TD1

Exercice 1 :

```
int main() {
   int A = 1;
   int B = 2;
   int C = 3;
   int *P1, *P2;
   P1=&A;
   P2=&C;
   *P1=(*P2)++;
   P1=P2;
   P2=&B;
   *P1-=*P2;
   ++*P2;
   *P1*=*P2;
   A=++*P2**P1;
   P1=&A;
   *P2=*P1/=*P2;
   return 0;
}
```

Complétez le tableau suivant :

	A	B	C	P1	P2
Init	1	2	3	/	/
P1=&A	1	2	3	&A	/
P2=&C	1	2	3	&A	&C
***P1=(*P2)++**	3	2	4	&A	&C
P1=P2	3	2	4	&C	&C
P2=&B	3	2	4	&C	&B
***P1-=*P2**	3	2	2	&C	&B
++*P2	3	3	2	&C	&B
***P1*=*P2**	3	3	6	&C	&B
A=++*P2P1**	24	4	6	&C	&B
P1=&A	24	4	6	&A	&B
***P2=*P1/=*P2**	6	6	6	&A	&B

Exercice 2 :

Soit P un pointeur qui 'pointe' sur un tableau A :

```
int main() {
   int A[] = {12, 23, 34, 45, 56, 67, 78, 89, 90};
   int *P;
   P = A;
   return 0;
}
```

Quelles valeurs ou adresses fournissent ces expressions :

***P+2**	14
***(P+2)**	34
&P+1	rarement utiliser l'adresse d'un pointeur
&A[4]-3	une adresse de la composante A[1]
A+3	une adresse de la composante A[3]
&A[7]-P	valeur indice 7
P+(*P-10)	une adresse
***(P+*(P+8)-A[7])**	23

Exercice 3 :

On considère le programme suivant :

```
#include <stdio.h>
int main() {
   int *ad1, *ad2, *ad3;
   int n = 10, p =20;
   ad1 = &n;
   ad2 = &p;
   *ad1 = 3;
   *ad2 = *ad1 + ad2 + p;
    *ad3 = NULL; => correction de l'erreur
   printf("n = %d, p = %d\n", n, p);
   printf("ad1 = %d, *ad2 = %d\n", *ad1, *ad2);
   printf("adresse de n =  %p\n", &n);
   printf("valeur de ad1 =  %d\n", ad1);
   printf("adresse de ad1 =  %p\n", &ad1);
   *ad3 = ad2; => erreur
   printf("ad2 = %d, *ad3 = %d\n", *ad2, *ad3);
   return 0;
}
```

Ce programme est syntaxiquement correct mais ne marche pas à l'exécution. Pourquoi ?

***ad3** n'est pas initialisé. Attention un pointeur doit être initialisé pour ne pas faire des dégâts. Ici, ***ad3** doit être initialisé à **NULL.**

Exercice 4 :

On dit qu'un nombre est parfait si la somme de ses diviseurs positifs est égale au nombre lui-même. Ecrire un algorithme qui permet de reconnaître si un nombre lu au clavier est parfait.

Lister un certain nombre (en TP).

```
Algorithme Nombre Parfait :
Entier : nb, somme, i ;
Début :
        afficher (« Entrez un nombre entier) ;
        lire(nb) ;
        somme <- 1 ;
        Pour i allant de 1 à nb faire:
                Si nb % i = 0 alors:
                        somme <- somme + i ;
                Fin Si
        Fin Pour
        Si nb = somme alors :
                afficher(« ce nombre est parfait ») ;
        Sinon :
                afficher(« ce nombre n'est pas parfait ») ;
        Fin Si
Fin
```

Exercice 5 :

Ecrire un algorithme qui lit un tableau (construit) unidimensionnel et qui le normalise pour d'autres des raisons d'exploitation. Normaliser ici les valeurs d'un tableau c'est ramener ses valeurs à des valeurs comprises entre 0 et 1.

