Correction Exercices diversifiés : programmation C

Exercice 1 :

```
#include <stdio.h>
#include <stdlib.h>
void main()
  {
    char c, preced=' ';
    int alph=0, chif=0;
    int l=0,a=0,ch=0;

    while ( (c=getchar()) !='#')
    {
      if (c==' ' || c==',' || c==';'|| c=='\n')
      {
        if (l!=0)
        {
          if(l==a) alph++;
          if(l==ch) chif++;
        }
        l=0;a=0; ch=0; preced=c;
      }
      else
      {
        if((c>='a' && c<='z') || (c>='A'&& c<='Z'))
          a++;
        else if (c>='0' && c<='9')
          ch++;
        l++;
        preced=c;
      }
    }
```

```
    printf("le nbre de mots alphabetiques est %d, le nbre de mots chiffres est %d", alph, chif);
}
```

Exercice 2 :

Solution en un seul bloc :

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
void main()
{
   int tab1[50],tab2[50], n1, n2,i,j=0,k;
   printf("saisir taille de tab1 et taille de n2 ");
   scanf("%d%d",&n1, &n2);
   printf("\n remplissage du 1er tableau ");
   for(i=0; i<n1; i++)
   scanf("%d", tab1+i);
   printf("\n remplissage du 2rd tableau ");
   for(i=0; i<n2; i++)
   scanf("%d", tab2+i);
   printf("\n affichage du 1er tableau ");
   for(i=0; i<n1; i++)
   printf("%d\t", *(tab1+i));
   printf("\n affichage du 2rd tableau ");
   for(i=0; i<n2; i++)
   printf("%d\t", *(tab2+i));
   for(i=0; i<n1;)
   {
      if( *(tab1+i)== *(tab2+j))
      {
         // parcours de tab1 et tab2 en parallèle et comparaison
         for(; j<n2 ; j++)
            if ( *(tab1+i+j)!= *(tab2+j)) break;
```

```
            if(j==n2) // tab2 existe dans tab1
            {
               // on va utiliser un autre indice k pour le décalage
               // suppression de tab2 de tab1
               for (k=i; k<(n1-n2); k++)
               *(tab1+k)=*(tab1+k+n2);
               n1=n1-n2; // la nouvelle taille du tableau
            }
            j=0;
         }
         else i++;//i ne doit avancer que s'il n ya pas eu de suppression de tab2
         // cad: i n'avance pas quand on supprime une suite de tab2
         //exemple: si tab1: 1231234 et tab2: 123; en supprimant tab2,
         // tab1 contiendra:1234;
         // c'est la premiere suite qui a été supprimée
         // si on fait i++; on se positionnera sur la valeur 2, et dans ce cas la
         // deuxième séquence 123 ne sera pas supprimée de 1234
      }
      printf(" \n affichage du nouveau tableau ");
      for(i=0; i<n1; i++)
      printf("%d", *(tab1+i));
      getch();
   }
```

Solution avec les fonctions :

```
   #include<stdio.h>
      void afficher(int *t, int x)
      {
         int *q1;
         printf("\n|");
         for(q1=t; q1<t+x; q1++)
         printf(" %d |",*q1);
```

```
}
int saisir(int *t)
{
   int i,x,*q1;
   printf("saisir la taille du tableau : ");
   scanf("%d", &x);
   for(i=0,q1=t;q1<t+x;i++,q1++)
   {
   printf("tab1[%d]:",i);
   scanf("%d", q1);
   }
   afficher(t,x);
   return x;
}
void supprimer(int *t1, int *x1, int *t2, int x2)
{
   int *q1,*q2,i,ok;
   for(q1=t1;q1<t1+*x1;q1++)
   {
      for(i=0,q2=t2;q2<t2+x2;i++,q2++)
      if(*q2==*(q1+i))
         ok=1;
      else
         ok=0;
      if(ok==1)
      {
         int *q=q1;
         while(q1<t1+*x1-x2)
         {
            i=0;
            *(q1+i)=*(q1+x2+i);
```

