TP1

Exercice 1 :

Soient deux tableaux T1 et T2 de taille respectivement N1 et N2 avec 5<N1,N2<100. Les tableaux contiennent des entiers distincts.

Ecrire un programme en C permettant de calculer l'intersection entre T1 et T2

1. Décomposer le problème en fonctions

2. Implémenter les fonctions en C

3. Proposer une fonction main de test

```c
#include<stdio.h>
#include<stdlib.h>
#include <time.h>

/*++++++++++++++++++++++++++++++++++++++----------
++++++++++++++++++++++++++++++++++++++*/

int Taille_du_tableau(int n)
{
    int i;
    do
    {
        printf("donner la taille du tableau \t");
        scanf("%d",&n);
    }while(n<=5 && n>=100);
    return n;
}

/*++++++++++++++++++++++++++++++++++++++----------
++++++++++++++++++++++++++++++++++++++*/

void Remplir(int T[],int N)
{
    int i;

    for(i=0;i<N;i++)
    {
        printf("donner l'elementdu tableau T[%d]\n",i);
        scanf("%d",&T[i]);
    }
}
```

```c
/*++++++++++++++++++++++++++++++++++++----------
++++++++++++++++++++++++++++++++++++++*/

void recherche(int T1[],int N,int T2[],int M,int Ti[],int *k)
{
    int i,j;
    int L=M+N;
    *k=0;

        for(i=0,j=0;i<L,j<L;i++,j++)
        {
           if(T1[i]=T2[j])
           {

             Ti[*k]=T1[i];
             (*k)++;

           }

        }
}

/*++++++++++++++++++++++++++++++++++++----------
++++++++++++++++++++++++++++++++++++++*/

void afficher (int Ti[], int K)
 {
    for(int i=0 ; i<K ;i++)
       {
            printf("| %d\t",Ti[i]) ;
       }
 }

/*++++++++++++++++++++++++++++++++++++----------
++++++++++++++++++++++++++++++++++++++*/

 main()
 {
    int *T1,*T2,*Ti,M,N,K;

    N=Taille_du_tableau(N);
    M=Taille_du_tableau(M);

    T1 = (int*)malloc(N) ;
```

```
    T2 = (int*)malloc(M) ;
    Ti = (int*)malloc(M+N) ;


    Remplir(T1,N);
    Remplir(T2,M);


    recherche(T1,N,T2,M,Ti,&K) ;


    afficher(Ti,K);
    printf("\n");
    printf("Temps ecoule = %d millisecondes\n",clock());


}
```

Exercice 2 :

Soient deux tableaux T1 et T2 de taille respectivement N1 et N2 avec 5<N1,N2<100. Ils contiennent des entiers distincts et triés dans l'ordre croissant. L'objectif est le calcul de l'intersection entre T1 et T2.

1. La solution de l'exercice 1 est-elle une solution pour l'exercice 2? Si oui, est-elle une solution optimale? →la solution de l'exercice 1 est la solution pour l'exercice 2 mais elle n'est pas optimale car on a le cas du tableau trié.

