Exercice 1

Ecrire un algorithme qui prend de l'utilisateur deux entiers puis affiche le signe du produit sans faire la multiplication et le signe de la somme sans faire l'addition.

Remarque
Le produit de 2 entiers est positif si les 2 entiers ont le meme signe.
La somme x + y >= 0 veut dire x >= -y.

```
Algorithme     SigneAddMult;
Var
   x,y, pos, neg: Entier;
   smul, sadd: Caractere;
Debut
   Ecrire("Donner 2 Entiers x et y");
   Lire(x, y);
   si ((x>= 0) et (y>= 0)) ou ((x <= 0) et (y <= 0))  alors
      smul <--- '+';
   sinon
      smul <--- '-';
   fsi

   si (x >= -y)  alors
       sadd <--- '+';
   sinon
       sadd <--- '-';
   Fsi
   Ecrire("Signe multiplication de x:", x, " et y:", y, " est;", smul) ;
   Ecrire("Signe addition de x:", x, " et y:", y, " est;", sadd) ;
Fin.
```

Exercice 2

Les tarifs d'affranchissement des lettres et cartes postales d'Algérie Poste sont donnés dans le tableau suivant :

Poids en Grs.	Tarif en DA
Jusqu'à 20	25
De 21 à 50	40
De 51 à 100	50
Au dessus de 100	Ajouter 10 pour chaque 100 Grs

Ecrire un algorithme qui lit le poids d'une lettre et affiche le tarif correspondant

```
Algorithme     TarifLettre;
Var
    poids, tarif: Entier;
Debut
   Ecrire("Donner Poids de la Lettre");
   Lire(poids);
   Si (poids <= 0)  alors
        Ecrire("Erreur: Poids doit avoir une Valeur Positive");
   Sinon
        Si (poids <= 20)  alors
             tarif <--- 25;
        Sinon
             Si (poids <= 50)  alors
                  tarif <--- 40;
             Sinon
                  Si (poids <= 100)  alors
                      tarif <--- 50;
                  Sinon
                       tarif <--- 50 + (poids Div 100);
```

```
                Fsi
            Fsi
        Fsi
        Ecrire("Pour une Lettre de poids: ", poids, " le Tarif est de: ", tarif, "DA");
    Fsi
Fin.
```

Exercice 3

Une classe contient 200 places réparties en 10 rangées et 20 colonnes. Chaque élève possède un numéro entre 1 et 200. Lors d'un examen, on place les élèves sur les places selon leur numéro en commençant par la première rangée puis la deuxième et ainsi de suite.

```
Algorithme    RangeColEleve;
Var
  num, rg, col: Entier;
Debut
  Ecrire("Donner Num compris entre 1 et 200");
  Lire(num);
  Si (Num < 1) ou (Num > 200)) alors
     Ecrire("Erreur");
  sinon
     col <--- num Mod 20;
     rg  <--- num Div 20;
     Si (col = 0) alors
       /* si col=0 alors mettre col a 20 c'est le cas ou num=20, 40, 60, 80, ... ,180,200 */
       col <--- 20;
     Sinon
       rg <--- rg + 1;
     Fsi
     Ecrire("Pour Numero:", num, "Rangee:", rg, "et colonne:", col);
  Fsi
Fin.
```

Exercice 4

Ecrire un algorithme permettant d'afficher le type de médaille obtenu par un athlète (Or, Argent, Bronze) ou Pas de médaille suivant sa position dans la compétition.

```
Algorithme    Medaille;
Var
  pos: Entier;
Debut
  Ecrire("Donner la position");
  Lire(pos);
  Cas pos Vaut
     1: Ecrire("Or");
     2: Ecrire("Argent");
     3: Ecrire("Bronze");
     sinon
       Si (pos <= 0) alors
         Ecrire("Erreur");
       Sinon
         Ecrire("Pas de Medaille");
       Fsi
  Fcas
Fin.
```