```
                q1++;
                i++;
            }
            *x1=*x1-x2;
            q1=q;
        }
    }
}
void main ( )
{
    int tab1[50],tab2[50],n1,n2;
    printf("\n***** TABLEAU 1 *****\n");
    n1=saisir(tab1);
    printf("\n\n***** TABLEAU 2 *****\n");
    n2=saisir(tab2);
    supprimer(tab1,&n1,tab2,n2);
    printf("\n\n***** TABLEAU RESULTAT*****\n");
    afficher(tab1,n1);
}
```

Exercice 3 :

```
#include<stdio.h>
#include<stdlib.h>
int saisir_n()
{
   int x;
   do
   {
      printf("\n saisir nbr elements ");
      scanf("%d",&x);
   }
   while(x<10||x>20);
```

```
    return x;
}
void creer(int n)
{
    int *tab;
    tab=(int *)malloc(n*sizeof(int));
    if(!tab) exit(-1);
}
void remplir(int *t, int n)
{
    int i;
    for(i=0;i<n;i++)
        scanf("%d",t+i);
}
void afficher(int *t, int n)
{
    int i;
    for(i=0;i<n;i++)
        printf("%3d",*(t+i));
}
int max_suites(int *t, int n)
{
    int i,j,l=1,lmax=1;
    for (i=0;i<n-1;i++)
    {
        if(*(t+i)==(*(t+i+1))-1)
            l++;
        else if(lmax<l && l!=1)
        {
            lmax=l;
            l=1;
```

```
        }
    }
    if(lmax<l)
        lmax=l;
    return lmax;
}

void main ( )
{
    int n,*tab,lmax;
    n=saisir_n();
    creer(n);
    printf("\n ******* remplissage *******\n ");
    remplir(tab,n);
    printf("\n ******* affichage *******\n ");
    afficher(tab,n);
    lmax=max_suites(tab,n);
    printf("\n la taille de la plus longue suite de nombres successifs est %d",lmax);
}
```

Exercice 4 :

```
/***Exercice 4: ***/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

//prototypes des fonctions
void remplir_tab(int*,int);
void afficher_tab(int*,int);
void inverser_tab(int*,int);
int max_tab(int*,int);
```

```
int min_tab(int*,int);

int main()
{
   int *tab,N;
   FILE* fichier=NULL;

   do
   {
      printf("saisir la taille du tableau!\n");
      scanf("%d",&N);
   }
   while(N<10 || N>50);
   tab = (int*)malloc(N*sizeof(int));
   if(!tab) exit(-1);
   //Remplissage du tableau
   remplir_tab(tab,N);
   //Affichage du tableau
   afficher_tab(tab,N);
   //Inverser le tableau
   inverser_tab(tab,N);
   //Affichage du tableau
   afficher_tab(tab,N);
   printf("le maximum du tab est: %d \n",max_tab(tab,N));
   printf("le minimum du tab est: %d \n",min_tab(tab,N));

   fichier=fopen("inverse.txt","w");
   if(fichier!=NULL)
   {
```

```
        fputs("le tableau inversé:\n",fichier);
        for(int j=0;j<N;j++)
        {
           fprintf(fichier,"%d\t",*(tab+j));
        }
        fprintf(fichier,"\nle maximum du tab est: %d\n",max_tab(tab,N));
        fprintf(fichier,"le minimum du tab est: %d\n",min_tab(tab,N));
        fclose(fichier);
    }
    else
        printf("impossible d'ouvrir le fichier");
}
void remplir_tab(int *tab,int N)
{
    int i;
    printf("Remplissage du tableau\n");
    for(i=0;i<N;i++)
        scanf("%d",tab+i);
}
 void afficher_tab(int*tab,int N)
 {
     printf("Affichage du tableau\n");
    for(int i=0;i<N;i++)
        printf("%d\t",*(tab+i));
    printf("\n");
 }
void inverser_tab(int* tab, int N)
{
    int i,inter;
```

