



جامعة ابن طفيل Université Ibn Tofail
 UNIVERSITÉ IBN TOFAIL Ibn Tofail University

Cycle Ingénieur en Génie Informatique

Année universitaire 2022-2023

Complexité et structures de données

Thème :

Compte rendu des TPs



Présenté par : BADRY ZAKARIA

Numéro Apogée : 22014301

Sous la direction de : Pr. AIT LAHCEN Ayoub

SOMMAIRE

TP 1 : -----	3
Exercice 1 : Recherche du maximum -----	3
Exercice 2 : Tri à bulles -----	6
Exercice 3 : Recherche dichotomique -----	8
TP 2 : -----	11
Exercice 1 : -----	11
Exercice 2 : -----	12
Exercice 3 : -----	24
Exercice 4 : -----	26
TP 3 : Les listes chainées -----	28
TP 4 : Les listes chainées (#personne#) -----	48
TP 5 : Les arbres -----	61
TP 6 : Les graphes -----	83

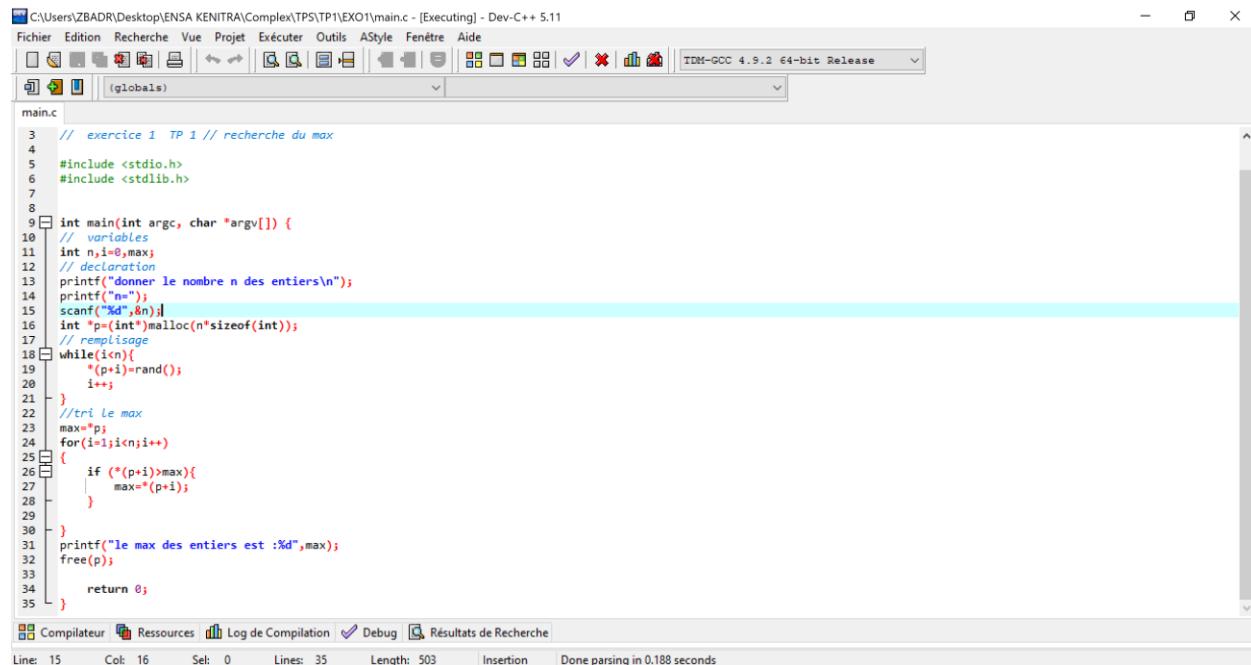
TP1

EXERCICE 1

1. Écrivez un programme en C qui implémente un algorithme de recherche du maximum dans un ensemble de n entiers. Testez-le avec $n = 5$, puis avec $n = 900\,000\,000$ (dans le cas où n est très grand, vous pouvez remplir le tableau en utilisant la fonction `rand()`).
2. Quel est le nombre de comparaisons effectuées dans votre algorithme en termes de n ?
3. Est-ce que votre algorithme est optimal ?

SOLUTION

1 -



The screenshot shows the Dev-C++ IDE interface with the main.c file open. The code implements a simple algorithm to find the maximum value in an array of integers. It includes comments explaining the steps: declaration of variables, reading input, filling the array with random values, and then iterating through the array to find the maximum value. The IDE shows syntax highlighting and a status bar at the bottom.

```
main.c
3 // exercice 1 TP 1 // recherche du max
4 #include <stdio.h>
5 #include <stdlib.h>
6
7
8 int main(int argc, char *argv[]) {
9     // variables
10    int n,i=0,max;
11
12    // declaration
13    printf("donner le nombre n des entiers\n");
14    printf("n=");
15    scanf("%d",&n);
16    int *p=(int*)malloc(n*sizeof(int));
17
18    // remplissage
19    while(i<n){
20        *(p+i)=rand();
21        i++;
22    }
23    //tri le max
24    max=*p;
25    for(i=1;i<n;i++){
26        if (*(p+i)>max){
27            max=*(p+i);
28        }
29    }
30
31    printf("le max des entiers est :%d",max);
32    free(p);
33
34    return 0;
35 }
```

```
// exercice 1 TP 1 // recherche du max

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

int main(int argc, char *argv[]) {
//     variables
int n,i=0,max;
// declaration
printf("donner le nombre n des entiers\n");
printf("n=");
scanf("%d",&n);
int *p=(int*)malloc(n*sizeof(int));
```

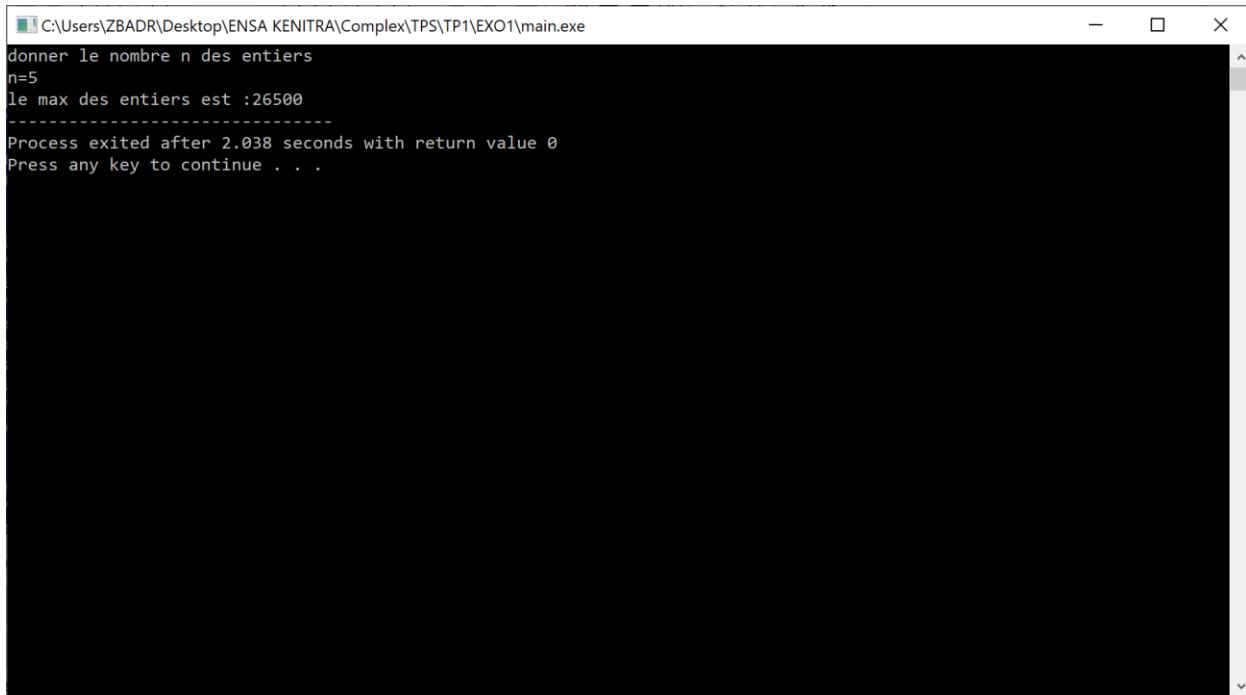
```

// remplisage
while(i<n) {
    *(p+i)=rand();
    i++;
}
//tri le max
max=*p;
for(i=1;i<n;i++)
{
    if  (*(p+i)>max) {
        max=*(p+i);
    }
}
printf("le max des entiers est :%d",max);
free(p);

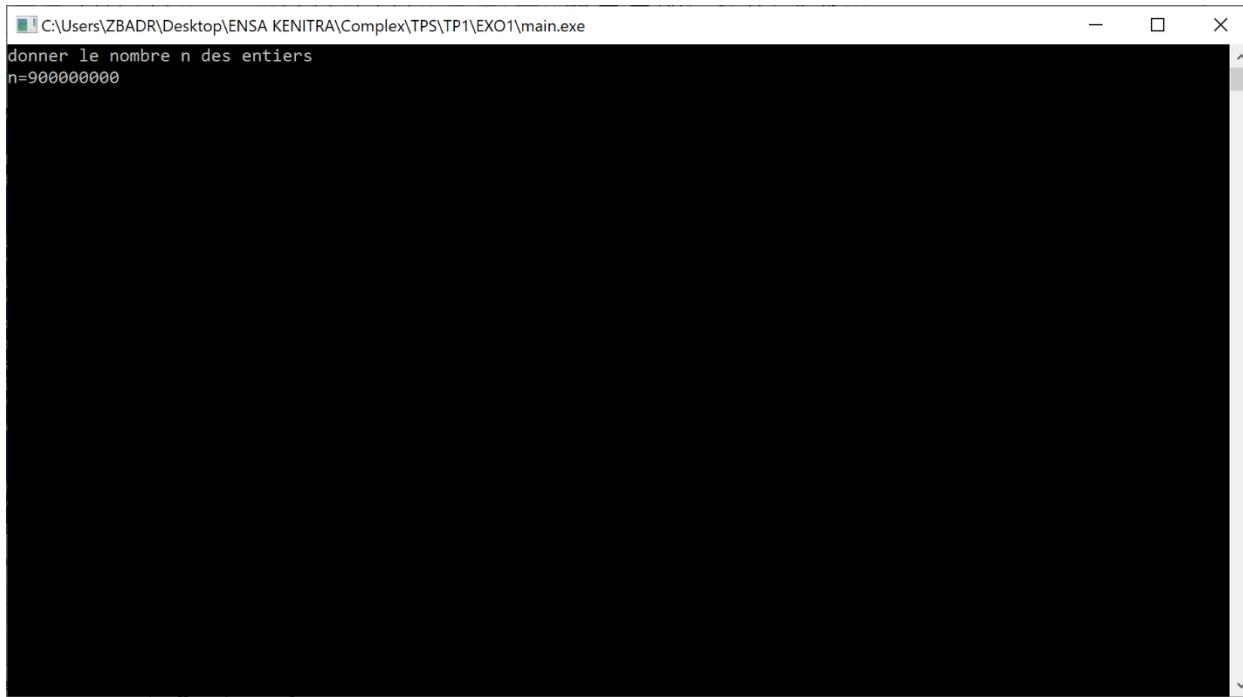
return 0;
}

```

Cas de n=5

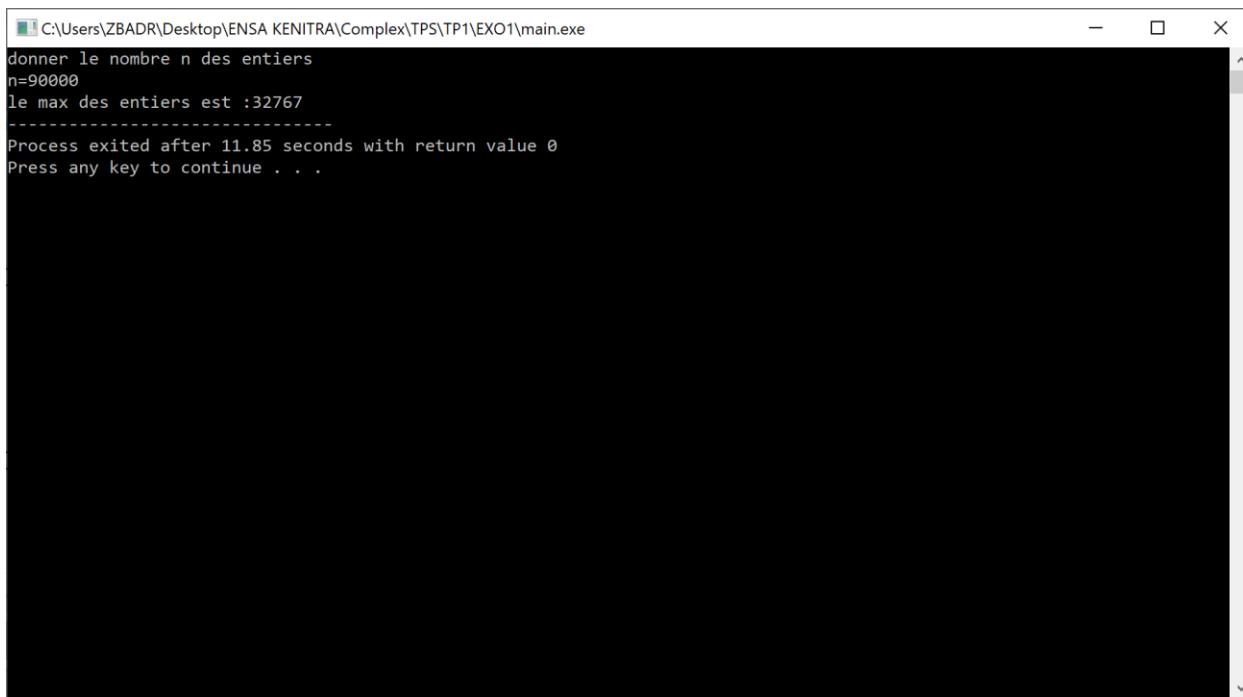


Cas n=900000000



```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP1\EXO1\main.exe
donner le nombre n des entiers
n=900000000
```

Cas n=90000



```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP1\EXO1\main.exe
donner le nombre n des entiers
n=90000
le max des entiers est :32767
-----
Process exited after 11.85 seconds with return value 0
Press any key to continue . . .
```

2- l'algorithme effectue n-1 comparaisons .