Exercice 5

1- Affichage du nombre de lettres majuscules et celui de lettres minuscules à partir d'une séquence de caractères se terminant par le caractère '#'.

```
Algorithme     nbrMajMin;
Var
   nbrMaj, nbrMin: Entier;
   car: Caractere;
Debut

   Ecrire("Donner le Premier Caractere");
   Lire(car);

   nbrMaj <--- 0;
   nbrMin <--- 0;

   Tant Que (car <> '#')
   Faire
      Si ( (car >= 'a') et (car <= 'z') ) alors
         nbrMin <--- nbrMin + 1;
      Sinon
         Si ( (car >= 'A') et (car <= 'Z') ) alors
            nbrMaj <--- nbrMaj + 1;
         Fsi
      Fsi
      Ecrire("Donner le Prochain Caractere");
      Lire(car);
   Fait
   Ecrire("Nombre de Lettres Majuscules: ", nbrMaj, " Nombre de Lettres Minuscules : ",
nbrMin);
Fin.
```

2- Recherche du minimum et du maximum dans un ensemble de N nombres réels.

```
Algorithme      minMaxNReels;
Var
   i, N: Entier;
   min, max, Nbr: Reel;
Debut
   Repeter
      Ecrire("Donner N Entier > 0");
      Lire(N);
   Jusqu'a (N > 0);

   Ecrire("Donner le Premier Nombre Reel");
   Lire(Nbr);

   /* Au debut on suppose que le 1er nombre est lui le maximum et lui aussi le minimum.
      Ensuite on commence les comparisons a partir du 2eme nombre.
    */
   max <--- Nbr;
   min <--- Nbr;

   Pour i <--- 2 a N
   Faire
      Ecrire("Donner ", i, " eme Nombre Reel");
      Lire(Nbr);
      Si (Nbr > max) alors
         max <--- Nbr;
      Sinon
         Si (Nbr < min) alors
            min <--- Nbr;
         Fsi
      Fsi
```

```
  Fait
  Ecrire("Maximum des ", N, " Nombres Reels : ", max, " et minimum : ", min);
Fin.
```

3- Calcul du quotient et reste de la division de deux entiers A et B sans utiliser l'opération de division.

```
Algorithme      quotRestAParB;
Var
  A, B, Q, R: Entier;
Debut
  Repeter
     Ecrire("Donner 2 Entiers A et B avec A >= 0 et B > 0");
     Lire(A, B);
  Jusqu'a ((A >= 0) ET (B > 0));

  Q <--- 0;
  R <--- A;

  /* Chercher le quotient et le reste de la division euclidienne de A par B revient a faire des
     soustractions successives jusqu'a avoir un reste < B.
   */
  Tant Que (R >= B)
  Faire
     R <--- R - B;
     Q <--- Q + 1;
  Fait
  Ecrire("Quotient de la division de ", A, " par ", B, " : ", Q, " et le Reste :", R);
Fin.
```

4- Vérification si un entier positif X est premier ou non.

```
Solution avec utilisation d'un Booleen
Algorithme      NombrePremier;
Var
  i, X: Entier;
  premier : Booleen;
Debut
  Repeter
     Ecrire("Donner X Entier > 0");
     Lire(X);
  Jusqu'a (X > 0);

  Si (X = 1) alors
    /* 1 est un cas partculier: 1 n'est pas premier */
    premier <--- Faux;
  Sinon
    premier <--- Vrai;

   /* On cherche si un des nombres 2, 3, 4, ... jusqu'a la motier de X divise X.
      Des qu'on trouve un diviseur on s'arrete en mettant le booleen premier a Faux.
    */
    i <--- 2;
    Tant Que ((i <= (X Div 2)) ET (premier = Vrai))
    Faire
       Si ((X mod i) = 0) alors
          premier <--- Faux;
       Sinon
          /* Essayer le prochain i apres avoir incrementer i */
          i <--- i + 1;
```

```
        Fsi
      Fait
    Fsi
    Si (premier = Vrai) alors
        Ecrire(X, " est un nombre premier");
    Sinon
        Ecrire(X, " N'est PAS un nombre premier");
    Fsi
Fin.
```