2. Quelles sont les fonctions à rectifier par rapport à l'exercice 1 -→la fonction de remplissage doit changer.

3. Implémenter les fonctions à rectifier en C

```c
#include<stdio.h>
#include<stdlib.h>
#include <time.h>

/*++++++++++++++++++++++++++++++++++++----------
++++++++++++++++++++++++++++++++++++++*/

int Taille_du_tableau(int n)
{
    int i;
    do
    {
        printf("donner la taille du tableau \t");
        scanf("%d",&n);
    }while(n<=5 && n>=100);
    return n;
}

/*++++++++++++++++++++++++++++++++++++----------
++++++++++++++++++++++++++++++++++++++*/

void Remplir(int T[],int N)
{
    int i;

    for(i=0;i<N;i++)
    {
        printf("donner l'elementdu tableau T[%d]\n",i);
```

```
        scanf("%d",&T[i]);
    }
}

/*++++++++++++++++++++++++++++++++++----------
++++++++++++++++++++++++++++++++++++++*/

void Tri(int T[],int n)
{
    int i,j,aux;
    for(i=0;i<n-1;i++)
      {
        int m=i;
        for(j=i+1;j<n;j++)

                if(T[j]<T[m])
                  {
                      m=j;
                      aux=T[i];
                      T[i]=T[j];
                      T[j]=T[i];
                  }
      }
}

/*++++++++++++++++++++++++++++++++++----------
++++++++++++++++++++++++++++++++++++++*/

void recherche(int T1[],int N,int T2[],int M,int Ti[],int *k)
{
    int i,j;
    int L=M+N;
    *k=0;

        for(i=0,j=0;i<L,j<L;i++,j++)
        {
           if(T1[i]=T2[j])
           {

             Ti[*k]=T1[i];
             (*k)++;

           }

        }
```

```c
}

/*++++++++++++++++++++++++++++++++++++----------
++++++++++++++++++++++++++++++++++++++*/

void afficher (int Ti[], int K)
 {
    for(int i=0 ; i<K ;i++)
       {
            printf("| %d\t",Ti[i]) ;
       }
 }

/*++++++++++++++++++++++++++++++++++++----------
++++++++++++++++++++++++++++++++++++++*/

 main()
 {
    int *T1,*T2,*Ti,M,N,K;

    N=Taille_du_tableau(N);
    M=Taille_du_tableau(M);

    T1 = (int*)malloc(N) ;
    T2 = (int*)malloc(M) ;
    Ti = (int*)malloc(M+N) ;


    Remplir(T1,N);
    Remplir(T2,M);

    Tri(T1,N);
    Tri(T2,M);

    afficher(T1,N);
        printf("\n");

    afficher(T2,M);
        printf("\n");


    printf("\n");

    recherche(T1,N,T2,M,Ti,&K) ;
```

```
   afficher(Ti,K);
   printf("\n");
   printf("Temps ecoule = %d millisecondes\n",clock());


}
```

Exercice 3 :

Soient les tableaux T1 contenant N1 entiers distincts et T2 contenant N2 entiers distincts et triés dans l'ordre croissant avec 5<N1,N2<100 . L'objectif est de calculer l'intersection entre T1 et T2.

1. Discuter les solutions des deux exercices précédents par rapport à la configuration initiale des tableaux T1 et T2.

2. Implémenter en C la fonction permettant de calculer l'intersection.

```c
#include<stdio.h>
#include<stdlib.h>
#include <time.h>

/*++++++++++++++++++++++++++++++++++++----------
++++++++++++++++++++++++++++++++++++++*/

int Taille_du_tableau(int n)
{
    int i;
    do
    {
        printf("donner la taille du tableau \t");
        scanf("%d",&n);
    }while(n<=5 && n>=100);
    return n;
}

/*++++++++++++++++++++++++++++++++++++----------
++++++++++++++++++++++++++++++++++++++*/

void Remplir(int T[],int N)
{
    int i;

    for(i=0;i<N;i++)
    {
        printf("donner l'elementdu tableau T[%d]\n",i);
        scanf("%d",&T[i]);
    }
}

/*++++++++++++++++++++++++++++++++++++----------
++++++++++++++++++++++++++++++++++++++*/
```

```c
void Tri(int T[],int n)
{
    int i,j,aux;
    for(i=0;i<n-1;i++)
      {
        int m=i;
        for(j=i+1;j<n;j++)

                if(T[j]<T[m])
                  {
                      m=j;
                      aux=T[i];
                      T[i]=T[j];
                      T[j]=T[i];
                  }
      }
}

/*++++++++++++++++++++++++++++++++++++----------
++++++++++++++++++++++++++++++++++++*/

void recherche(int T1[],int N,int T2[],int M,int Ti[],int *k)
{
    int i,j;
    int L=M+N;
    *k=0;

        for(i=0,j=0;i<L,j<L;i++,j++)
        {
           if(T1[i]=T2[j])
           {

             Ti[*k]=T1[i];
             (*k)++;

           }

        }
}

/*++++++++++++++++++++++++++++++++++++----------
++++++++++++++++++++++++++++++++++++*/

void afficher (int Ti[], int K)
```

```
{
   for(int i=0 ; i<K ;i++)
      {
           printf("| %d\t",Ti[i]) ;
      }
}

/*++++++++++++++++++++++++++++++++++----------
++++++++++++++++++++++++++++++++++++*/

main()
{
   int *T1,*T2,*Ti,M,N,K;

   N=Taille_du_tableau(N);
   M=Taille_du_tableau(M);

   T1 = (int*)malloc(N) ;
   T2 = (int*)malloc(M) ;
   Ti = (int*)malloc(M+N) ;


   Remplir(T1,N);
   Remplir(T2,M);

   Tri(T2,M);

   afficher(T1,N);
       printf("\n");

   afficher(T2,M);
       printf("\n");


   printf("\n");

   recherche(T1,N,T2,M,Ti,&K) ;


   afficher(Ti,K);
   printf("\n");
   printf("Temps ecoule = %d millisecondes\n",clock());
}
```