Algorithme de normalisation d'un tableau ;

```
réel : t1, t2, max ;
entier : taille, i ;
Début :
        afficher(« Entrez une taille : ») ;
        lire(taille) ;
        Pour i allant de 1 à taille faire :
                afficher(« Entrez une valeur pour t1[i] : ») ;
                lire(t1[i]) ;
        Fin Pour
        afficher (« Affichage de t1 : » ) ;
        Pour i allant de 1 à taille faire :
                afficher (t1[i]) ;
        Fin Pour
        max <- t1[0] ;
        Pour i allant de 1 à taille faire :
                Si t1[i] > max alors :
                        max <- t1[i] ;
                Fin Si
        Fin Pour
        afficher(«  Max :  » + max) ;
        Pour i allant de 1 à taille faire :
                t2[i] <- t1[i] / max ;
        Fin Pour
        afficher (« Affichage de t2 : ») ;
        Pour i allant de 1 à taille faire :
                afficher (t2[i]) ;
        Fin Pour
Fin
```

Exercice 6 :

Réaliser l'algorithme :

- Intersection entre 2 images binaires
- Union entre 2 images binaires
- Complément d'une image
- Maximum et minimum de 2 images

Pour chaque algorithme, les images (matrice de petites tailles) :

- Initialisation au clavier
- Traitement
- Affichage

Algorithme intersection :

```
Variables :

Définir :  MAX = 2
Entier mat1, mat2, inter, i, j,

Début :
        Afficher ("Première matrice binaire:");
        Pour i allant de 0 à MAX Faire:
           Pour j allant de 0 à MAX Faire:
                Tant que (mat1[i][j]<0) OU (mat1[i][j]>1) Faire:
                        Afficher("mat1[i][j]");
                        Lire(mat1[i][j]);
                Fin tant que
           Fin pour
        Fin pour

        Afficher ("Deuxième matrice binaire:");
        Pour i allant de 0 à MAX Faire:
           Pour j allant de 0 à MAX Faire:
                Tant que (mat2[i][j]<0) OU (mat2[i][j]>1) Faire:
                        Afficher("mat2[i][j]");
                        Lire(mat2[i][j]);
                Fin tant que
           Fin pour
        Fin pour

        Pour i allant de 0 à 2 Faire:
           Pour j allant de 0 à 2 Faire:
              Si ((mat1[i][j]==mat2[i][j]) ET mat1[i][j]==1) Alors :
                 inter[i][j]<=1;
              Sinon Si ((mat1[i][j]==mat2[i][j]) ET mat1[i][j]==0) Alors :
                inter[i][j] <=0;
              Sinon
                 inter[i][j] <=0;
              Fin si
           Fin pour
        Fin pour

        Afficher("Matrice 1 :");
```

```
        Pour i allant de 0 à MAX Faire:
            Pour j allant de 0 à MAX Faire:
                Afficher("mat1[i][j]");
            Fin pour
            Afficher ("")
        Fin pour

        Afficher("Matrice 2 :");
        Pour i allant de 0 à MAX Faire:
            Pour j allant de 0 à MAX Faire:
                Afficher("mat2[i][j]");
            Fin pour
            Afficher ("")
        Fin pour

        Afficher("Matrice intersection :");
        Pour i allant de 0 à MAX Faire:
            Pour j allant de 0 à MAX Faire:
                Afficher("inter[i][j]");
            Fin pour
            Afficher ("")
        Fin pour
Fin
```

Algorithme union :

```
Variables :

Définir :  MAX = 2
Entier mat1, mat2, inter, i, j,

Début :
        Afficher ("Première matrice binaire:");
        Pour i allant de 0 à MAX Faire:
            Pour j allant de 0 à MAX Faire:
                Tant que (mat1[i][j]<0) OU (mat1[i][j]>1) Faire:
                        Afficher("mat1[i][j]");
                        Lire(mat1[i][j]);
                Fin tant que
            Fin pour
        Fin pour

        Afficher ("Deuxième matrice binaire:");
        Pour i allant de 0 à MAX Faire:
            Pour j allant de 0 à MAX Faire:
                Tant que (mat2[i][j]<0) OU (mat2[i][j]>1) Faire:
                        Afficher("mat2[i][j]");
                        Lire(mat2[i][j]);
                Fin tant que
            Fin pour
        Fin pour

        Pour i allant de 0 à 2 Faire:
            Pour j allant de 0 à 2 Faire:
```

```
            Si ((mat1[i][j] == mat2[i][j]) ET mat1[i][j] == 1) Alors:
               inter[i][j] <= 1;
            Sinon Si ((mat1[i][j]==mat2[i][j]) ET mat1[i][j]==0)
              inter[i][j] <= 0;
            Sinon
               inter[i][j] <= 1;
            Fin si
         Fin pour
      Fin pour

      Afficher("Matrice 1 :");
      Pour i allant de 0 à MAX Faire:
         Pour j allant de 0 à MAX Faire:
            Afficher("mat1[i][j]");
         Fin pour
         Afficher ("")
      Fin pour

      Afficher("Matrice 2 :");
      Pour i allant de 0 à MAX Faire:
         Pour j allant de 0 à MAX Faire:
            Afficher("mat2[i][j]");
         Fin pour
         Afficher ("")
      Fin pour

      Afficher("Matrice union :");
      Pour i allant de 0 à MAX Faire:
         Pour j allant de 0 à MAX Faire:
            Afficher("inter[i][j]");
         Fin pour
         Afficher ("")
      Fin pour
Fin
```