Solution sans utilisation de Booleen

```
Algorithme     NombrePremier;
Var
   i, X: Entier;
Debut
   Repeter
      Ecrire("Donner X Entier > 0");
      Lire(X);
   Jusqu'a (X > 0);

   Si (X = 1) alors
      /* 1 est un cas partculier: il n'est pas premier */
      Ecrire("1 N'est pas un nombre premier");
   Sinon

     /* En commencant a partir de I=2, tant qu'on n'a pas trouve de diviseur de X <= (X Div 2)
essayer le prochain i. */
      i <--- 2;
      Tant Que ((i <= (X Div 2)) ET ((X mod i) <> 0))
      Faire
          /* Essayer le prochain i apres avoir incrementer I */
          i <--- i + 1;
      Fait

      /* Si on sort avec avec i <= X Div 2 cela veut dire qu'on est sorti du Tant Que a cause de
la condition
         (X mod i) <> 0) qui est fausse et que dans ce cas on a trouve un diviseur et X n'est pas
premier.
       */
      Si (i <= (X Div 2)) alors
         Ecrire(X, " N'est PAS un nombre premier");
      Sinon
         Ecrire(X, " est un nombre premier");
      Fsi
   Fsi
Fin.
```

6- Calcule le nombre d'occurrences d'un chiffre C (0 ≤ C < 10) dans un entier positif A.

```
Algorithme      nmbrOccChiffre;
Var
   A, C, Q, nbrOcc: Entier;
Debut
   Repeter
      Ecrire("Donner Entier A > 0");
      Lire(A);
   Jusqu'a (A > 0);
   Repeter
      Ecrire("Donner un Chiffre compris entre 0 et 9");
      Lire(C);
   Jusqu'a ((C >= 0)  et (C <= 9));
```

```
  Q <--- A;
  nbrOcc <--- 0;

  /* On obtient chacun des chiffres de X en faisant des divisions successives par 10 jusqu'a
avoir un quotient = 0. Les chiffres sont les restes apres chaque division par 10.
   */
  Tant Que (Q <> 0)
  Faire
     Si ((Q mod 10) = C) alors
        nbrOcc <-- nbrOcc + 1;
     Fsi
     Q <--- Q Div 10;
  Fait
  Ecrire("Le nombre d'occurrences de ', C, " dans ", A, " est :",  nbrOcc);
Fin.
```

Exercice 6 :

Ecrire un algorithme permettant d'afficher le Miroir d'un entier positif A. Exemple : Miroir de 26538 = 83562

```
Algorithme      nombreMiroir;
Var
  A, mir, Q: Entier;
Debut
  Repeter
     Ecrire("Donner Entier A > 0");
     Lire(A);
  Jusqu'a (A > 0);

  Q <--- A;
  mir <--- 0;

  /* On obtient chacun des chiffres de X en faisant des divisions successives par 10 jusqu'a
avoir un quotient = 0. Les chiffres sont les restes apres chaque division par 10.
   */
  Tant Que (Q <> 0)
  Faire
     mir <--- (mir * 10) + (Q mod 10);
     Q <--- Q Div 10;
  Fait
  Ecrire("Le nombre miroir de ", A, " est :", mir);
Fin.
```

Exercice 7 :

Ecrire l'algorithme permettant de déterminer le PGCD de deux nombres entiers A et B.

Methode 1 : soustractions successives.

PGCD (3465 , 1575

A	B	Reste
3465 –	1575	= 1890
1890 -	1575	= 315
1575 –	315	= 1260
1260 –	315	= 945
945 –	315	= 630
630 –	315	= 315
315 –	315	= 0

```
Algorithme     PGCDMethodeSoustraction;
Var
```

```
   A, B, max, min: Entier;
Debut
   Repeter
      Ecrire("Donner 2 Entiers A et B avec A > 0 et B > 0");
      Lire(A, B);
   Jusqu'a ((A > 0)  et (B > 0));

   /* on met dans max le plus grand de A et B et dans min le plus petit de A et B. */
   Si (A > B) alors
      max <--- A;
      min <--- B;
   Sinon
      max <--- B;
      min <--- A;
   Fsi

   reste <--- max - min;
   Tant Que (reste <> 0)
   Faire
      Si (reste >= min) alors
         max <--- reste;
      Sinon
         max <--- min;
         min <--- reste;
      Fsi
      reste <--- max - min;
    Fait
    Ecrire("PGCD de ", A, " et ", B, "est :", max);

Fin.
```

Methode 2 : division euclidienne.