3- si l'élément max est place dans la dernière cas du tableau (dans les pires des cas) chaque élément après le premier étant comparé une fois, donc on faites n – 1 comparaisons . l'algorithme est optimale .

EXERCICE 2

Le tri à bulles s'effectue en n–1 étapes (pour un tableau de n éléments). A la ième étape on balaye le tableau en partant de la fin jusqu'à la case i et à chaque fois que l'élément courant est plus petit que son prédecesseur, on échange leur position. Cette méthode a pour effet de faire remonter au fur et à mesure les bulles les plus légères vers la surface.

- 1) Implémentez en langage C le tri à bulles.**
- 2) Calculez et affichez le nombre de comparaison effectuées par ce dernier.**
- 3) Comparez le résultat obtenu avec celui du cours.**

1-

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

/* run this program using the console pauser or add your own getch,
system("pause") or input loop */

int main(int argc, char *argv[]) {
    //variable
    int i=0,n,ech=0;
    //declaration
    printf("donner le nombre n des entiers\n");
    printf("n=");
    scanf("%d",&n);
    int *p=(int*)malloc(n*sizeof(int));
    // remplisage
    while(i<n){
        *(p+i)=rand();
        i++;
    }
    printf("affichage avant tri\n");
    printf("\n");
    // affichage avant tri
    for(i=0;i<n;i++){
        printf("%d \t",*(p+i));
    }

    //tri a bulle
    do{
        int ech=0;
        for(i=0;i<n-1;i++)
            if(*(p+i)>*(p+i+1)) {
```

```

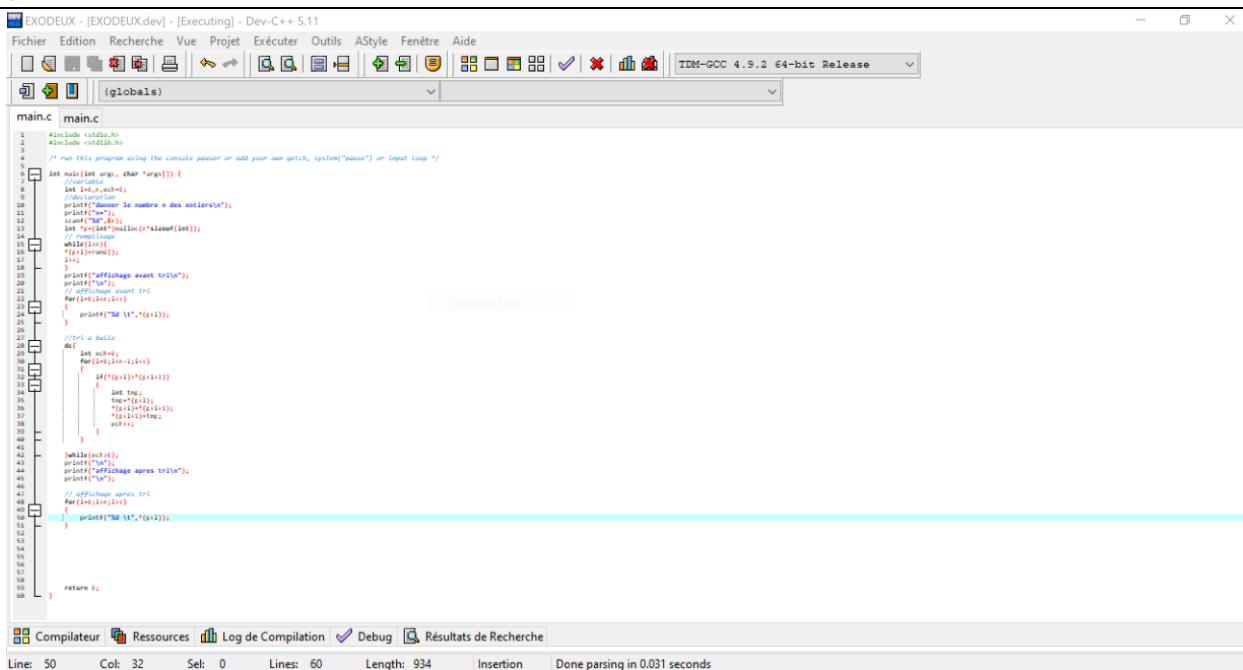
    {
        if(* (p+i)>* (p+i+1))
        {
            int tmp;
            tmp=* (p+i);
            * (p+i)=* (p+i+1);
            * (p+i+1)=tmp;
            ech++;
        }
    }

}while(ech>0);
printf("\n");
printf("affichage apres tri\n");
printf("\n");

// affichage apres tri
for(i=0;i<n;i++)
{
    printf("%d \t",* (p+i));
}
}

return 0;
}

```



```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP1\EXO2\EXODEUX.exe
donner le nombre n des entiers
n=5
affichage avant tri
41      18467    6334    26500    19169
affichage apres tri
41      6334     18467   19169    26500
-----
Process exited after 2.153 seconds with return value 0
Press any key to continue . . .
```

2- À chaque itération, il y a exactement $n-1$ comparaisons et au plus $n-1$ permutations. Le pire cas (n itérations) est atteint lorsque le plus petit élément est à la fin du tableau. La complexité est alors $\Theta(n^2)$.

3-Le résultat obtenu est similaire à celui du cours .

EXERCICE 3 :

On considère un tableau A de n éléments, que l'on suppose trié en ordre croissant. On définit l'algorithme suivant (on supposera que clé est dans le tableau et on recherchera la première occurrence de cette valeur)

```
fonction Cherche_Dich_lt(A : tableau; n, clé : entier) : entier
d ← 1; f ← n; trouve ← faux
répéter
  i ← (d + f) / 2 // Partie entière inférieure de (d + f) / 2
  si A[i] = clé
  alors
    trouve ← vrai
  sinon
    si A[i] < clé alors d ← i + 1 sinon f ← i - 1 fin si
  fin si
  jusqu'à trouve
  retourner(i)
```

1. Implémentez et testez cet algorithme sur le tableau suivant avec clé = 30.

1 7 8 9 12 15 18 22 30 31

2. On suppose que le tableau A[1..n] contient $n = 2k$ éléments (où k est un entier positif). Combien d'itérations l'algorithme effectuera-t-il au maximum ? En déduire la complexité (en O) de l'algorithme.

1 -

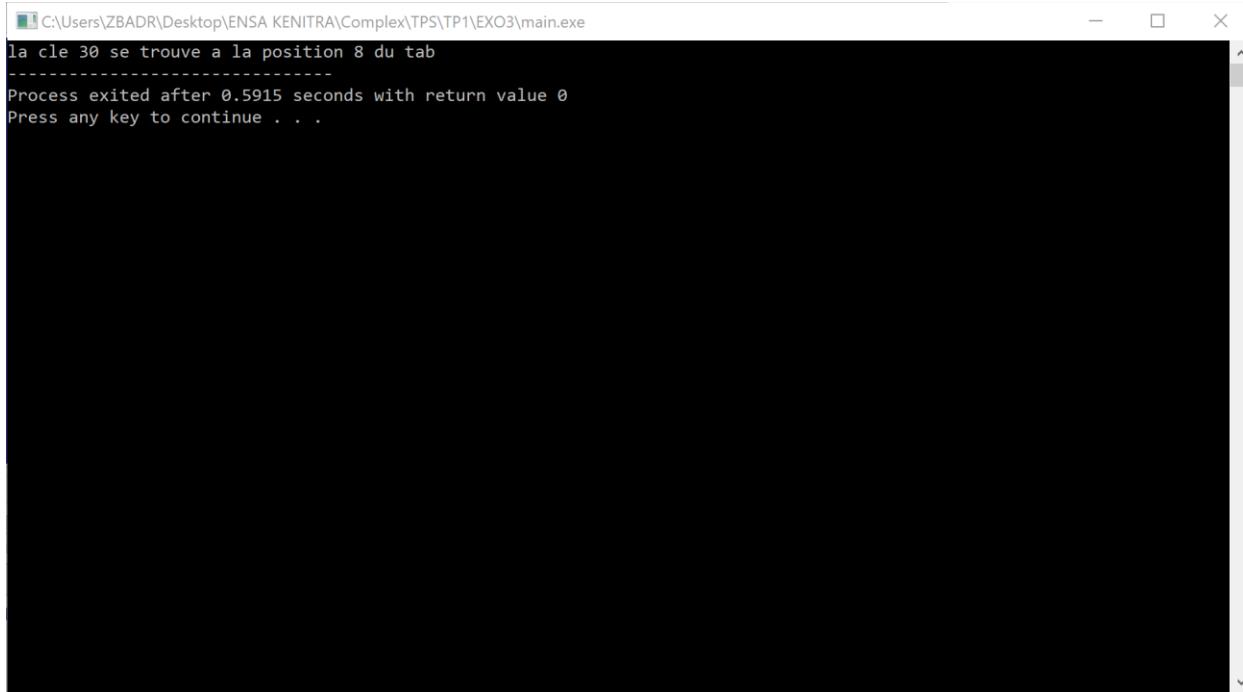
```
#include <stdio.h>
#include <stdlib.h>

/* run this program using the console pauser or add your own getch,
system("pause") or input loop */

//fonction dech
    int chercher_dich(int tab[],int n,int cle)
{
    int i;
    int d=1;
    int f=n;
    int trouve=0;
    do{
        i=((d+f)/2);
        if (tab[i]==cle)
        {
            trouve=1;
        }
        else if(tab[i]<cle)
        {
            d=i+1;
        }
        else {
            f=i-1;
        }

    }while(trouve==0);
    return i;
}
int main(int argc, char *argv[])
{
    int tab[10]={1,7,8,9,12,15,18,22,30,31};
    printf("la cle 30 se trouve a la position %d du tab
",chercher_dich(tab,10,30));

    return 0;
}
```



```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP1\EXO3\main.exe
la cle 30 se trouve a la position 8 du tab
-----
Process exited after 0.5915 seconds with return value 0
Press any key to continue . . .
```

2- Pour pouvoir majorer le nombre maximum d'itérations, si le tableau contient 1 valeurs, et si on a un entier n tel que $1 \leq 2^n$, alors puisque qu'à chaque itération, on sélectionne une moitié de ce qui reste, au bout d'une itération, une moitié de tableau aura au plus $2^n / 2 = 2^{n-1}$ éléments, un quart aura au plus 2^{n-2} et au bout de k itérations, la taille de ce qui reste à étudier est de taille au plus 2^{n-k} . En particulier, si l'on fait n itérations, il reste au plus $2^{n-n} = 1$ valeur du tableau à examiner. On est sûr de s'arrêter cette fois-ci. On a donc montré que si l'entier n vérifie $1 \leq 2^n$, alors l'algorithme va effectuer au plus n itérations. La plus petite valeur est obtenue pour $n = \lceil \log_2 l \rceil$.

TP2

EXERCICE1

Écrire une fonction en C qui prend en paramètre : (un tableau de taille NB_MAX, son nombre d'éléments n < NB_MAX, un entier i \leq n, et un entier m). La fonction doit insérer l'élément m en position i dans le tableau (sans supprimer des éléments).