Exercice 4 :

1. Discuter la complexité du traitement de calcul d'intersection des solutions données dans les trois exercices précédents.

2. Calculer le temps d'exécution en utilisant les 3 solutions données pour les trois exercices sur une configuration des deux tableaux T1 et T2 comme elle est indiquée dans l'exercice 2.
Sachant que les tailles des tableaux données dans les exemples sont petites pour avoir une différence remarquable dans le temps d'exécution des 3 solutions, proposez une démarche permettant d'avoir une telle différence.

Exercice	1	2	3
Entiers	Deux tableaux non triées	Deux tableaux triées	T1 : non trié T2 : trié
Condition	5<N,M<100	5<N,M<100	5<N,M<100
Temps D'execution	8037 millisecondes	10018 millisecondes	6671 millisecondes
Complexité Spatiale temporelle	O(n) O(n°2)	O(n) O(n(n-1)/2)	O(n) O(n°2/2)

TP2

```c
#include<stdio.h>
#include<string.h>
#include<stdlib.h>

typedef struct dateNaissance
{
unsigned j;
unsigned m;
unsigned a;
}dateNaissance;

/*première proposition*/
/*typedef struct personne
{
char nom[20];
char prenom[20];
char matricule[8];
dateNaissance dn;
}personne;*/

/*2ème proposition*/
typedef struct personne
{
char *nom;
char *prenom;
char *matricule;
dateNaissance dn;
}personne;


void remplir_tab_personnes(personne P[], int n)
{
char ch[100];
int i;
for(i=0;i<n;i++)
{
printf("*********Personne %d\n",i+1);
printf("nom: ");
fflush(stdin);
gets(ch);
P[i].nom=(char*)malloc(sizeof(char)*(strlen(ch)+1));
strcpy(P[i].nom,ch);
printf("prenom: ");
```

```
gets(ch);
P[i].prenom=(char*)malloc(sizeof(char)*(strlen(ch)+1));
strcpy(P[i].prenom,ch);
printf("matricule: ");
gets(ch);
P[i].matricule=(char*)malloc(sizeof(char)*(strlen(ch)+1));
strcpy(P[i].matricule,ch);
printf("Date de naissance :\n");
printf("jour: ");
scanf("%u",&P[i].dn.j);
printf("mois: ");
scanf("%u",&P[i].dn.m);
printf("annee: ");
scanf("%u",&P[i].dn.a);
}
}

void affiche_tab_personnes(personne P[], int n)
{
int i;
for(i=0;i<n;i++)
{
printf("*********Personne %d\n",i+1);
printf("nom: %s\n",P[i].nom);
printf("prenom: %s\n",P[i].prenom);
printf("matricule: %s\n",P[i].matricule);
printf("Date de naissance : %u/%u/%u\n",P[i].dn.j,P[i].dn.m,P[i].dn.a);
}
}
void affiche_tab_pointeurs_personnes(personne* P[], int n)
{
int i;
for(i=0;i<n;i++)
{
printf("*********Personne %d\n",i+1);
printf("nom: %s\n",P[i]->nom);
printf("prenom: %s\n",P[i]->prenom);
printf("matricule: %s\n",P[i]->matricule);
printf("Date de naissance : %u/%u/%u\n",P[i]->dn.j,P[i]->dn.m,P[i]->dn.a);
}
}

void remplir_tab_pointeurs_personnes(personne **P/*personne* P[]*/, int n)
{
int i;
```

```