PGCD (7038 , 5474)

A	B	Reste
7038	/ 5474	1564
5474	/ 1564	782
1564	/ 782	0

PGCD (3465, 1575)

3465	/ 1575	315
1575	/ 315	0

```
Algorithme     PGCDMethodeDivision;
Var
   A, B, nbr1, nbr2, reste: Entier;
Debut
   Repeter
      Ecrire("Donner 2 Entiers A et B avec A > 0 et B > 0");
      Lire(A, B);
   Jusqu'a ((A > 0)  et (B > 0));

   nbr1 <--- A;
   nbr2 <--- B;

   reste <--- nbr1 mod nbr2;
   Tant Que (reste <> 0)
   Faire
      nbr1 <--- nbr2;
      nbr2 <--- reste;
      reste <--- nbr1 mod nbr2;
```

```
    Fait
    Ecrire("PGCD de ", A, " et ", B, "est :", nbr2);

Fin.
```

Exercice 8 :

Ecrire l'algorithme permettant de déterminer le PPCM de deux nombres entiers A et B (sans passer par le PGCD).

```
Algorithme     PPCMde2Nombres;
Var
   A, B, max, min, ppcm: Entier;
Debut
   Repeter
      Ecrire("Donner 2 Entiers A et B avec A > 0 et B > 0");
      Lire(A, B);
   Jusqu'a ((A > 0)  et (B > 0));

   /* on met dans max le plus grand de A et B et dans min le plus petit de A et B. */
   Si (A > B) alors
      max <--- A;
      min <--- B;
   Sinon
      max <--- B;
      min <--- A;
   Fsi

   /* On cherche le ppcm de min et max. Idee: On vas initialiser le ppcm au max et on vas
chaque fois ajouter au ppcm la valeur de max jusqu'a avoir ppcm multiple de min.  Par
exemple si max=20 et min=8 on initialise ppcm a 20, 20 n'est pas un multiple de 8 mais des
qu'on ajoute un 20 a 20 on a ppcm  = 40 et 40 est un multiple de 8. Boucler sur max permet
d'avoir moins de boucles.*/

   ppcm <--- max;
   Tant Que (ppcm mod min <> 0)
   Faire
      ppcm <-- ppcm + max;
    Fait
    Ecrire("PPCM de ", A, " et ", B, "est :", ppcm);

Fin.
```

Exercice 9 :

Ecrire un algorithme qui détermine et affiche la $N^{ème}$ valeur de la suite (U_N) sachant que : $U_0 = 0$; $U_1 = 1$; $U_2 = 2$; $\mathbf{U_N = 2U_{N-1} + 3U_{N-3}}$ pour N > 2.

```
Algorithme      suiteUN;
Var
   i, N, UN, UN1, UN2, UN3: Entier;

Debut
   Repeter
      Ecrire("Donner N Entier >= 0");
      Lire(N);
   Jusqu'a (N >= 0);

   Cas   N    Vaut
         0: UN <--- 0;
         1:  UN <--- 1;
         2:  UN <--- 2;
```

```
        Sinon
            UN3 <--- 0;  /* UN3 represente UN-3 */
            UN2 <--- 1;  /* UN2 represente UN-2 */
            UN1 <--- 2;  /* UN1 represente UN-1 */

            Pour i <--- 3 a N
            Faire
                 UN <--- (2 * UN1) + (3 * UN3);
                /* A la prochaine iteration: UN2 devient UN3, UN1 devient UN2 et UN devient
UN1 */
                UN3 <--- UN2;
                UN2 <--- UN1;
                UN1 <--- UN;
             Fait
      FCas
      Ecrire("Pour N = ", N, " UN = ", UN);
Fin.
```