```
#include <stdio.h>
#include <stdlib.h>
#define n 20
/* run this program using the console pauser or add your own getch,
system("pause") or input loop */
void insr(int tab[],int taille,int pos, int m){
    int i;
    for(i=taille;i>=pos;i--) {
        tab[i]=tab[i-1];
    }
    tab[pos-1]=m;
    taille++;

    printf("apres ");
    for(i=0;i<taille;i++) {
        printf("%d\t",tab[i]);
    }
}
int main(int argc, char *argv[]) {

    int tab[n],i,pos,m,taille;
    printf("donner la taille du tab");
    scanf("%d",&taille);
    printf("donner les element");
    for(i=0;i<taille;i++) {
        printf("tab[%d]=",i+1);
        scanf("%d",&tab[i]);
    }
    printf("la val inserer");
    scanf("%d", &m);

    printf("position");
    scanf("%d", &pos);

    if(pos<=0 || pos>taille+1) {
        printf("erreur");
    }
    else {
        insr(tab,taille,pos,m);
    }
}
```

```
        return 0;  
}
```

```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP2\EXO1\ProjetAA.exe  
donner la taille du tab4  
donner les elementtab[1]=9  
tab[2]=5  
tab[3]=3  
tab[4]=7  
la val inserer98  
position4  
apres 9 5      3      98      7  
-----  
Process exited after 21.56 seconds with return value 0  
Press any key to continue . . .
```

Exercice 2

Supprimer un élément de la position i d'un tableau

```
#include <stdio.h>  
#include <stdlib.h>  
  
/* run this program using the console pauser or add your own getch,  
system("pause") or input loop */  
int i;  
void supp (int tab[],int position,int nbr){  
    for (i = position - 1; i < nbr - 1; i++)  
        tab[i] = tab[i+1];  
}  
int main(int argc, char *argv[]){  
    int position, i, nbr,v;  
    int tab[100];  
  
    printf(" Entrez le nombre des éléments dans le tableau : ");  
    scanf("%d", &nbr);  
  
    printf(" Entrez les %d éléments : ", nbr);  
  
    for (i = 0; i < nbr; i++)  
        scanf("%d", &tab[i]);
```

```

printf(" Entrez l'emplacement où vous souhaitez supprimer l'élément: ");
scanf("%d", &position);
if (position >= nbr+1)
    printf("Suppression impossible.\n");
else
{
    supp (tab,position,nbr);
}
for (i = 0; i < nbr - 1; i++)
    printf("%4d", tab[i]);

return 0;
}

```

```

C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP2\EXO2\fct supp elt tab vers 2\main.exe
Entrez le nombre des éléments dans le tableau : 5
Entrez les 5 éléments : 44
7
8
1
Entrez l'emplacement où vous souhaitez supprimer l'élément: 2
44 7 8 1
-----
Process exited after 28.24 seconds with return value 0
Press any key to continue . .

```

Inverser le contenu d'un tableau

```

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

/* run this program using the console pauser or add your own getch,
system("pause") or input loop */
void inverser(int t[],int n){
    int i,temp;
    while(i<n/2)
    {
        temp = t[i];
        //n-1 parce que le tableau commence par 0
        t[i]=t[n-1-i];
        t[n-1-i]=temp;
    }
}

```

```

    i++;
}
}

int main(int argc, char *argv[]) {

    int i,n,temp;

    printf("Taille du tableau ");
    scanf("%d",&n);
    int t[n];

    for(i=0;i<n;i++)
    {
        printf("t[%d]=",i);
        scanf("%d",&t[i]);
    }
    i=0;
    //
    inverser(t,n);
    printf("\nTableau inverse: \n");
    for(i=0;i<n;i++)
    {
        printf("\nt[%d]=%d",i,t[i]);
    }

    return 0;
}

```

```

C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP2\EXO2\Inverser le contenu d'un tableau\Projet1.exe
Taille du tableau 6
t[0]=99
t[1]=23
t[2]=5
t[3]=00
t[4]=2
t[5]=499

Tableau inverse:

t[0]=499
t[1]=2
t[2]=0
t[3]=5
t[4]=23
t[5]=99
-----
Process exited after 13.76 seconds with return value 0
Press any key to continue . .

```

Fusionner deux tableaux triés de taille différente

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

/* run this program using the console pauser or add your own getch,
system("pause") or input loop */
void fusion(int n1,int n2,int tab3[],int tab1[],int tab2[]){
    int i;
    for(i=0;i<n1;i++){
        tab3[i]=tab1[i];
    }
    for(i=0;i<n2;i++){
        tab3[i+n1]=tab2[i];
    }

}

int main(int argc, char *argv[]) {
    int n1,n2,n3,i;
    printf("donner le nombre n 1 tab1 \n");
    printf("n1=");
    scanf("%d",&n1);
    int tab1[n1];
    for(i=0;i<n1;i++){
        printf("T[%d]=",i);
        scanf("%d",&tab1[i]);
    }

    int ech=0;
    do{
        int ech=0;
        for(i=0;i<n1-1;i++)
        {
            if(tab1[i]>tab1[i+1])
            {
                int tmp;
                tmp=tab1[i];
                tab1[i]=tab1[i+1];
                tab1[i+1]=tmp;
                ech++;
            }
        } while(ech>0);

        for(i=0;i<n1;i++)
    {
        printf("%d \t",tab1[i]);
    }
    printf("\n");
}
```

```

printf("donner le nombre n 2 tab1 \n");
printf("n2=");
scanf("%d",&n2);
int tab2[n2];
for(i=0;i<n2;i++)
{
    printf("T[%d]=",i);
    scanf("%d",&tab2[i]);
}

do{
    int ech=0;
    for(i=0;i<n2-1;i++)
    {
        if(tab2[i]>tab2[i+1])
        {
            int tmp;
            tmp=tab2[i];
            tab2[i]=tab2[i+1];
            tab2[i+1]=tmp;
            ech++;
        }
    }
} while(ech>0);

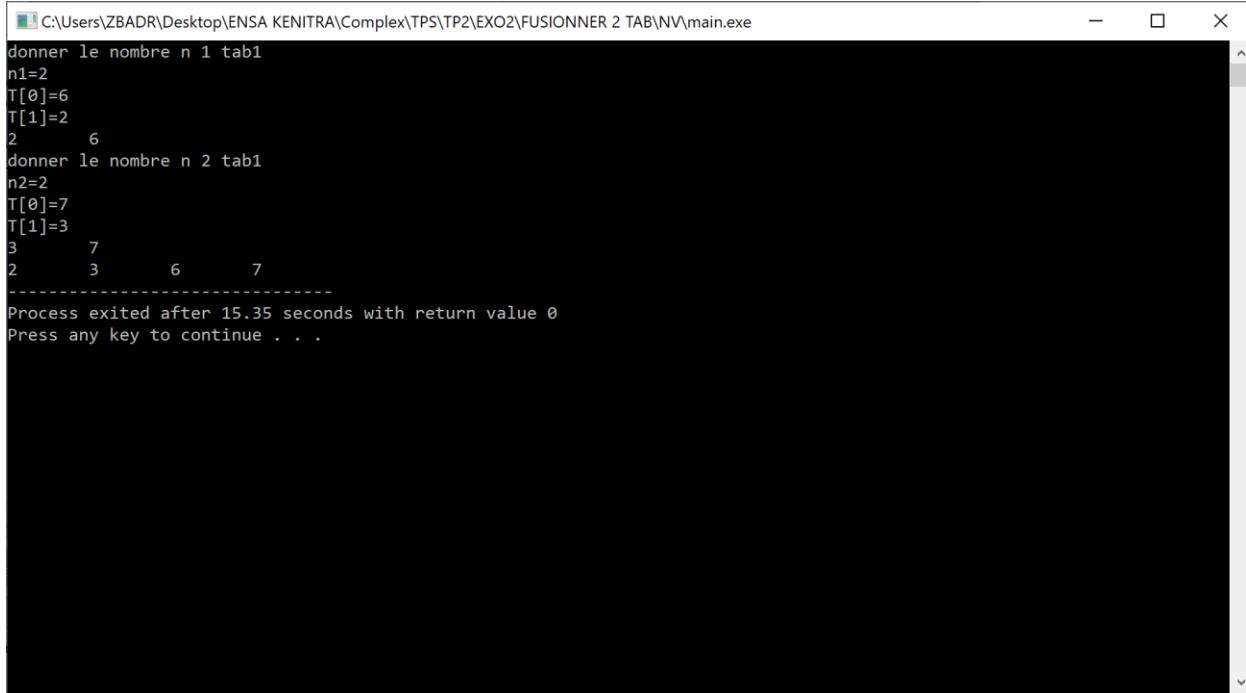
for(i=0;i<n2;i++)
{
    printf("%d \t",tab2[i]);
}
printf("\n");

n3=n1+n2;
int tab3[n3];
fusion(n1,n2,tab3,tab1,tab2);
do{
    int ech=0;
    for(i=0;i<n3-1;i++)
    {
        if(tab3[i]>tab3[i+1])
        {
            int tmp;
            tmp=tab3[i];
            tab3[i]=tab3[i+1];
            tab3[i+1]=tmp;
            ech++;
        }
    }
} while(ech>0);

for(i=0;i<n3;i++)
{
    printf("%d \t",tab3[i]);
}

```

```
    return 0;  
}
```



```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP2\EXO2\FUSIONNER 2 TAB\NV\main.exe  
donner le nombre n 1 tab1  
n1=2  
T[0]=6  
T[1]=2  
2      6  
donner le nombre n 2 tab1  
n2=2  
T[0]=7  
T[1]=3  
3      7  
2      3      6      7  
-----  
Process exited after 15.35 seconds with return value 0  
Press any key to continue . . .
```

Créer une matrice avec des valeurs aléatoires

```
#include <stdio.h>  
#include <stdlib.h>  
  
/* run this program using the console pauser or add your own getch,  
system("pause") or input loop */  
  
void remplissage(int n,int m){  
    int i,j,matrice[n][m];  
    for(i=0;i<n;i++)  
        {            for(j=0;j<m;j++)  
        {  
            matrice[i][j]=rand(); } } }  
  
int main(int argc, char *argv[]) {  
    int n,m,i,j;  
    printf("donner n lignes");  
    scanf("%d",&n);  
    printf("donner m colonnes");  
    scanf("%d",&m);
```

```

        int *mat=malloc(sizeof(int [n][m]));
//      int mat[n][m];
      remplissage(n,m);

      for(i=0;i<n;i++)
      {
        for(j=0;j<m;j++)
        {
          printf("\n matrice[%d] [%d]= %d",i,j,(mat+i,mat+j));
        }
      }

      return 0;
}

```

- Afficher le contenu de la matrice

```

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

/* run this program using the console pauser or add your own getch,
system("pause") or input loop */

void remplissage(int n,int m) {

```

```

int i,j,matrice[n][m];
    for(i=0;i<n;i++)
        {
            for(j=0;j<m;j++)
            {

matrice[i][j]=rand(); }
        }

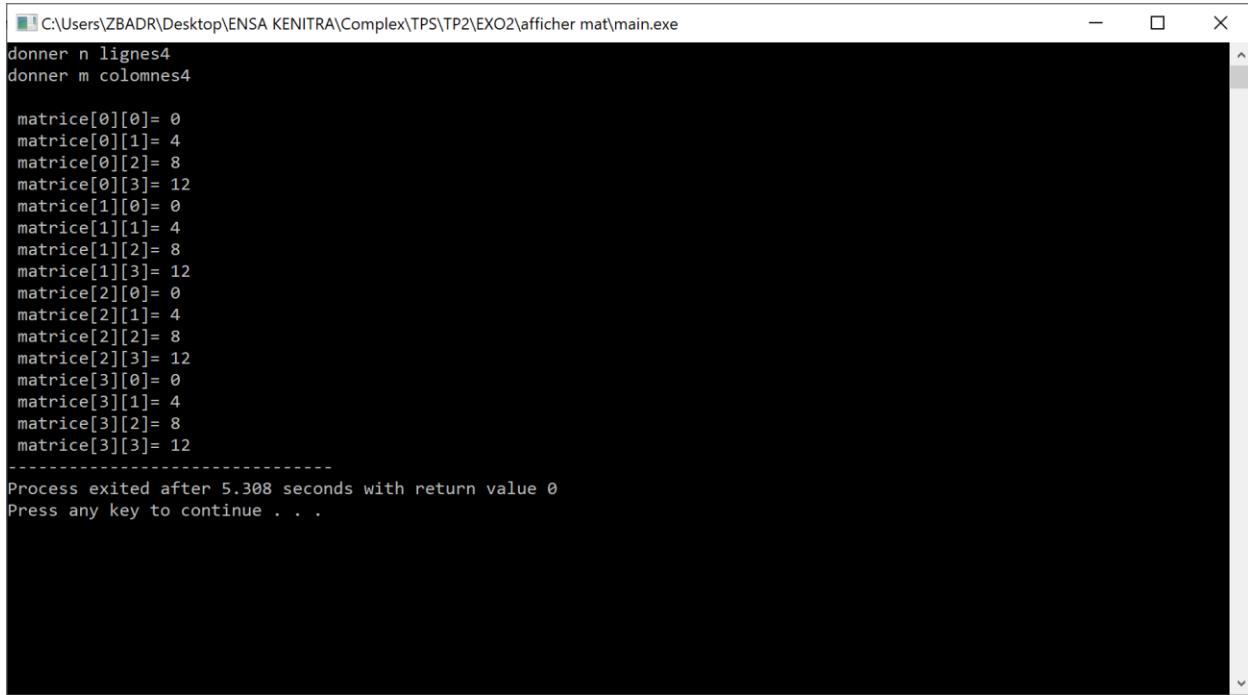
void affichermat(int n, int m){
    int i,j,*mat;
    for(i=0;i<n;i++)
    {
        for(j=0;j<m;j++)
        {
            printf("\n matrice[%d] [%d]= %d",i,j,(mat+i,mat+j));
        }
    }
}

int main(int argc, char *argv[]) {
    int n,m,i,j;
    printf("donner n lignes");
    scanf("%d",&n);
    printf("donner m colonnes");
    scanf("%d",&m);
    int *mat=malloc(sizeof(int [n][m]));
//    int mat[n][m];
    remplissage(n,m);

    affichermat(n,m);

    return 0;
}