```
    for(i=0;i<(N/2);i++)
    {
        inter = tab[i];
        tab[i]=tab[N-1-i];
        tab[N-1-i]=inter;
    }
}
int max_tab(int*tab,int N)
{
    int i,max=0;
    for(i=0;i<N;i++)
    {
        if(*(tab+i)>max)
            max = *(tab+i);
    }
    return max;
}
int min_tab(int*tab,int N)
{
    int i,min=0;
    for(i=0;i<N;i++)
    {
        if(*(tab+i)<min)
            min = *(tab+i);
    }
    return min;
}
```

Exercice 5 :

```
#include <stdio.h>
```

```c
#include <stdlib.h>
#include <string.h>
int IsPalindrome(char *set);

  int main()
{
   FILE *P_FICHIER1, *P_palindrome; /* pointeur sur FILE */
   char Mot[50];
   int r, i=0, j=0;
   P_FICHIER1 = fopen("mots.txt", "r");  /* ouverture fichier */
   P_palindrome = fopen("palindrome.txt", "w");  /* ouverture fichier */

   if (P_FICHIER1==NULL|| P_palindrome==NULL)
      printf("impossible to open the file\n");

   r=fscanf(P_FICHIER1,"%s",Mot);
   while(r!= EOF)
     {
        printf(" Le %d mot est %s \n",i+1, Mot);
        if(IsPalindrome(Mot))
           {
              fprintf(P_palindrome, "%s \n", Mot);
           }
        r=fscanf(P_FICHIER1,"%s",Mot);
        i++;
     }
   fclose(P_FICHIER1);
   fclose(P_palindrome);

   /* lire le ficher de palindrome */
```

```
    P_palindrome = fopen("palindrome.txt", "r");  /* ouverture fichier */
    if (P_palindrome==NULL)
        printf("impossible to open the file\n");

    printf("Afficher les elements du fichier palindrome \n");
    fscanf(P_palindrome,"%s", Mot);
        while (!feof(P_palindrome))
        {
            printf("le  %d mot est %s \n", j+1, Mot);
            fscanf(P_palindrome,"%s", Mot);
            j++;
        }

    fclose(P_palindrome);
    return 0;
}

int IsPalindrome(char *set)
{
    int j, l=strlen(set);
    if (l==0||l==1)
        return 1;

for (j=0; j<l/2;j++)
    {
        if (set[j]!=set[l-1-j])
            return 0;
    }

    return 1;
}
```

Exercice 6 :

1. Ecrire une fonction ETUDIANT saisir_etudiant() qui lit et retourne un etudiant (cin, nom, date de naissance et les trois notes). La moyenne est calculée en fonction des 3 notes saisies.

```
float calcul_moyenne (float *t)
{
   float m=0.0;
   for(int i=0;i<3; i++)
   m+=*(t+i);
   return (m/3);
}
void remplir_notes (float *t)
{
   int i;
   for(i=0; i<3; i++)
   scanf("%f", t+i);
}
ETUDIANT saisir_etudiant()
{
   ETUDIANT etd;
   printf("\n saisir cin ");
   scanf("%d", &etd.inf.cin);
   printf("\n saisir le nom ");
   scanf("%s", etd.inf.nom);
   printf("\n saisir la date de naissance ");
   scanf("%d%d%d", &etd.inf.date_naiss.jour, &etd.inf.date_naiss.mois,
   &etd.inf.date_naiss.annee);
   printf("\n saisir les notes ");
   remplir_notes (etd.inf.notes);
   etd.moyenne=calcul_moyenne (etd.inf.notes);
   // calcul_moyenne2(etd.inf.notes, &etd.moyenne);
```

```
    return etd;
}
```

2. Même question: void saisir_etudiant2 (ETUDIANT*)