char ch[100];
for(i=0;i<n;i++)
{
printf("*********Personne %d\n",i+1);
P[i]=(personne*)malloc(sizeof(personne));
printf("nom: ");
fflush(stdin);
gets(ch);
P[i]->nom=(char*)malloc(sizeof(char)*(strlen(ch)+1));
strcpy(P[i]->nom,ch);
printf("prenom: ");
gets(ch);
P[i]->prenom=(char*)malloc(sizeof(char)*(strlen(ch)+1));
strcpy(P[i]->prenom,ch);
printf("matricule: ");
gets(ch);
P[i]->matricule=(char*)malloc(sizeof(char)*(strlen(ch)+1));
strcpy(P[i]->matricule,ch);
printf("Dat.e de naissance :\n");
printf("jour: ");
scanf("%u",&P[i]->dn.j);
printf("mois: ");
scanf("%u",&P[i]->dn.m);
printf("annee: ");
scanf("%u",&P[i]->dn.a);
}
}


void sauvegarder_tab_personnes_fichier(personne P[],int n, char*path)
{
FILE *file;
int i;

file=fopen(path,"w");
for(i=0;i<n;i++)
{
fprintf(file,"%s\t%s\t%s\t%u/%u/%u\n",P[i].nom,P[i].prenom,P[i].matricule,P[i].dn
.j,P[i].dn.m,P[i].dn.a);
}
fclose(file);
}


void chargement_donnees_personnes_tab(personne P[],int* n, char*path)
```

```
{
FILE *file;
char nom[100], prenom[100],matricule[10];
unsigned j,m,a;
int i=0;
file=fopen(path,"r");
while(!feof(file))
{
fscanf(file,"%s\t%s\t%s\t%u/%u/%u\n",nom,prenom,matricule,&j,&m,&a);
P[i].nom=(char*)malloc(sizeof(char)*(strlen(nom)+1));
strcpy(P[i].nom,nom);

P[i].prenom=(char*)malloc(sizeof(char)*(strlen(prenom)+1));
strcpy(P[i].prenom,prenom);

P[i].matricule=(char*)malloc(sizeof(char)*(strlen(matricule)+1));
strcpy(P[i].matricule,matricule);

P[i].dn.j=j;
P[i].dn.m=m;
P[i].dn.a=a;
i++;
}
*n=i;
}

void chargement_donnees_personnes_tab_dynamique( personne **P,int* n, char*path)
{
FILE *file;
char nom[100], prenom[100],matricule[10];
unsigned j,m,a;
int i=0,nbrLigne=0;
file=fopen(path,"r");
while(!feof(file))
{
fscanf(file,"%s\t%s\t%s\t%u/%u/%u\n",nom,prenom,matricule,&j,&m,&a);
nbrLigne++;
}
fclose(file);
printf("nbrLigne=%d\n",nbrLigne);
*n=nbrLigne;
(*P)=(personne*)malloc(nbrLigne*sizeof(personne));

file=fopen(path,"r");
while(!feof(file))
```

```
{
fscanf(file,"%s\t%s\t%s\t%u/%u/%u\n",nom,prenom,matricule,&j,&m,&a);
(*P)[i].nom=(char*)malloc(sizeof(char)*(strlen(nom)+1));
strcpy((*P)[i].nom,nom);

(*P)[i].prenom=(char*)malloc(sizeof(char)*(strlen(prenom)+1));
strcpy((*P)[i].prenom,prenom);

(*P)[i].matricule=(char*)malloc(sizeof(char)*(strlen(matricule)+1));
strcpy((*P)[i].matricule,matricule);

(*P)[i].dn.j=j;
(*P)[i].dn.m=m;
(*P)[i].dn.a=a;
i++;
}
fclose(file);
}

/*void tri_insertion_personnes_nom(personne T[],int n)
{
int i,j;
personne v;
for(i=1;i<n;i++)
{
v=T[i];
j=i-1;
while(j>=0 && strcmp(T[j].nom,v.nom)>0)
{
T[j+1]=T[j];
j--;
}
T[j+1]=v;
}
}