```



```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\EXO2\afficher mat\main.exe
donner n lignes4
donner m colonnes4

matrice[0][0]= 0
matrice[0][1]= 4
matrice[0][2]= 8
matrice[0][3]= 12
matrice[1][0]= 0
matrice[1][1]= 4
matrice[1][2]= 8
matrice[1][3]= 12
matrice[2][0]= 0
matrice[2][1]= 4
matrice[2][2]= 8
matrice[2][3]= 12
matrice[3][0]= 0
matrice[3][1]= 4
matrice[3][2]= 8
matrice[3][3]= 12
-----
Process exited after 5.308 seconds with return value 0
Press any key to continue . . .
```

Additionner deux matrices m1 et m2

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

/* run this program using the console pauser or add your own getch,
system("pause") or input loop */
void    addmat(int l,int c,int m1[l][c],int m2[l][c],int m3[l][c]){
    int i,j;
    for(i=0;i<l;i++)
    {
        for(j=0;j<c;j++)
        {
            m3[i][j]=m1[i][j]+m2[i][j];
        }
    }
}

int main(int argc, char *argv[]) {
    int l,c,i,j;
    printf("donner l=");
    scanf("%d",&l);
    printf("donner c=");
    scanf("%d",&c);
}
```

```

printf("\n Matrice 1 \n");
int m1[l][c];
for(i=0; i < l; ++i){
    for(j = 0; j < c; ++j){
        printf("T[%d] [%d]=",i,j);
        scanf("%d",&m1[i][j]);
    }
}
printf("\n Matrice 2 \n");
int m2[l][c];
for(i=0; i < l; ++i){
    for(j = 0; j < c; ++j){
        printf("T[%d] [%d]=",i,j);
        scanf("%d",&m2[i][j]);
    }
}

printf("\n Matrice 1 + Matrice 2 \n");
int m3[l][c];
addmat(l,c,m1,m2,m3);
for(i=0;i<l;i++)
{
    for(j=0;j<c;j++)
    {
        printf("%d ",m3[i][j]);
    }
}

return 0;
}

```

```

C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP2\EXO2\add 2 mat\test\Projet1.exe
donner l=2
donner c=2

Matrice 1
T[0][0]=2
T[0][1]=2
T[1][0]=2
T[1][1]=2

Matrice 2
T[0][0]=2
T[0][1]=2
T[1][0]=2
T[1][1]=2

Matrice 1 + Matrice 2
4 4 4 4
-----
Process exited after 7.221 seconds with return value 0
Press any key to continue . . .

```

Multiplier deux matrices m1 et m2 :

```

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

/* run this program using the console pauser or add your own getch,
system("pause") or input loop */
int l1,c1,c2,l2;
void multimat(int l1,int c2,int l2,int c1,int m1[l1][c1],int m2[l2][c2],int m3[l1][c2]){
    int i,j,k;

    for(i=0;i<l1;i++) {
        for(j=0;j<c2;j++) {
            m3[i][j]=0;
            for(k=0;k<c1;k++) {
                m3[i][j]+=m1[i][k]*m2[i][k];
            }
        }
    }

    int main(int argc, char *argv[]) {
    int l1,c1,c2,l2,i,j;
    printf("donner l1=");
    scanf("%d",&l1);

```

```

printf("donner c1=");
scanf("%d",&c1);

printf("\n Matrice 1 \n");
int m1[11][c1];
for(i=0; i < 11; ++i){
for(j = 0; j < c1; ++j){
    printf("T[%d] [%d]=",i,j);
    scanf("%d",&m1[i][j]);

}
}

//-----

printf("donner l2=");
scanf("%d",&l2);

printf("donner c2=");
scanf("%d",&c2);

if(c1!=l2){
    printf("-----");
    return 0;

}

printf("\n Matrice 2 \n");
int m2[12][c2];
for(i=0; i < 12; ++i){
for(j = 0; j < c2; ++j){
    printf("T[%d] [%d]=",i,j);
    scanf("%d",&m2[i][j]);
}
}

printf("\n Matrice 1 * Matrice 2 \n");
int m3[11][c2];
multimat(l1,c2,l2,c1,m1,m2,m3);
int k;

    for(i = 0; i < 11; i++)
{
    for(j = 0; j < c2; j++)
{
    printf("%d\t",m3[i][j]);
}
}

```

```

        }

    return 0;
}

```

```

C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP2\EXO2\multpl\main.exe
donner l1=2
donner c1=2

Matrice 1
T[0][0]=2
T[0][1]=2
T[1][0]=2
T[1][1]=2
donner l2=2
donner c2=2

Matrice 2
T[0][0]=2
T[0][1]=2
T[1][0]=2
T[1][1]=2

Matrice 1 * Matrice 2
8     8     8     8
-----
Process exited after 7.945 seconds with return value 0
Press any key to continue . . .

```

Exercice 3 :

Soit la suite $u_0 = 1$ et $u_n + 1 = 3u_{n-2} + 2u_{n-1}$

- 1) Écrire une fonction qui prend en paramètre un entier n et qui retourne un tableau contenant les n premiers termes de la suite u_n . La fonction doit marcher quel que soit l'entier n rentré.
- 2) Écrire le programme principal qui saisit l'entier n et affiche les n premiers termes de la suite u_n en utilisant la fonction définie au 1).

```

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

/* run this program using the console pauser or add your own getch,
system("pause") or input loop */

void suite(int n {

```

```

int k[n],i;
printf("%d",k[0]=0);
for (i=0;i<n;i++){
    k[i+1]=3*k[i]*k[i]+2*k[i]+1;
    printf("%d\n",k[i+1]);
}

}

int main(int argc, char *argv[]) {
    int n;
    printf("donner n= ");
    scanf("%d",&n);
    suite(n);

    return 0;
}

```

C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP2\EXO3\ProjetU.exe

donner n= 3

01

6

121

Process exited after 1.367 seconds with return value 0

Press any key to continue . . .

Exercice 4 :

Implémentez une fonction :

char * multichar(char *s, int n);

qui prend en argument une chaîne de caractères et renvoie une deuxième chaîne où l'ordre des lettres est inversé et chaque lettre est répétée n fois. Par exemple, si la fonction reçoit "ENSA" et 3 en arguments, elle renverra "AAASSSNNNEEE".

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

char *multichar(char *s, int n){
int i,j;
int l=strlen(s);
char *NVS;
NVS=(char*)malloc(n*l*sizeof(char));
for(i=0;i<l;i++)
{
for(j=0;j<n;j++)
{
NVS[n*i+j]=s[l-1-i];
}
}
NVS[n*l]='\0';
return NVS;
}

int main()
{
int n;
char *s;
s=(char*)malloc(sizeof(s));
printf(" Saisissez une chaine de caracteres: ");
scanf("%s",s);
printf(" Saisissez un nombre de repetition de caractere: ");
scanf("%d",&n);
printf(" %s",multichar(s,n));
}
```

C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP2\EXO4\NV\main.exe

Saisissez une chaine de caracteres: ENSA
Saisissez un nombre de repetition de caractere: 3
AAASSNNNEEE

Process exited after 3.544 seconds with return value 13
Press any key to continue . . .

Tp3

Créer dans un nouveau projet (Fichier->nouveau->projet->console application). Puis, ajoutez et

implémentez dans ce projet les trois fichiers suivants :

- liste.h : contient la déclaration de la structure liste et les prototypes des fonctions de gestion d'une liste.
- liste.c : contient le corps des fonctions dont le prototype est défini dans liste.h.
- mainliste.c : programme principal des listes. Il propose le menu suivant :

```
int menu () {
    printf ("\n\n GESTION D'UNE LISTE D'ENTIERS \n\n");

    printf ("1 - Créer une liste\n");
    printf ("2 - Insertion en tête de liste \n");
    printf ("3 - Insertion en fin de liste \n");
    printf ("4 - Retrait en tête de liste \n");
    printf ("5 - Retrait en fin de liste \n");
    printf ("6 - Retrait d'un objet à partir de sa référence\n");
    printf ("7 - Afficher les objets de la liste \n");
    printf ("8 - Chercher Un objet \n");
    printf ("9 - Destruction de la liste \n");
    printf ("10 - Fin\n");
    printf ("\n");
    printf ("Votre choix ? ");
    int cod; scanf ("%d", &cod); getchar();
    printf ("\n");
    return cod;
}
```

liste.h

```
#define faux 0
#define vrai 1
#define NONORDONNE 0
#define CROISSANT 1
#define DECROISSANT 2

//#include <stdbool.h>

typedef void objet;
typedef int boolean;

//un element de la liste
typedef struct element {
    objet* reference ;//reference un objet
    struct element* suivant; //element suivant de la liste
}Element;

//le type listes
typedef struct {
    Element* premier;//premier elemnt de la liste
    Element* dernier;
    Element* courant;
    int nbElt;
    int type;
    char* (*afficher) (objet* );
    int (*comparer) (objet*,objet* );
}Liste;

///////////////////////////////
int comparer (objet* objet1, objet* objet2);
char* afficher(objet* objet);

///////////////////////////////

//initialiser la liste pointee pa li
void initListe(Liste* li,int
type,char*(*afficher) (objet* ),int(*comparer) (objet*,objet*)) ;

//CREATION DUNE LISTE

Liste* creerListe(int type,char* (*afficher) (objet* ),int
(*comparer) (objet*,objet*));
```

```

//boolean listevide(liste* li);
boolean listevide(Liste* li);

int nbElement(Liste* li);

//AJOUT EN TETE DE LISTE
static Element* creeElement();

void insererEnTeteDeListe(Liste* li,objet* objet);

//ajout apres lelement precedent
// insert dans la liste li,objet apres precedent
//si precedent est NULL,inserer en tete de liste
static void insererApres(Liste*li,Element *precedent,objet* objet);
//ajout en fin de liste
void insererEnFinDeListe(Liste *li,void *objet);

//se positionner sur le premier element de la liste li
void ouvrirListe(Liste* li);

boolean finListe(Liste* li);

//courant elemnt
static Element *elementCourant (Liste *li);

//objet courant
objet* objetCourant (Liste *li);

void listerListe(Liste *li);

//chercher un objet
objet* chercherUnobjet(Liste *li,objet *objetCherche);

```

```
//retrait en tete de liste
objet* extraireEnTeteDeListe (Liste *li);
//retrait de elt qui suit elet precedent

static objet* extraireApres(Liste *li, Element *precedent);

// retrait de lobjet en fin de liste
objet* extraireEnFinDeListe(Liste *li );

//retrait dun objet a partir de sa reference
booleen extraireUnobjet (Liste *li,objet *objet);

void detruireListe(Liste *li);
```

liste.c

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

#define faux  0
#define vrai 1
typedef void objet;
typedef int booleen;

//initialiser la liste pointee pa li
void initListe (Liste * li,int type,char*
(*afficher) (objet*),int (*comparer) (objet*,objet*)){

    li->premier=NULL;
    li->dernier=NULL;
    li->courant=NULL;
    li->nbElt=0;
    li->type=type;
    li->afficher=afficher;
    li->comparer=comparer;
}