Serie Complementaire

Exercice No 1

Soient trois chiffres A, B et C (0 <=A, B, C <= 9). Ecrire un algorithme qui génère et affiche le plus grand et le plus petit nombre qu'on peut former en combinant A, B et C.

```
Algorithme       minMaxCombinaison3Chiffres;
Var
   A, B, C, C1, C2, C3, min, max: Entier;
Debut
   Repeter
      Ecrire("Donner 3 Chiffres A, B et C");
      Lire(A, B, C);
   Jusqu'a ((A >= 0)  et (A <= 9)) et ((B >= 0)  et (B <= 9)) et ((C >= 0)  et (C <= 9));

   /* Dans ce qui suit on vas mettre dans C1 le plus petit de A, B et C - dans C2 le milieu et
dans
      C3 le plus grand de A, B et C. */
   Si (A < B) alors
        Si (C < A) alors
                /* on C - A - B */
                C1 <-- C; C2 <-- A; C3 <-- B;
         Sinon /* cas ou C >= A alors il faut comparer C avec B */
                 Si (C > B) alors
                         C1 <-- A; C2 <-- B; C3 <-- C;
                 Sinon
                         C1 <-- A; C2 <-- C; C3 <-- B;
                 Fsi
          Fsi
     Sinon  /* cas ou A >= B */
           Si (C < B) alors
                 /* on C - B - A */
                 C1 <-- C; C2 <-- B; C3 <-- A;
            Sinon  /* cas ou C>=B alors il faut comparer C avec A */
                 Si (C > A) alors
                         C1 <-- B; C2 <-- A; C3 <-- C;
                 Sinon
                         C1 <-- B; C2 <-- C; C3 <-- A;
                  Fsi
            Fsi
     Fsi
```

```
    max <-- C3 * 100 + C2 * 10 + C1;
    min <-- C1 * 100 + C2 * 10 + C3;
Fin.
```

Exercice No 2

Ecrire un algorithme en utilisant l'instruction "Cas … Vaut" pour résoudre le problème suivant :

Etant donné l'âge d'un enfant, on veut l'informer de sa catégorie :

- Poussin de 6 à 7 ans
- Pupille de 8 à 9 ans
- Minime de 10 à 11 ans
- Cadet de 12 à 15 ans
- Junior de 16 à 18 ans
- Senior 19 ans et plus.

```
Algorithme     CategorieEnfant;
Var
   age: Entier;

Debut
      Ecrire("Donner age de l'enfant");
      Lire(age);
      Si (age >= 6) alors
          Cas A vaut
                6,7: ecrire("Poussin");
                8,9: ecrire("Pupille");
                10,11: ecrire("Minime");
                 12, 13, 14, 15: ecrire("Cadet");
                 16,17,18: ecrire("Junior");
                 sinon
                       ecrire("Senior");
          Fcas
      sinon
          Ecrire("Erreur: Pas de categorie");
      Fsi
Fin.
```

Exercice No 3

2- Le calcul du produit de deux entiers en utilisant uniquement l'opération d'addition '+'.

```
Algorithme     ProduitParAdditions;
Var
  A, B, P, i: Entier;
Debut
      /* Noter A et B peuvent etre negatifs */
      Ecrire("Donner 2 Entiers A et B");
      Lire(A, B);
      P <-- 0;
      Si (A<>0) et (B<>0) alors
          Si (B > 0) alors
              /* Si A>0 on vas ajouter des nombres > 0 et P sera >0.
                  Si A<0 on vas ajouter des nombres < 0 et P sera <0.
               */
              Pour i <-- 1 a B Faire
                    P <-- P + A;
              Fait
          Sinon  /* cas ou B < 0 */
              /* Si A>0 on vas retrancher des nombres > 0 et P sera <0.
                  Si A<0 on vas retrancher des nombres < 0 et P sera >0.
               */
```

```
            Pour i <-- 1 a (-B) Faire
                P <-- P - A;
            Fait
        Fsi
    Fsi
    Ecrire(" Produit de ", A, " par ", B, " = ", P);
Fin.
```

3- Calcul du nombre d'occurrences des caractères 'E' et 'e' dans une suite de N caractères.

```
Algorithme    nbrOccEe;
Var
  nbrEe, i, N: Entier;
  car: Caractere;
Debut
   Repeter
      Ecrire("Donner N > 0");
      Lire(N);
   Jusqu'a (N > 0);

   nbrEe <-- 0;
   Pour i <-- 1 a N
   Faire
      Ecrire("Donner un Caractere");
      Lire(car);
      Si ( (car = 'E') OU (car = 'e') ) alors
            nbrEe <--- nbrEe + 1;
      Fsi
  Fait
  Ecrire("Nombre d'occurrences de E et e est:",nbrEe);
Fin.
```

