comme acceptable. Ecrire un programme qui demande à l'utilisateur (que l'on supposera suffisamment motivé pour ne pas se décourager devant l'ampleur des saisies) les résultats de lancé de dé permettant d'estimer les fréquences d'apparition de chaque face à l'aide de N = 1000 lancés puis analyse si le dé est pipé ou non, en affichant le résultat à l'écran.