//CREATION DUNE LISTE

Liste* creerListe(int type,char* (*afficher) (objet*),int
(*comparer) (objet*,objet*)){

    Liste* li=(Liste*)malloc(sizeof(Liste));
    initListe(li,type,afficher,comparer);
    return li;
}

// vérifier si la liste est vide
booleen listevide(Liste* li){
return li->nbElt == 0;
}

int nbElement(Liste * li){
    return li->nbElt;
}
```

```

//AJOUT EN TETE DE LISTE
static Element* creerElement(){
    return (Element *)malloc(sizeof(Element));
}

void insererEnTeteDeListe(Liste * li,objet * objet){
    Element * nouveau=creerElement();
    nouveau->reference=objet;
    nouveau->suivant=li->premier;
    li->premier=nouveau;
    if(li->dernier==NULL) li->dernier=nouveau;
    li->nbElt++;
}

//ajout apres lelement precedent
// insert dans la liste li,objet apres precedent
//si precedent est NULL,inserer en tet de liste
static void insererApres(Liste *li,Element *precedent,objet* objet){
    if(precedent==NULL){
        insererEnTeteDeListe(li,objet);
    }
    else{
        Element* nouveau=creerElement();
        nouveau->reference=objet;
        nouveau->suivant=precedent->suivant;
        precedent->suivant=nouveau;
        if(precedent==li->dernier) li->dernier=nouveau;
        li->nbElt++;
    }
}
//ajout en fin de liste
void insererEnFinDeListe(Liste *li,void *obj){
    insererApres(li,li->dernier,obj);
}

//se positionner sur le premier element de la liste li
void ouvrirListe(Liste * li){
    li->courant=li->premier;
}

booleen finListe(Liste *li){
    return li->courant==NULL;
}

```

```

//courant elemt
static Element* elementCourant (Liste* li){
    Element *ptc=li->courant;
    if(li->courant != NULL){
        li->courant=li->courant->sivant;
    }
    return ptc;
}

//objet courant
objet* objetCourant (Liste *li){
    Element *ptc = elementCourant(li);
    return ptc==NULL ? NULL : ptc->reference;
}

//lister liste(liste* li)
void listerListe(Liste* li){
    ouvrirListe(li);
    while(!finListe(li)){
        objet* objet=objetCourant(li);
        printf("%s\n",li->afficher(objet));
    }
}

//chercher un objet
objet* chercherUnobjet(Liste* li,objet* objetCherche) {
    booleen trouve = faux;
    objet* objet;
    ouvrirListe(li);
    while(!finListe(li) && !trouve){
        objet=objetCourant(li);
        trouve =li->comparer(objetCherche,objet)==0;
    }
    return trouve ? objet:NULL;
}

//retrait en tete de liste
objet* extraireEnTeteDeListe (Liste* li){
    Element* extrait=li->premier;
    if(!listeVide(li)){
        li->premier=li->premier->sivant;
        if(li->premier==NULL) li->dernier=NULL;
    }
}

```

```

        li->nbElt--;

    }

    return extrait !=NULL ? extrait->reference : NULL;
}

//retrait de elt qui suit elet precedent

static objet* extraireApres(Liste* li, Element *precedent){
    if(precedent==NULL){
        return extraireEnTeteDeListe(li);
    }else{
        Element *extrait=precedent->suivant;
        if(extrait != NULL){
            precedent->suivant=extrait->suivant;
            if(extrait==li->dernier) li->dernier=precedent;
            li->nbElt--;
        }
        return extrait !=NULL ? extrait->reference :NULL;
    }
}

// retrait de lobjet en fin de liste
objet* extraireEnFinDeListe(Liste *li ){
    objet *extrait;
    if(listeVide(li)){
        extract=NULL;
    }else if (li->premier==li->dernier){
        extract = extraireEnTeteDeListe(li);
    }else {
        Element *ptc = li->premier;
        while(ptc->suivant !=li->dernier) ptc=ptc->suivant;
        extract=extraireApres (li,ptc);
    }
    return extract;
}

//retrait dun objet a partir de sa reference
booleen extraireUnobjet (Liste *li,objet *objet){
    Element* precedent=NULL;
    Element* ptc =NULL;
    booleen trouve = faux;

    ouvrirListe(li);
    while(!finListe(li) && !trouve){
        precedent =ptc;
        ptc=elementCourant(li);
        trouve=(ptc->reference==objet) ? vrai:faux;
    }
}

```

```
    if(!trouve)  return faux;
    void* extrait = extraireApres(li,precedent);
    return vrai;
}

void detruireListe(Liste *li){
    ouvrirListe(li);
    while(!finListe(li)){
        Element *ptc =elementCourant(li);
        free(ptc);
    }
    initListe(li,0,NULL,NULL);
}
///////////
int comparer (objet* objet1, objet* objet2){
return strcmp ((char*)objet1,(char*)objet2);
}

char* afficher (objet* objet){
return (char*)objet;
}
///////////
///////////
```

mainliste.c

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

///////////
/////
int comparer (objet* objet1, objet* objet2){
    return strcmp ((char*)objet1,(char*)objet2);
}

char* afficher (objet* objet){
    return (char*)objet;
}

char * ecrireEntier(void* obj){
    int *entier = (int*) obj;
    char * output = (char*) malloc (sizeof(int));
    sprintf(output,sizeof(int),"%d \n", *entier);
    return output;
}

int comparerInt(void *obj1, void *obj2){
    int *b =(int*) obj2;
    int *a =(int*) obj1;
    if(*a>*b) return 1;
    else if(*a==*b) return 0;
    else return -1;
}

Liste* creerListe(int type,char* (*afficher)(objet*),int
(*comparer)(objet*,objet*)) {
    Liste* li=(Liste*)malloc(sizeof(Liste));
    initListe(li,type,afficher,comparer);
    return li;
}

void initListe (Liste * li,int type,char*
(*afficher)(objet*),int(*comparer)(objet*,objet*)) {
    li->premier=NULL;
    li->dernier=NULL;
    li->courant=NULL;
    li->nbElt=0;
    li->type=type;
    li->afficher=afficher;
    li->comparer=comparer;
}

static Element* creerElement(){
    return (Element *)malloc(sizeof(Element));
}

void insererEnTeteDeListe(Liste * li,objet * objet){
    Element * nouveau=creerElement();
    nouveau->reference=objet;
```

```

nouveau->suivant=li->premier;
li->premier=nouveau;
if(li->dernier==NULL) li->dernier=nouveau;
li->nbElt++;
}

static void insererApres(Liste *li,Element *precedent,objet* objet){
    if(precedent==NULL){
        insererEnTeteDeListe(li,objet);
    }
    else{
        Element* nouveau=creerElement();
        nouveau->reference=objet;
        nouveau->suivant=precedent->suivant;
        precedent->suivant=nouveau;
        if(precedent==li->dernier) li->dernier=nouveau;
        li->nbElt++;
    }
}

void insererEnFinDeListe(Liste *li,void *obj){
    insererApres(li,li->dernier,obj);
}

objet* extraireEnTeteDeListe (Liste* li){
    Element* extrait=li->premier;
    if(!listevide(li)){
        li->premier=li->premier->suivant;
        if(li->premier==NULL) li->dernier=NULL;
        li->nbElt--;
    }
    return extrait !=NULL ? extrait->reference : NULL;
}

booleen listevide(Liste* li){
return li-> nbElt == 0;
}

static void * extraireApres(Liste *li,Element *precedent){
    if (precedent ==NULL){
        return extraireEnTeteDeListe(li);
    }
    else{
        Element *extrait=precedent->suivant;
        if (extrait!=NULL){
            precedent->suivant=extrait->suivant;
            if(extrait==li->dernier) li->dernier=precedent;
            li->nbElt--;
        }
        return extrait != NULL? extrait->reference :NULL;
    }
}

```

```

objet* extraireEnFinDeListe(Liste *li ){
    objet *extrait;
    if(listevide(li)){
        extract=NULL;
    }else if (li->premier==li->dernier){
        extract = extraireEnTeteDeListe(li);
    }else {
        Element *ptc = li->premier;
        while(ptc->suivant !=li->dernier) ptc=ptc->suivant;
        extract = extraireApres(li,ptc);
    }
    return extract;
}

booleen extraireUnobjet (Liste *li,objet *objet){
    Element* precedent=NULL;
    Element* ptc =NULL;
    booleen trouve = faux;

    ouvrirListe(li);
    while(!finListe(li) && !trouve){
        precedent =ptc;
        ptc=elementCourant(li);
        trouve=(ptc->reference==objet) ? vrai:faux;
    }
    if(!trouve) return faux;
    void* extract = extraireApres(li,precedent);
    return vrai;
}
static Element* elementCourant (Liste* li){
    Element *ptc=li->courant;
    if(li->courant != NULL){
        li->courant=li->courant->suivant;
    }
    return ptc;
}
void ouvrirListe(Liste * li){
    li->courant=li->premier;
}
booleen finListe(Liste *li){
    return li->courant==NULL;
}
void listerListe(Liste* li){
    ouvrirListe(li);
    while(!finListe(li)){
        objet* objet=objetCourant(li);
        printf("%s\n",li->afficher(objet));
    }
}

objet* objetCourant (Liste *li){
    Element *ptc = elementCourant(li);
    return ptc==NULL ? NULL : ptc->reference;
}

```

```

}

objet* chercherUnobjet(Liste* li,objet* objetCherche) {
    booleen trouve = faux;
    objet* objet;
    ouvrirListe(li);
    while(!finListe(li) && !trouve){
        objet=objetCourant(li);
        trouve =li->comparer(objetCherche,objet)==0;
    }
    return trouve ? objet:NULL;
}

void detruireListe(Liste *li){
    ouvrirListe(li);
    while(!finListe(li)){
        Element *ptc =elementCourant(li);
        free(ptc);
    }
    initListe(li,0,NULL,NULL);
}

///////////////////////////////
///////
int menu () {
    printf ("\n\n GESTION D UNE LISTE D ENTIERS \n\n");
    printf ("1 - Creer une liste\n");
    printf ("2 - Insertion en tete de liste \n");
    printf ("3 - Insertion en fin de liste \n");
    printf ("4 - Retrait en tete de liste \n");
    printf ("5 - Retrait en fin de liste \n");
    printf ("6 - Retrait d un objet à partir de sa reference\n");
    printf ("7 - Afficher les objets de la liste \n");
    printf ("8 - Chercher Un objet \n");
    printf ("9 - Destruction de la liste \n");
    printf ("10 - Fin\n");
    printf ("\n");
    printf ("Votre choix ? ");
    int cod; scanf ("%d", &cod); getchar();
    printf ("\n");
    return cod;
}
int main()
{
    Liste * li=NULL;
    int cod;
    do {
        cod=menu();
        switch(cod){
            case 1:
                li=creerListe(0,afficher,comparer);

                break;

```

```

case 2:

    printf("Tapez l'entier a inserer : ");
    int* itete;
    itete = (int*) malloc(sizeof(int));
    scanf("%d",itete);
    insererEnTeteDeListe(li,itete);
    break;

case 3:

    printf("Tapez l'entier a inserer : ");
    int* ifin;
    ifin = (int*) malloc(sizeof(int));
    scanf("%d",ifin);
    insererEnFinDeListe(li,ifin);
    break;

case 4 :
{
    int* extete;
    extete = (int*)extraireEnTeteDeListe(li);
    printf("l'objet extrait en tete de liste est %d\n",*extete);
    break;
}

case 5:
{
    int* exfin;
    exfin = (int*)extraireEnFinDeListe(li);
    if(exfin == NULL) printf("Liste Vide !");
    else printf("L'element extrait : %d",*exfin);
    break;
}

```

```

    case 6 :

        printf("Tapez le nombre que vous voulez
               extraire : ");
        int objext;
        scanf("%d",&objext);
        if(extraireUnobjet(li,&objext)==vrai)
            printf("Le nombre est extrait !!");
        else printf("L'element n'est pas trouve");
        break;

    case 7 :
        listerListe(li);
        break;

    case 8 :

        printf("Tapez le nombre a chercher : ");
        int* objrech;
        objrech = (int*) malloc(sizeof(int));
        scanf("%d",objrech);
        if(chercherUnobjet(li,objrech)) printf("Le nombre
               est trouve !!");
        else printf("Le nombre n'est pas trouve !!");
        break;

    case 9:

        detruireListe(li);
        printf("Liste est detruite .");
        break;

    }

}while(cod!=10);

}