4- Détermination si A est divisible par B. Avec A et B des entiers positifs.

```
Algorithme     ADivisibleParB;
Var
  A, B, r: Entier;
Debut
  Repeter
     Ecrire("Donner 2 Entiers A et B avec A > 0 et B > 0");
     Lire(A, B);
  Jusqu'a ((A > 0)  et (B > 0));
  /* On vas faire des soustractions successives de A par B jusqu'a arriver a un reste r. A = q *
B + r
      A est divisible par B si r=0 car dans ce cas A= q * B
      A n'est pas divisible par B si A = q * B + r avec r#0
   */
   r <-- A;
  Tant Que (r >= B)
   Faire
       r <-- r - B;
  Fait
  Si (r = 0) alors
       ecrire(A, " est divisible par ", B);
  sinon
       ecrire(A, " N'est Pas divisible par ", B);
  Fsi
Fin.
```

5- Détermination de tous les diviseurs d'un entier X donné.

```
Algorithme     DiviseursX;
Var
   i, X: Entier;
Debut
   Repeter
      Ecrire("Donner X Entier > 0");
      Lire(X);
   Jusqu'a (X > 0);

   /* Afficher le 1er diviseur de X qui est 1 */
   Ecrire(1);

   /* Parcourir tous les nombres de 2 a X Div 2 et voir s'ils divisent X */
   Pour i <--- 2 a (X Div 2)
   Faire
        Si ((X mod i) = 0) alors
            Ecrire(i);
        Fsi
    Fait
    Si (X <> 1) alors  /* On teste si X <> 1 car on ne veut pas ecrire 2 fois 1 lorsque X = 1 */
       Ecrire(X);
    Fsi

Fin.
```

6- Calcul de la somme des K premiers nombres premiers.
Voici par exemple la liste des 10 premiers nombres premier:
2, 3, 5, 7, 11, 13, 17, 19, 23, 29
L'utilisateur nous donne un K>0
Si K=4 par exemple alors l'algorithme doit calculer la somme des 4 premiers nombres premier: 2 + 3 + 5 + 7
Si K=6 par exemple alors l'algorithme doit calculer la somme des 6 premiers nombres premier: 2 + 3 + 5 + 7 + 11 + 13
Si K=10 par exemple alors l'algorithme doit calculer la somme des 10 premiers nombres premier: 2 + 3 + 5 + 7 + 11 + 13 + 17 + 19 + 23 + 29
Donc il faut trouver les K premiers nombres premier et les ajouter a la somme. Pour cela l'idee est de commencer a partir de 2 (variable nbr) et de voir si nbr est un nombre premier si c'est le cas incrementer i qui compte le nombre de nombres premier, ajouter nbr a SPremK qui est la somme des K premiers nombres premier et incrementer nbr et voir si le prochain nbr est un nombre premier ou non et refaire la meme chose. On s'arretera lorsque i aura meme valeur que K qui veut dire qu'on a trouve les K premiers nombres premier.

```
Algorithme      SommeKPremiers;
Var
   i, K, J, SPremK, nbr: Entier;
   premier : Booleen;
Debut
   Repeter
      Ecrire("Donner K Entier > 0");
      Lire(K);
   Jusqu'a (K > 0);

   SPremK <--- 0;
   i <-- 0;    /* i compte combien de nombres premier on a trouve */
   nbr <-- 2;  /* on commence a partir de 2 et voir si 2 est un nombre premier */

   /* Trouver tous les K premiers nombres premiers et les ajouter a SPremK */
   Tant Que (i < K)
   Faire
```

```
        /* supposer au debut que nbr est premier */
         premier <---  Vrai;
        J <--- 2;
        Tant Que ((J <= (nbr Div 2)) ET (premier = Vrai))
        Faire
            Si ((nbr mod J) = 0) alors
                /* on a trouve un diviseur de nbr mettre le booleen premier a Faux pour sortir de
la boucle */
                 premier <--- Faux;
            Sinon
                /* Essayer le prochain J apres avoir incrementer J */
                J <--- J + 1;
            Fsi
         Fait

         Si (premier = Vrai) alors
              /* nbr est un nombre premier dans ce cas */
              SPremK <--- SPremK + nbr;
              i <-- i + 1; /* on a trouve un nombre premier, ajouter 1 a i qui compte le nombre de
nombres preniers */
          Fsi
          nbr <-- nbr + 1;  /* voir si prochain nbr est premier ou non */
    Fait
     Ecrire("La somme des ", K, " nombres premiers est:", SPremK);
Fin.
```