void tri_insertion_personnes_prenom(personne T[],int n)
{
int i,j;
personne v;
for(i=1;i<n;i++)
{
v=T[i];
j=i-1;
while(j>=0 && strcmp(T[j].prenom,v.prenom)>0)
```

```
{
T[j+1]=T[j];
j--;
}
T[j+1]=v;
}
}*/

void tri_insertion_personnes(personne T[],int n, int (*oper)(personne,personne))
{
int i,j;
personne v;
for(i=1;i<n;i++)
{
v=T[i];
j=i-1;
while(j>=0 && (*oper)(T[j],v)>0)//strcmp(T[j].nom,v.nom)>0)
{
T[j+1]=T[j];
j--;
}
T[j+1]=v;
}
}
int compare_personne_nom(personne p1,personne p2)
{
return strcmp(p1.nom,p2.nom);
}

int compare_personne_prenom(personne p1,personne p2)
{
return strcmp(p1.prenom,p2.prenom);
}
int compare_personne_matricule(personne p1,personne p2)
{
return strcmp(p1.matricule,p2.matricule);
}
void main()
{
personne tabPers[100];
personne *tab[100];
personne *tabDyn;
int nbr;
```

```c
do
{
printf("nbr=");
scanf("%d",&nbr);
}while(nbr<=0 || nbr>100);

printf("\n\n----------Remplir Tab\n\n\n");
remplir_tab_personnes(tabPers, nbr);
//remplir_tab_pointeurs_personnes(tab,nbr);
printf("\n\n----------Affichage Tab\n\n\n");
affiche_tab_personnes(tabPers,nbr);
//affiche_tab_pointeurs_personnes(tab, nbr);

printf("\n\n----------Trier par Nom Tab\n\n\n");
tri_insertion_personnes(tabPers,nbr,compare_personne_nom);
printf("\n\n----------Affichage Tab\n\n\n");
affiche_tab_personnes(tabPers,nbr);
printf("\n\n----------Sauvegrade fichier\n\n\n");
sauvegarder_tab_personnes_fichier(tabPers,nbr, "sauvegarde\\monFichier_nom.txt");


printf("\n\n----------Trier par Prenom Tab\n\n\n");
tri_insertion_personnes(tabPers,nbr,compare_personne_prenom);
printf("\n\n----------Affichage Tab\n\n\n");
affiche_tab_personnes(tabPers,nbr);
printf("\n\n----------Sauvegrade fichier\n\n\n");
sauvegarder_tab_personnes_fichier(tabPers,nbr, "sauvegarde\\monFichier_prenom.txt
");

printf("\n\n----------Trier par Matricule Tab\n\n\n");
tri_insertion_personnes(tabPers,nbr,compare_personne_matricule);
printf("\n\n----------Affichage Tab\n\n\n");
affiche_tab_personnes(tabPers,nbr);
printf("\n\n----------Sauvegrade fichier\n\n\n");
sauvegarder_tab_personnes_fichier(tabPers,nbr, "sauvegarde\\monFichier_matricule.
txt");

/*printf("\n\n----------Loading fichier\n\n\n");
chargement_donnees_personnes_tab(tabPers,&nbr, "sauvegarde\\monFichier.txt");

printf("\n\n----------Affichage Tab\n\n\n");
affiche_tab_personnes(tabPers,nbr);*/

/* printf("\n\n----------Loading fichier\n\n\n");
```

```
chargement_donnees_personnes_tab_dynamique(&tabDyn,&nbr, "sauvegarde\\monFichier.
txt");
printf("\n\n----------Affichage Tab\n\n\n");
affiche_tab_personnes(tabDyn,nbr);*/
}
```

#Elglace_w_il_bira


BECK'S
BECK'S