```
void saisir_etudiant2 ( ETUDIANT *e)
{
   printf("\n saisir cin ");
   scanf("%d", &e->inf.cin);
   printf("\n saisir le nom ");
   scanf("%s", e->inf.nom);
   printf("\n saisir la date de naissance ");
   scanf("%d%d%d", &e->inf.date_naiss.jour, &e->inf.date_naiss.mois,&e->inf.date_naiss.annee);
   printf("\n saisir les notes ");
   remplir_notes (e->inf.notes);
   e->moyenne=calcul_moyenne (e->inf.notes);
   // calcul_moyenne2(e->inf.notes, &e->moyenne);
}
```

3. Ecrire la fonction afficher_etudiant (ETUDIANT etd) qui affiche toutes les informations sur un étudiant (cin, nom, date de naissance, les trois notes et sa moyenne)

```
void afficher_notes (float *t)
{
   int i;
   for(i=0; i<3; i++)
   printf("%5.2f \t", *(t+i));
}
void afficher_etudiant(ETUDIANT etd )
{
   printf("\n le cin est: %d ", etd.inf.cin);
   printf("\n le nom est: %s", etd.inf.nom);
   printf ("\n la date de naissance est: %d / %d / %d",
   etd.inf.date_naiss.jour, etd.inf.date_naiss.mois, etd.inf.date_naiss.annee);
   printf("\n affichage des notes ");
```

```
        afficher_notes (etd.inf.notes);
        printf("\n la moyenne est: %5.2f ", etd.moyenne);
    }
```

4. Programme principal

a. Remplissage et affichage par déclaration

```
void main()
{
   ETUDIANT etd;
   etd = saisir_etudiant();
   //saisir_etudiant2(&etd);
   afficher_etudiant(etd);
}
```

b. Remplissage et affichage par allocation

```
void main()
{
   ETUDIANT *etd;
   etd = (ETUDIANT*) malloc(sizeof(ETUDIANT));
   //*etd=saisir_etudiant();
   saisir_etudiant2(etd);
   afficher_etudiant(*etd);
}
```

Exercice 7 :

Remplir le tableau suivant (**seulement les cellules contenant des pointillés**), en supposant que : &tab[0] : F50 ; &tab[1] : F54 ; &tab[2] : F58 ; &tab[3] : F5C ; &tab[4] : F60 ;

Instructions	p	*p	q	*q	tab[0]	tab[1]	tab[2]	tab[3]	tab[4]
p=tab	F50	4			4	7…	2…	9…	6…
q=tab+4;			F60	6					
q-=2;			F58	2					
p++;	F54	7							
*(tab+3)=*p+*q;								9	
(tab+4)=(q-1)-*(p+1);									5
p+=*tab-*q ;	F5C	9							
q-=*p- *(tab+3);			F58	2					
tab[0]=p-q ;					1				

p=q ;	F58	2							

Exercice 8 :

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define IMAX 5


typedef struct list
{
   char *nom;
   struct list *next;
} Liste;
Liste* ajout_tete(Liste *p, char* chaine);
void afficher_Liste(Liste *p);
Liste* rechercher_element(Liste *p, char* chaine);
int main()
{
   Liste *pr1=NULL, *pr11=NULL;
   char nom[50]; //=malloc(sizeof(char));

   for (int i=0; i<IMAX; i++)
   {
      printf("saisir le %d mot \n", i+1);
      scanf("%s", nom);
      pr1=ajout_tete(pr1,nom);
   }
   afficher_Liste(pr1);
   printf("saisir le mot a chercher \n");
   scanf("%s", nom);
```

```
    pr11=rechercher_element(pr1, nom);
    afficher_Liste(pr11);
        return 0;
}

Liste* ajout_tete(Liste *p, char* chaine)
{
    Liste *nouvelle;
    nouvelle = (Liste*)malloc(sizeof(Liste));
    //nouvelle->nom = malloc(sizeof(char)*(strlen(chaine)));
    nouvelle->nom = malloc(sizeof(char)*(strlen(chaine)+1));
    strcpy(nouvelle->nom , chaine);
    nouvelle -> next= p;
    return nouvelle;
}
void afficher_Liste(Liste *p)
{
    Liste *l;
    l=p;
    if(p==NULL)
        printf("rien a afficher, liste est vide\n");

    while (l!=NULL)
    {
        printf("la cellule rangee a l'adresse %p contient la chaine %s\n", l, l->nom);
        printf("la cellule qui suie est rangee a l'adresse %p \n", l->next);
        l=l->next;
    }
}
Liste* rechercher_element(Liste *p, char* chaine)
{
```

```
   Liste *l=p;
   int i=1;

   while(l!=NULL && strcmp(l->nom, chaine)!=0)
      {i++;
      l=l->next;
      }
   if(l==NULL)
   printf("la chaine existe pas!\n");
   else
   {
   l->next=NULL;
    printf("indice de la cellule est %d \n", i);
   }
   return l;
}
```

