Université de Mila
Institut des Sciences et de la Technologie
Département de Mathématique et Informatique
1ère Année LMD MI

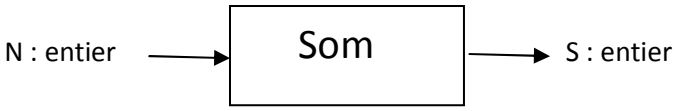
SÉRIE DE TD N°3 (LA RÉCURSIVITÉ) SOLUTIONS

Module : Algorithmique et structures de données 2 Année universitaire : 2019/2020

Exercice 1

1) Calculer la somme : Som(N)= 1+2+3+4+5+6+……..+(N-1)+N

Etape 1 : on va chercher l'opération qui se répète, dans notre cas c'est l'addition (+).
Etape 2 : on décompose le problème en plusieurs sous problèmes de même type mais de taille inférieure. Som (N) = Som (N-1)+ N
Som (N-1) = Som (N-2)+ N-1 …
Etape 3 : on distingue les cas possibles : dans cette question : cas **N=1** et cas **N≠1**
Donc : si N= 1 alors Som (N) = 1
Sinon Som (N) = Som (N-1) + N
Puisque on a un seul résultat, on va utiliser une fonction récursive comme suit :



Rôle : calculer la somme de 1 à N

```
Fonction Som (n : entier) : entier ;
   S: entier ;
Début
    Si  (n = 1)  alors
S ← 1 ;
Sinon
S ← Som (n-1) + n; // l'appel récursive
Finsi
Retourner (S) ;
Fin ;
```

2) Calculer la puissance X^N : puis (X, N) **(avec les mêmes étapes)**

```
    Fonction puis (X, N : entier) : entier ;
    P: entier ;
    Début
        Si  (N = 1 )  alors
    P ← X ;
    Sinon
    P ← puis  (X, N - 1) *X ; // l'appel récursive
    Finsi
     Retourner (S) ;
    Fin ;
```

Remarque importante : on a juste écrit la fonction récursive, l'algorithme n'est pas complet, il faut compléter l'algorithme par l'entête et les déclarations et le programme principal.

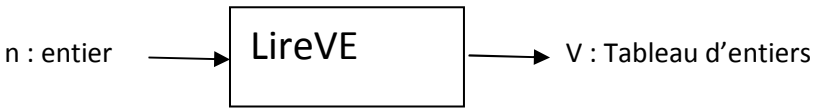
Exercice 2

```
Algorithme Exo3
       Nbr1, Nbr2 : entier ;

// Cette fonction récursive calcule le PGCD de A et B par
//l'algorithme d'Euclide, à condition que A > B.

Fonction PGCD  (A, B : entier) : entier ;
Res, R: entier ;
Début
  R ← A mod B ;
  Si  (R = 0)  alors
Res ← B;
Sinon
Res ← PGCD (B, R) ; // l'appel récursive
Finsi

  Retourner (Res) ;

Fin ;

Début

    Lire (Nbr1, Nbr2) ;

Si( Nbr1 > Nbr2) alors

    Ecrire (PGCD (Nbr1, Nbr2)) ;
Sinon // Nbr1 < Nbr2

Ecrire (PGCD (Nbr2, Nbr1)) ;
Finsi ;

Fin .
```

Exercice 3

Remarque importante : si le résultat est un vecteur ou matrice, on ne peut pas utiliser **la fonction**, on utilise toujours **la procédure.**

1) La procédure ***RemplirVE*** permettant de remplir V.



Rôle : Lire un vecteur ***V***de ***n*** éléments entiers

```
Procédure LireVE (n : entier ; Var V []:tableau d'entiers)
   i : entier;
Début
Si (n=1) alors
      Lire (V[1]);
Sinon
      Lire (V[n]) ;
      lireVE(n-1, V) ; // l'appel récursive
finsi
Fin;
```

2) La procédure ***AfficherVE*** permettant d'afficher les valeurs de V. (remplacez lire par écrire)

3) La fonction ***Somme*** qui retourne la somme des éléments du vecteur. (similaire à som de exo 1)
4) La fonction ***Max*** qui retourne le maximum du vecteur.

```
fonction MaxVE(n : entier ; V []:tableau d'entiers ;) : entier ;
max,i : entier;
Début
Si (n=1) alors
 max←V[1] ;
Sinon
 max ←MaxVE(n-1, V) ; // l'appel récursive
 Si (max < V[n]) alors
  max←V[n] ;
 finsi
Finsi
 Retourner (max) ;
Fin;
```

5) La fonction ***Appartient*** permettant de vérifier si un élément existe dans le vecteur ou non.

```
Fonction Appartient(Val,n : entier ;V []:tableau
d'entiers) :booléen ;
R :booléen ;
Début
Si (Val=V[n]) alors
R← vrai;
Sinon
Si (n=0) alors
R← faux;
Sinon
R←Appartient(Val,n-1,V) ; // l'appel récursive
finsi
finsi
Retourner (R) ;
Fin;
```

Exercice 4 : la fonction retourne le rang de l'élément s'il existe sinon elle retourne -1.

```
Fonction Recherche (Val:entier; V[]:tableau d'entiers; Iinf,Isup:entier):
entier ;
Imil : entier ;
Début
Imil ← (Iinf+Isup) div 2 ;
si (Iinf>Isup ou V [Imil]= Val) alors// condition d'arrêt
Si (Iinf>Isup) alors
Retourner(-1); // Val n'existe pas dans V
Sinon
Retourner(Imil);
finsi
Sinon
Si( Val< V [Imil] ) alors
Retourner(Recherche (Val, V, Iinf, Imil-1)); // l'appel récursive
Sinon
Retourner (Recherche (Val, V, Imil+1, Isup)); // l'appel récursive
finsi
finsi
Fin ;
```