```

MENU

```
GESTION D UNE LISTE D ENTIERS

1 - Creer une liste
2 - Insertion en tete de liste
3 - Insertion en fin de liste
4 - Retrait en tete de liste
5 - Retrait en fin de liste
6 - Retrait d un objet à partir de sa référence
7 - Afficher les objets de la liste
8 - Chercher Un objet
9 - Destruction de la liste
10 - Fin

Votre choix ?
```

1 – Créer une liste

```
1 - Creer une liste
2 - Insertion en tete de liste
3 - Insertion en fin de liste
4 - Retrait en tete de liste
5 - Retrait en fin de liste
6 - Retrait d un objet à partir de sa référence
7 - Afficher les objets de la liste
8 - Chercher Un objet
9 - Destruction de la liste
10 - Fin

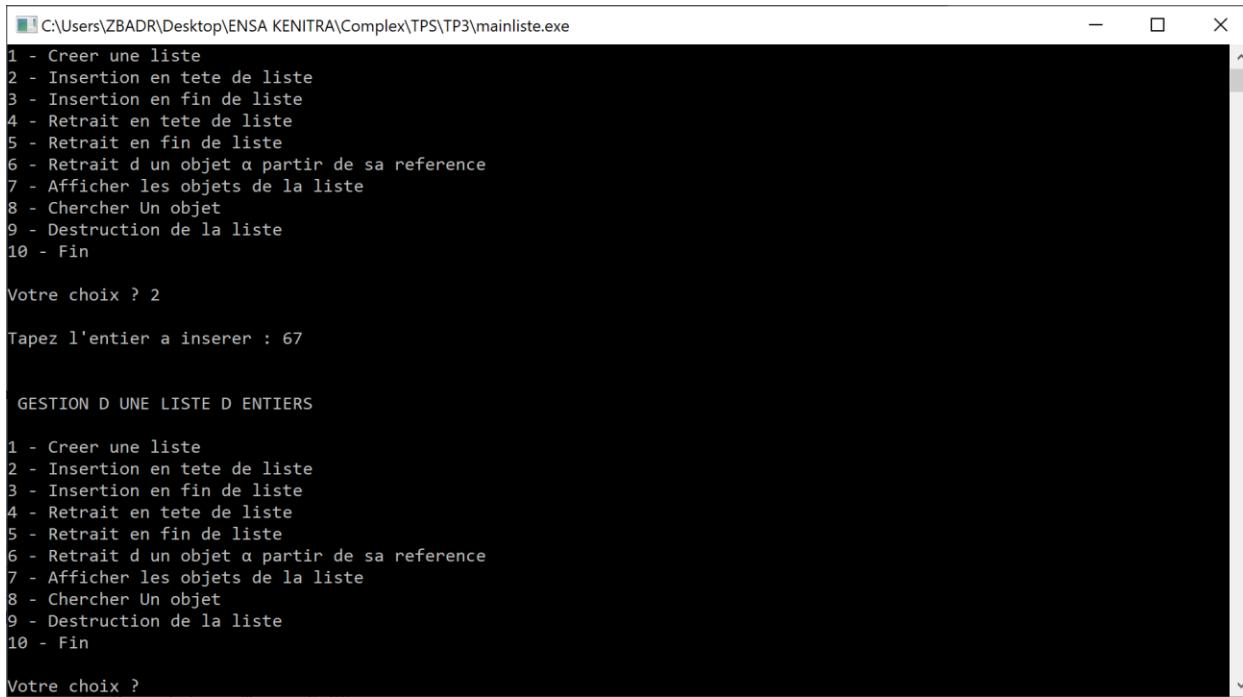
Votre choix ? 1

GESTION D UNE LISTE D ENTIERS

1 - Creer une liste
2 - Insertion en tete de liste
3 - Insertion en fin de liste
4 - Retrait en tete de liste
5 - Retrait en fin de liste
6 - Retrait d un objet à partir de sa référence
7 - Afficher les objets de la liste
8 - Chercher Un objet
9 - Destruction de la liste
10 - Fin

Votre choix ?
```

2 - Insertion en tête de liste



```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP3\mainliste.exe

1 - Creer une liste
2 - Insertion en tete de liste
3 - Insertion en fin de liste
4 - Retrait en tete de liste
5 - Retrait en fin de liste
6 - Retrait d un objet a partir de sa reference
7 - Afficher les objets de la liste
8 - Chercher Un objet
9 - Destruction de la liste
10 - Fin

Votre choix ? 2

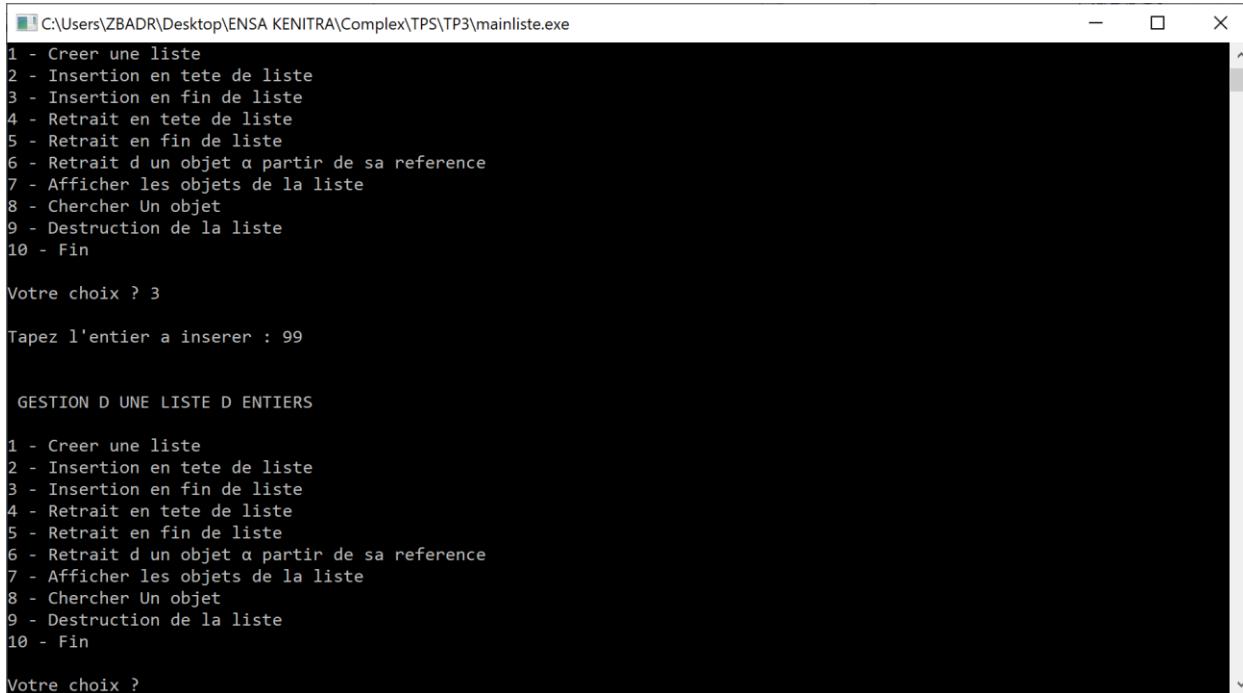
Tapez l'entier a inserer : 67

GESTION D UNE LISTE D ENTIERS

1 - Creer une liste
2 - Insertion en tete de liste
3 - Insertion en fin de liste
4 - Retrait en tete de liste
5 - Retrait en fin de liste
6 - Retrait d un objet a partir de sa reference
7 - Afficher les objets de la liste
8 - Chercher Un objet
9 - Destruction de la liste
10 - Fin

Votre choix ?
```

3 - Insertion en fin de liste



```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP3\mainliste.exe

1 - Creer une liste
2 - Insertion en tete de liste
3 - Insertion en fin de liste
4 - Retrait en tete de liste
5 - Retrait en fin de liste
6 - Retrait d un objet a partir de sa reference
7 - Afficher les objets de la liste
8 - Chercher Un objet
9 - Destruction de la liste
10 - Fin

Votre choix ? 3

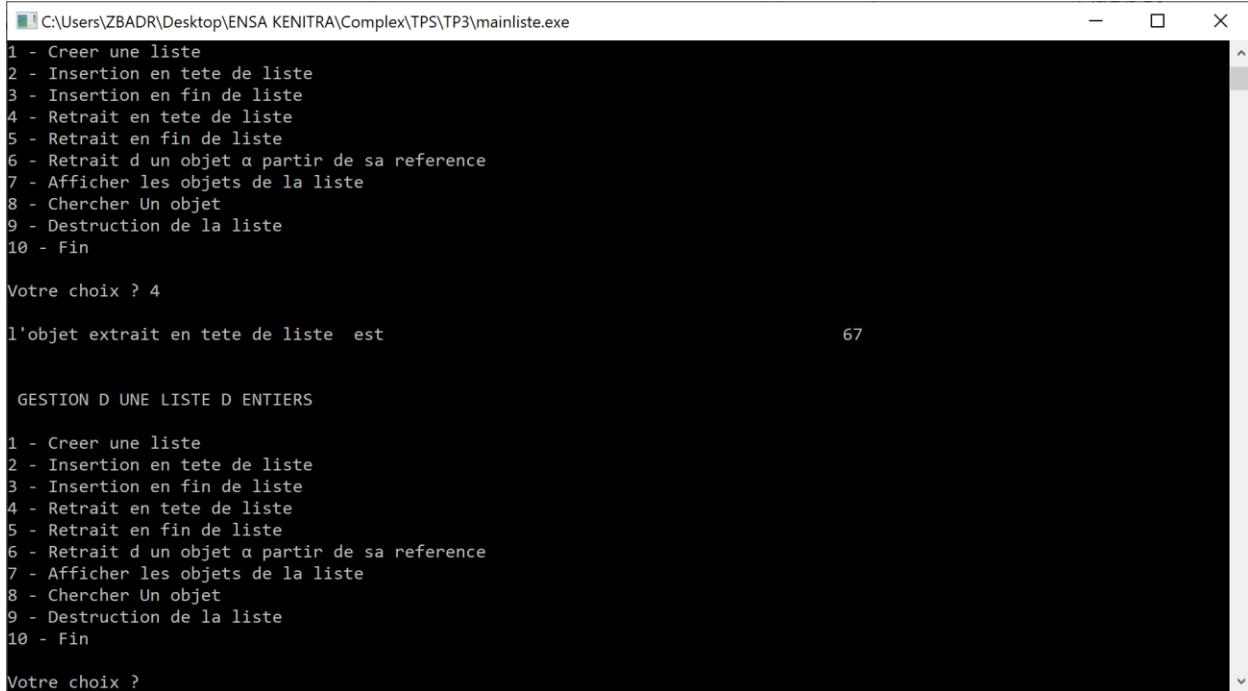
Tapez l'entier a inserer : 99

GESTION D UNE LISTE D ENTIERS

1 - Creer une liste
2 - Insertion en tete de liste
3 - Insertion en fin de liste
4 - Retrait en tete de liste
5 - Retrait en fin de liste
6 - Retrait d un objet a partir de sa reference
7 - Afficher les objets de la liste
8 - Chercher Un objet
9 - Destruction de la liste
10 - Fin

Votre choix ?
```