Exercice 9 :

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

//Définition de la structure d'une liste simplement chainée  */
typedef struct Cell
{
   int val;
   struct Cell *suiv;
}Cellule;

/**prototypes des fonctions de gestion de la liste**/

Cellule* Creation_Liste();
void initialisation(Cellule *);
```

```
Cellule* ajout_Debut(Cellule*, int);
Cellule* Inserer_Element(Cellule* ,int , int );
void Afficher_Liste(Cellule*);


void main()
{
   Cellule *maListe;
   maListe = Creation_Liste();
   initialisation(maListe);
   maListe= ajout_Debut(maListe,4);
   maListe= ajout_Debut(maListe,3);
   maListe= ajout_Debut(maListe,2);
   maListe= ajout_Debut(maListe,1);
   Afficher_Liste(maListe);
   Inserer_Element(maListe,0,3);
   Afficher_Liste(maListe);

}

/********* Création d'une liste (allocation dynamique)**********/
Cellule* Creation_Liste()
{
   Cellule *p;
   p = (Cellule*)malloc(sizeof(Cellule));
   return p;
}

/******** Initialisation de la liste ************/
void initialisation(Cellule *p)
```

```
{
   p->val=0;
   p->suiv=NULL;
}

/***************** Tester si une liste est vide ********/
int liste_vide(Cellule *p)
{
   return(p==NULL); //retourne 1 si la liste est vide et 0 sinon
}

/********Ajout d'un élément au début de la liste ********/
Cellule* ajout_Debut(Cellule *p,int x)
{
   Cellule *nouveau;
   nouveau = Creation_Liste();
   /**tester si la liste est vide ou non**/
   if(liste_vide(p))
   {
      nouveau->val=x;
      nouveau->suiv=NULL;
      return nouveau;
   }
   else
   {
      nouveau->val=x;
      nouveau->suiv=p;
      return nouveau;
   }
}
```

```
void Afficher_Liste(Cellule *p)
{
   Cellule *l;
   l=p;

   if(liste_vide(p))
      printf("rien a afficher, liste est vide\n");

    while (l!=NULL)
      {
         printf("%d->", l->val);
         l=l->suiv;
      }
      printf("NULL\n");
}

Cellule* Inserer_Element(Cellule* p,int x, int position)
{
   int posSuiv=1;
   Cellule *l;
   l=Creation_Liste();
   l=p;
   if(position==1)
   {
      p=ajout_Debut(p,x);
      return p;
   }
   else
   {
```

```
    while(l!=NULL)
    {
       posSuiv++;
       if(position==posSuiv)
          goto sortie;
       else
          l=l->suiv;
    }

    Cellule *nouveau;
    sortie:nouveau = Creation_Liste();
       nouveau->val=x;
       nouveau->suiv=l->suiv;
       l->suiv=nouveau;
       return p;
       }
}
```