8- Le calcul de A^N en utilisant seulement l'opérateur de multiplication. (A entier et N naturel).

```
Algorithme     APuissanceN;
Var
  A, N, P, i: Entier;
Debut

      Ecrire("Donner un Entier A");
      Lire(A);
      Repeter
          Ecrire("Donner un Entier Naturel N");
          Lire(N);
      Jusqu'a (N >= 0);

      Si ((A = 0) et (N = 0)) alors
           Ecrire("Erreur: 0 a la puissance 0 N'est Pas definie");
      Sinon
         P <-- 1;
         Pour  i <-- 1 a N
         Faire
             P <-- P * A;
        Fait
        Ecrire(A, " Puissance ", N, " = ", P);
      Fsi
Fin.
```

Exercice 4 :

Ecrire l'algorithme qui affiche les tables de multiplication de 1 à 9 pour toutes les valeurs de 1 à 9.

```
1 x 1 = 1          2 x 1 = 2          …          9 x 1 = 9
1 x 2 = 2          2 x 2 = 4          …          9 x 2 = 18
.                                     ... .
.                                     …  .
.                                     …  .
```

1 x 9 = 9 2 x 9 = 18 … 9 x 9 = 81

```
Algorithme    Table1A9;
Var
   i, J : Entier;
Debut
   Pour i <--- 1 a 9
   Faire
      Pour J <--- 1 a 9
      Faire
         Ecrire(J, " x ", i, " = ", i * J);
      Fait
   Fait
Fin.
```

Exercice 6 :

Ecrire un algorithme qui calcule la somme d'ordre N de Sn définie comme suit en utilisant seulement les opérateurs de base (sans l'utilisation de l'opérateur de puissance).

$$\sum_{i=0}^{n} (-1)^{i+1}/X^{i}$$

```
Algorithme  suiteSN;
Var
   i, N, signe: Entier;
   X, P, SN : Reel;

Debut
   Repeter
       Ecrire("Donner N Entier >= 0");
       Lire(N);
   Jusqu'a (N >= 0);

   Repeter
        Ecrire("Donner Reel X Non Nul");
        Lire(X);
   Jusqu'a (X <> 0);

   SN <--- -1;
   signe <--- 1;
   P <--- 1;
   Pour i <--- 1 a N
   Faire
      P <--- P * X;     /* X puissance i */
      SN <--- SN + (signe / P);
      /* Une fois on fait une addition et par la suite on fait une soustraction. On met pour cela
          -signe dans la variable signe (une fois signe contient 1 une fois contient -1).
        */
      signe <---   -signe;

   Fait
   Ecrire("Pour N = ", N, "et X = ", X, " SN = ", SN);
Fin.
```

Exercice 7 :

Ecrire un algorithme qui détermine et affiche la $N^{ème}$ valeur U_N de la suite de 'FIBONACCI' sachant que $U_1 = 1$; $U_2 = 1$; $U_N = U_{N-1} + U_{N-2}$ pour $N > 2$.

```
Algorithme  suiteFibonacci;
Var
   i, N, UN, UN1, UN2: Entier;

Debut
   Repeter
      Ecrire("Donner N Entier > 0");
      Lire(N);
   Jusqu'a (N > 0);

   Si (N < 2) alors
        UN <---  1;
   Sinon
       UN2 <--- 1;  /* UN2 represente UN-2 */
       UN1 <--- 1;  /* UN1 represente UN-1 */

      Pour i <--- 3 a N
      Faire
           UN <--- UN1 + UN2;
           /* A la prochaine iteration: UN1 devient UN2 et UN devient UN1 */
          UN2 <--- UN1;
          UN1 <--- UN;
      Fait
   Fsi
   Ecrire("Pour N = ", N, " UN = ", UN);
Fin
```