4 - Retrait en tête de liste



```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP3\mainliste.exe
1 - Creer une liste
2 - Insertion en tete de liste
3 - Insertion en fin de liste
4 - Retrait en tete de liste
5 - Retrait en fin de liste
6 - Retrait d un objet a partir de sa reference
7 - Afficher les objets de la liste
8 - Chercher Un objet
9 - Destruction de la liste
10 - Fin

Votre choix ? 4

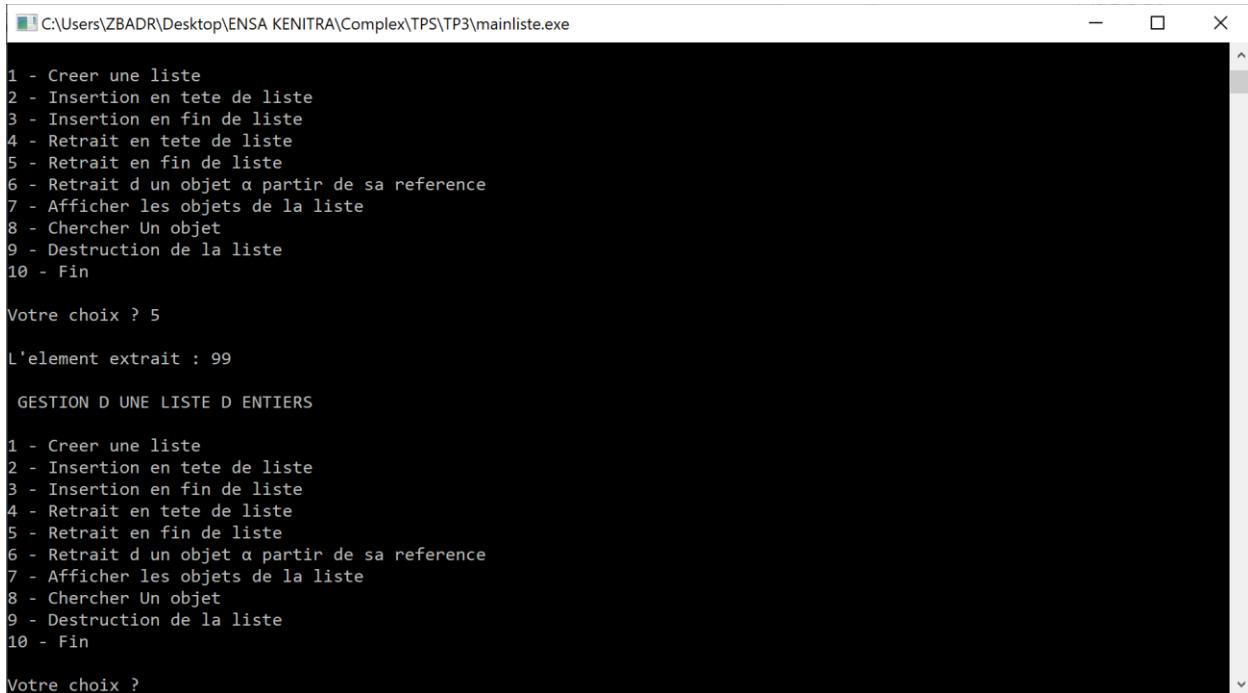
l'objet extrait en tete de liste est 67

GESTION D UNE LISTE D ENTIERS

1 - Creer une liste
2 - Insertion en tete de liste
3 - Insertion en fin de liste
4 - Retrait en tete de liste
5 - Retrait en fin de liste
6 - Retrait d un objet a partir de sa reference
7 - Afficher les objets de la liste
8 - Chercher Un objet
9 - Destruction de la liste
10 - Fin

Votre choix ?
```

5 - Retrait en fin de liste



```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP3\mainliste.exe
1 - Creer une liste
2 - Insertion en tete de liste
3 - Insertion en fin de liste
4 - Retrait en tete de liste
5 - Retrait en fin de liste
6 - Retrait d un objet a partir de sa reference
7 - Afficher les objets de la liste
8 - Chercher Un objet
9 - Destruction de la liste
10 - Fin

Votre choix ? 5

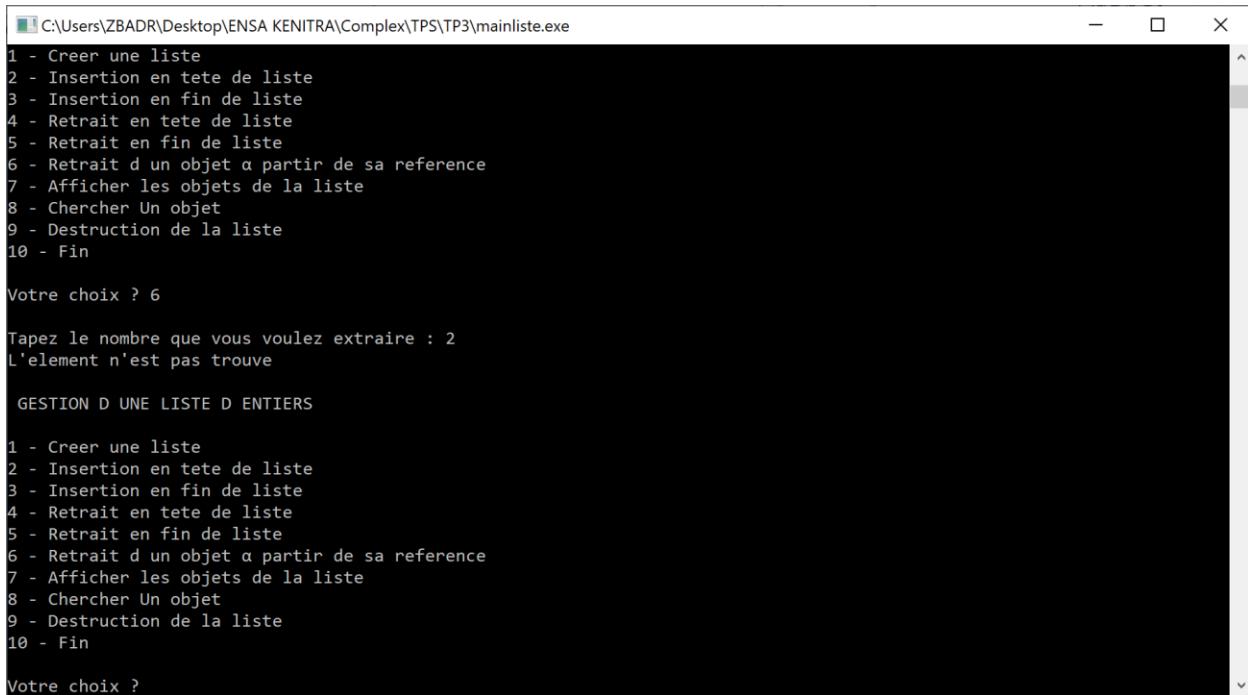
L'element extrait : 99

GESTION D UNE LISTE D ENTIERS

1 - Creer une liste
2 - Insertion en tete de liste
3 - Insertion en fin de liste
4 - Retrait en tete de liste
5 - Retrait en fin de liste
6 - Retrait d un objet a partir de sa reference
7 - Afficher les objets de la liste
8 - Chercher Un objet
9 - Destruction de la liste
10 - Fin

Votre choix ?
```

6 - Retrait d'un objet à partir de sa référence



```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP3\mainliste.exe
1 - Creer une liste
2 - Insertion en tete de liste
3 - Insertion en fin de liste
4 - Retrait en tete de liste
5 - Retrait en fin de liste
6 - Retrait d un objet à partir de sa reference
7 - Afficher les objets de la liste
8 - Chercher Un objet
9 - Destruction de la liste
10 - Fin

Votre choix ? 6

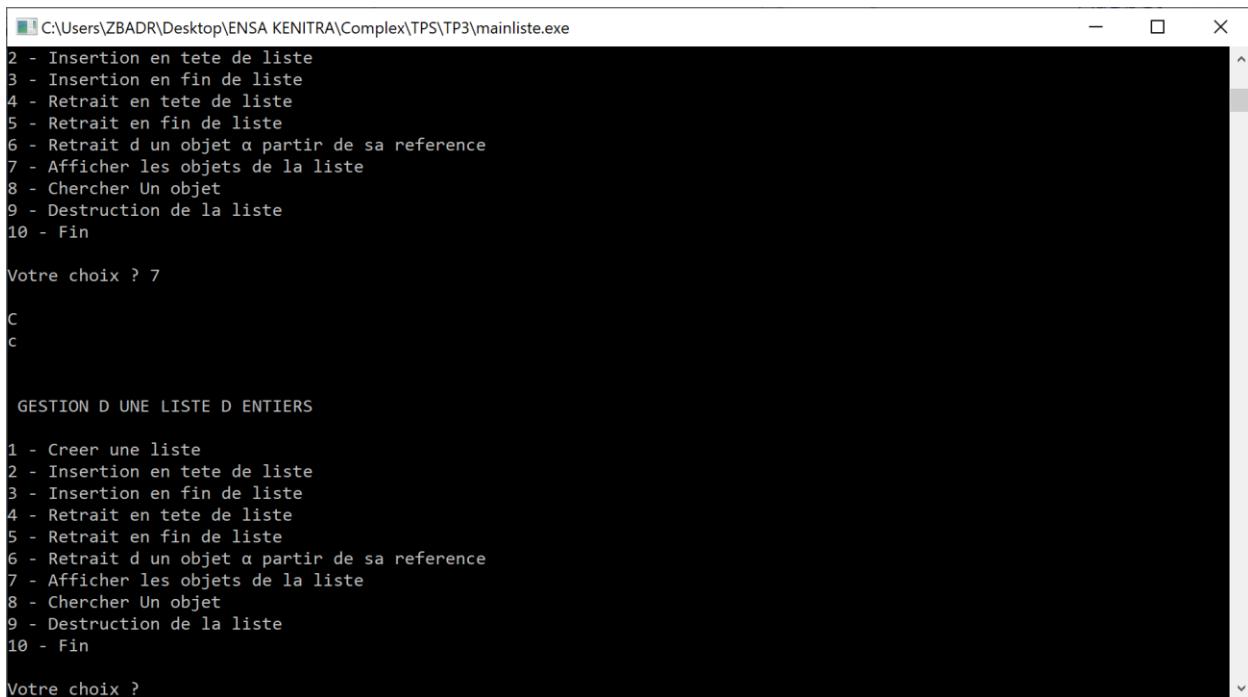
Tapez le nombre que vous voulez extraire : 2
L'element n'est pas trouve.

GESTION D UNE LISTE D ENTIERS

1 - Creer une liste
2 - Insertion en tete de liste
3 - Insertion en fin de liste
4 - Retrait en tete de liste
5 - Retrait en fin de liste
6 - Retrait d un objet à partir de sa reference
7 - Afficher les objets de la liste
8 - Chercher Un objet
9 - Destruction de la liste
10 - Fin

Votre choix ?
```

7 – Afficher les objets de la liste



```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP3\mainliste.exe
2 - Insertion en tete de liste
3 - Insertion en fin de liste
4 - Retrait en tete de liste
5 - Retrait en fin de liste
6 - Retrait d un objet à partir de sa reference
7 - Afficher les objets de la liste
8 - Chercher Un objet
9 - Destruction de la liste
10 - Fin

Votre choix ? 7

C
c

GESTION D UNE LISTE D ENTIERS

1 - Creer une liste
2 - Insertion en tete de liste
3 - Insertion en fin de liste
4 - Retrait en tete de liste
5 - Retrait en fin de liste
6 - Retrait d un objet à partir de sa reference
7 - Afficher les objets de la liste
8 - Chercher Un objet
9 - Destruction de la liste
10 - Fin

Votre choix ?
```

8 – Chercher Un objet

```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP3\mainliste.exe
1 - Creer une liste
2 - Insertion en tete de liste
3 - Insertion en fin de liste
4 - Retrait en tete de liste
5 - Retrait en fin de liste
6 - Retrait d un objet a partir de sa reference
7 - Afficher les objets de la liste
8 - Chercher Un objet
9 - Destruction de la liste
10 - Fin

Votre choix ? 8

Tapez le nombre a chercher : 67
Le nombre est trouve !!

GESTION D UNE LISTE D ENTIERS

1 - Creer une liste
2 - Insertion en tete de liste
3 - Insertion en fin de liste
4 - Retrait en tete de liste
5 - Retrait en fin de liste
6 - Retrait d un objet a partir de sa reference
7 - Afficher les objets de la liste
8 - Chercher Un objet
9 - Destruction de la liste
10 - Fin

Votre choix ?
```

9 - Destruction de la liste

```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP3\mainliste.exe
1 - Creer une liste
2 - Insertion en tete de liste
3 - Insertion en fin de liste
4 - Retrait en tete de liste
5 - Retrait en fin de liste
6 - Retrait d un objet a partir de sa reference
7 - Afficher les objets de la liste
8 - Chercher Un objet
9 - Destruction de la liste
10 - Fin

Votre choix ? 9

Liste est detruite .

GESTION D UNE LISTE D ENTIERS

1 - Creer une liste
2 - Insertion en tete de liste
3 - Insertion en fin de liste
4 - Retrait en tete de liste
5 - Retrait en fin de liste
6 - Retrait d un objet a partir de sa reference
7 - Afficher les objets de la liste
8 - Chercher Un objet
9 - Destruction de la liste
10 - Fin

Votre choix ?
```

TP 4

Exercice 1 :

Créer dans Code::Blocks un nouveau projet (Fichier->nouveau->projet->console application). Puis,

ajoutez les deux fichiers suivants :

- liste.h : contient la déclaration de la structure liste et les prototypes des fonctions de gestion d'une liste.
- liste.c : contient le corps des fonctions dont le prototype est défini dans liste.h.