Algorithme complément :

```
Variables :

Définir :  MAX = 2
Entier:  mat[MAX][ MAX], comp[MAX][ MAX], i, j,

Début :
      Afficher("Matrice");
      Pour i allant de 0 à MAX Faire :
         Pour j allant de 0 à MAX Faire :
            Tant que (mat[i][j]<0 OU mat[i][j]>1) Faire :
               Afficher("mat[i][j]);
               Lire(mat[i][j]);
            Fin tant que
         Fin pour
      Fin pour

      Pour i allant de 0 à MAX Faire :
         Pour J allant de 0 à MAX Faire :
            Si (mat[i][j] == 1) Alors :
```

```
                comp[i][j] <= 0;
            Sinon :
                comp[i][j] <= 1;
            Fin si
        Fin pour
    Fin pour

    Afficher("Matrice : ");
    Pour i allant de 0 à MAX Faire :
        Pour j allant de 0 à MAX Faire :
            Afficher("comp[i][j]");
        Fin pour
        Afficher("");
    Fin pour

    Afficher("Complémentaire : ");
    Pour i allant de 0 à MAX Faire :
        Pour j allant de 0 à MAX Faire :
            Afficher("comp[i][j]");
        Fin pour
        Afficher("");
    Fin pour
Fin
```

Algorithme maximum et minimum :

```
Variables :

Définir :  MAX = 2
Entier : mat1[MAX][MAX], mat2[MAX][MAX], max[MAX][MAX], min[MAX][MAX], i, j ;

Début :
    Afficher("Première matrice binaire: ");
    Pour i allant de 0 à MAX Faire:
        Pour j allant de 0 à MAX Faire:
            Afficher("mat1[i][j]");
            Lire(mat1[i][j]);
        Fin pour
    Fin pour

    Afficher("Deuxieme matrice binaire: ");
    Pour i allant de 0 à MAX Faire:
        Pour j allant de 0 à MAX Faire:
            Afficher("mat2[i][j]");
            Lire(mat2[i][j]);
        Fin pour
    Fin pour

    Pour i allant de 0 à MAX Faire:
        Pour j allant de 0 à MAX Faire:
            Si mat1[i][j] <= mat2[i][j] Alors:
                min[i][j] <= mat2[i][j];
            Sinon
                min[i][j] <= mat1[i][j];
            Fin si
        Fin pour
```

```
        Fin pour

        Afficher("Matrice 1: ");
        Pour i allant de 0 à MAX Faire:
            Pour j allant de 0 à MAX Faire:
                Afficher("mat1[i][j]");
            Fin pour
            Afficher("");
        Fin pour

        Afficher("Matrice 2: ");
        Pour i allant de 0 à MAX Faire:
            Pour j allant de 0 à MAX Faire:
                Afficher("mat2[i][j]");
            Fin pour
            Afficher("");
        Fin pour

        Afficher("Min: ");
        Pour i allant de 0 à MAX Faire:
            Pour j allant de 0 à MAX Faire:
                Afficher("min[i][j]");
            Fin pour
            Afficher("");
        Fin pour

        Afficher("Max: ");
        Pour i allant de 0 à MAX Faire:
            Pour j allant de 0 à MAX Faire:
                Afficher("Max[i][j]");
            Fin pour
            Afficher("");
        Fin pour
Fin
```