Implémentez ensuite les trois fichiers suivants :

- personne.h : contient la structure et les prototypes des fonctions de gestion d'une personne.
- personne.c : contient le corps des fonctions dont le prototype est défini dans personne.h.
- listePersonne.c : programme principal qui propose le menu suivant :

```
int menu () {
    printf ("\n\nGESTION D'UNE LISTE DE PERSONNES\n\n");
    printf ("0 - Fin\n");
    printf ("1 - Insertion en tête de liste\n");
    printf ("2 - Insertion en fin de liste\n");
    printf ("3 - Retrait en tête de liste\n");
    printf ("4 - Retrait en fin de liste\n");
    printf ("5 - Retrait d'un élément à partir de son nom\n");
    printf ("6 - Parcours de la liste\n");
    printf ("7 - Recherche d'un élément à partir de son nom\n");
    printf ("8 - Destruction de la liste\n");
    printf ("\n");
    printf ("Votre choix ? ");
    int cod; scanf ("%d", &cod);
    printf ("\n");
    return cod;
}
```

personne.h

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

typedef char ch15 [16];
typedef void Objet;

typedef struct {
    ch15 nom;
    ch15 prenom;

} Personne;

Personne* creerPersonne(char* nom,char* prenom) ;
char* ecrirePersonne (Objet* objet);
int comparerPersonne (Objet* objet1,Objet* objet2);
```

personne.c

```
#include <stdio.h>
#include <string.h>
#include "personne.h"

Personne* creerPersonne(char* nom,char* prenom) {
    Personne* p= malloc (sizeof(Personne));
    strcpy(p->nom, nom);
    strcpy(p->prenom,prenom);
    return p;
}

char* ecrirePersonne(Objet* objet){
    Personne* p =(Personne*) objet;
    char* output =(char*)malloc (sizeof(Personne));

    sprintf(output,sizeof(Personne) , "%s %s\n",p->nom,p->prenom);
    return output;
}

int comparerPersonne(Objet* objet1,Objet* objet2){
    Personne* p1=(Personne*)objet1;
    Personne* p2=(Personne*)objet2;
    return strcmp(p1->nom,p2->nom);
}
```

listePersonne.c

```
#include <stdio.h>
#include <string.h>
#include "personne.h"
#include "liste.h"
typedef int booleen;

int comparerPersonne(Objet* objet1,Objet* objet2){
    Personne* p1=(Personne*)objet1;
    Personne* p2=(Personne*)objet2;
    return strcmp(p1->nom,p2->nom);
}
char* ecrirePersonne(Objet* objet){
    Personne* p =(Personne*) objet;
    char* output =(char*)malloc (sizeof(Personne));

    sprintf(output,sizeof(Personne) , "%s %s\n",p->nom,p->prenom);
    return output;
}
Personne* creerPersonne(char* nom,char* prenom){
    Personne* p= malloc (sizeof(Personne));
    strcpy(p->nom, nom);
    strcpy(p->prenom,prenom);
    return p;
}
Liste* creerListe(int type,char* (*afficher)(objet*),int
(*comparer)(objet*,objet*)){
    Liste* li=(Liste*)malloc(sizeof(Liste));
    initListe(li,type,afficher,comparer);
    return li;
}
void initListe (Liste * li,int type,char*
(*afficher)(objet*),int(*comparer)(objet*,objet*)){
    li->premier=NULL;
    li->dernier=NULL;
    li->courant=NULL;
    li->nbElt=0;
    li->type=type;
    li->afficher=afficher;
    li->comparer=comparer;
}
static Element* creerElement(){
    return (Element *)malloc(sizeof(Element));
}

void insererEnTeteDeListe(Liste * li,objet * objet){
    Element * nouveau=creerElement();
    nouveau->reference=objet;
    nouveau->suivant=li->premier;
```

```

        li->premier=nouveau;
        if(li->dernier==NULL) li->dernier=nouveau;
        li->nbElt++;
    }

static void insererApres(Liste *li,Element *precedent,objet* objet){
    if(precedent==NULL){
        insererEnTeteDeListe(li,objet);
    }
    else{
        Element* nouveau=creerElement();
        nouveau->reference=objet;
        nouveau->suivant=precedent->suivant;
        precedent->suivant=nouveau;
        if(precedent==li->dernier) li->dernier=nouveau;
        li->nbElt++;
    }
}

void insererEnFinDeListe(Liste *li,void *obj){
    insererApres(li,li->dernier,obj);
}

objet* extraireEnTeteDeListe (Liste* li){
    Element* extrait=li->premier;
    if(!listevide(li)){
        li->premier=li->premier->suivant;
        if(li->premier==NULL) li->dernier=NULL;
        li->nbElt--;
    }
    return extrait !=NULL ? extrait->reference : NULL;
}

booleen listevide(Liste* li){
return li-> nbElt == 0;
}

static void * extraireApres(Liste *li,Element *precedent){
    if (precedent ==NULL){
        return extraireEnTeteDeListe(li);
    }
    else{
        Element *extrait=precedent->suivant;
        if (extrait!=NULL){
            precedent->suivant=extrait->suivant;
            if(extrait==li->dernier) li->dernier=precedent;
            li->nbElt--;
        }
        return extract != NULL? extract->reference :NULL;
    }
}

objet* extraireEnFinDeListe(Liste *li ){

```

```

objet *extrait;
if(listevide(li)){
    extrait=NULL;
}else if (li->premier==li->dernier){
    extrait = extraireEnTeteDeListe(li);
}else {
    Element *ptc = li->premier;
    while(ptc->suivant !=li->dernier) ptc=ptc->suivant;
    extrait = extraireApres(li,ptc);
}
return extrait;
}

booleen extraireUnobjet (Liste *li,objet *objet){
    Element* precedent=NULL;
    Element* ptc =NULL;
    booleen trouve = faux;

    ouvrirListe(li);
    while(!finListe(li) && !trouve){
        precedent =ptc;
        ptc=elementCourant(li);
        trouve=(ptc->reference==objet) ? vrai:faux;
    }
    if(!trouve) return faux;
    void* extrait = extraireApres(li,precedent);
    return vrai;
}
static Element* elementCourant (Liste* li){
    Element *ptc=li->courant;
    if(li->courant != NULL){
        li->courant=li->courant->suivant;
    }
    return ptc;
}
void ouvrirListe(Liste * li){
    li->courant=li->premier;
}
booleen finListe(Liste *li){
    return li->courant==NULL;
}
void listerListe(Liste* li){
    ouvrirListe(li);
    while(!finListe(li)){
        objet* objet=objetCourant(li);
        printf("%s\n",li->afficher(objet));
    }
}

objet* objetCourant (Liste *li){
    Element *ptc = elementCourant(li);
    return ptc==NULL ? NULL : ptc->reference;
}

```

```

objet* chercherUnobjet(Liste* li,objet* objetCherche) {
    boolean trouve = faux;
    objet* objet;
    ouvrirListe(li);
    while(!finListe(li) && !trouve){
        objet=objetCourant(li);
        trouve =li->comparer(objetCherche,objet)==0;
    }
    return trouve ? objet:NULL;
}

void detruireListe(Liste *li){
    ouvrirListe(li);
    while(!finListe(li)){
        Element *ptc =elementCourant(li);
        free(ptc);
    }
    initListe(li,0,NULL,NULL);
}

////////////////////////////////////////////////////////////////
int menu () {
    printf ("\n\nGESTION D'UNE LISTE DE PERSONNES\n\n");
    printf ("0 - Fin\n");
    printf ("1 - Insertion en tête de liste\n");
    printf ("2 - Insertion en fin de liste\n");
    printf ("3 - Retrait en tête de liste\n");
    printf ("4 - Retrait en fin de liste\n");
    printf ("5 - Retrait d'un élément à partir de son nom\n");
    printf ("6 - Parcours de la liste\n");
    printf ("7 - Recherche d'un élément à partir de son nom\n");
    printf ("8 - Destruction de la liste\n");
    printf ("\n");
    printf ("Votre choix ? ");
    int cod; scanf ("%d", &cod);
    printf ("\n");
    return cod;
}
int main(){
    Liste* lp=creerListe(0,ecrirePersonne,comparerPersonne);
    boolean fini=faux;
    while(!fini){
        switch(menu()){
            case 0:
                fini=vrai;
                break;

            case 1:{
```

```

        printf("nom de la personne a creer?");
        ch15 nom ;
        scanf("%s",nom);
        printf("prenom de la personne a creer?");
        ch15 prenom;
        scanf("%s",prenom);
        Personne* nouveau = creerPersonne(nom,prenom) ;
        insererEnTeteDeListe(lp,nouveau) ;
        break;
    }

    case 2:
    {
        printf("nom de la personne a creer?");
        ch15 nom ;
        scanf("%s",nom);
        printf("prenom de la personne a creer?");
        ch15 prenom;
        scanf("%s",prenom);
        Personne* nouveau= creerPersonne(nom,prenom) ;
        insererEnFinDeListe(lp,nouveau) ;
        break;
    }

    case 3:
    {
        Personne* extrait=(Personne*)
            extraireEnTeteDeListe(lp);
        if(extrait != NULL){
            printf("element %s %s extrait en
tete de liste",extrait->nom,extrait->prenom);

        }
        else{
            printf("liste vide");
        }
        break;
    }

    case 4:
    {
        Personne* extrait=(Personne*)
            extraireEnFinDeListe(lp);
        if(extrait != NULL){
            printf("element %s %s extrait en
fin de liste", extrait->nom,extrait->prenom);

        }
        else{
            printf("liste vide");
        }
        break;
    }
}

```

```

        }

case 5:
{
    printf("nom de la personne a
           extraire?");
    ch15 nom;scanf("%s",nom);
    Personne*cherche=creerPersonne(nom,"?");
    Personne*pp=(Personne*)chercherUnobjet(lp,cherche);
    if(extraireUnobjet(lp,pp)){
        printf("element %s %s extrait de
               la liste",pp->nom,pp->prenom);
    }
    break;

}
case 6 :
    listerListe(lp);
    break;

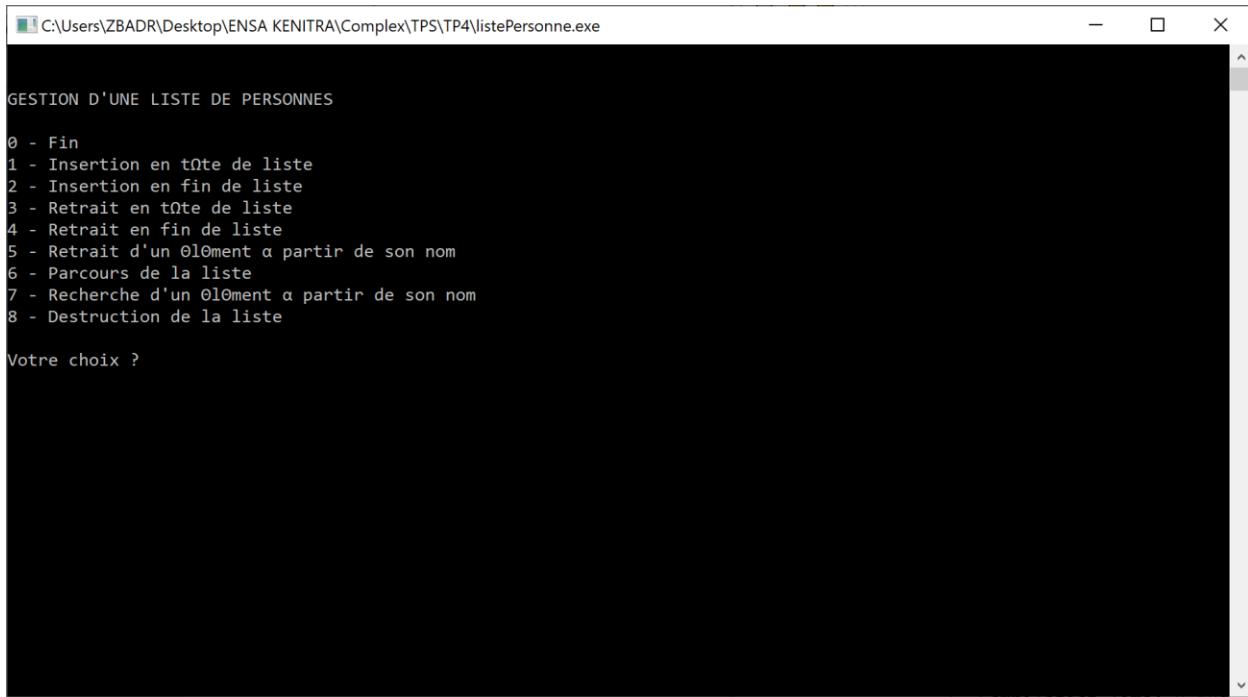
case 7:
{
    printf("nom de la personne
           recherchée?");
    ch15 nom;scanf("%s",nom);
    Personne*cherche=creerPersonne(nom,"");
    Personne*pp=(Personne*)chercherUnobjet(lp,cherche);
    if(pp != NULL){
        printf("%s %s trouvée dans la
               liste\n",pp->nom,pp->prenom);

    }else{
        printf("%s innconnue dans la
               liste\n",nom);
    }
    break;
}

case 8:
    detruireListe(lp);
    break;
}
}

```

Menu



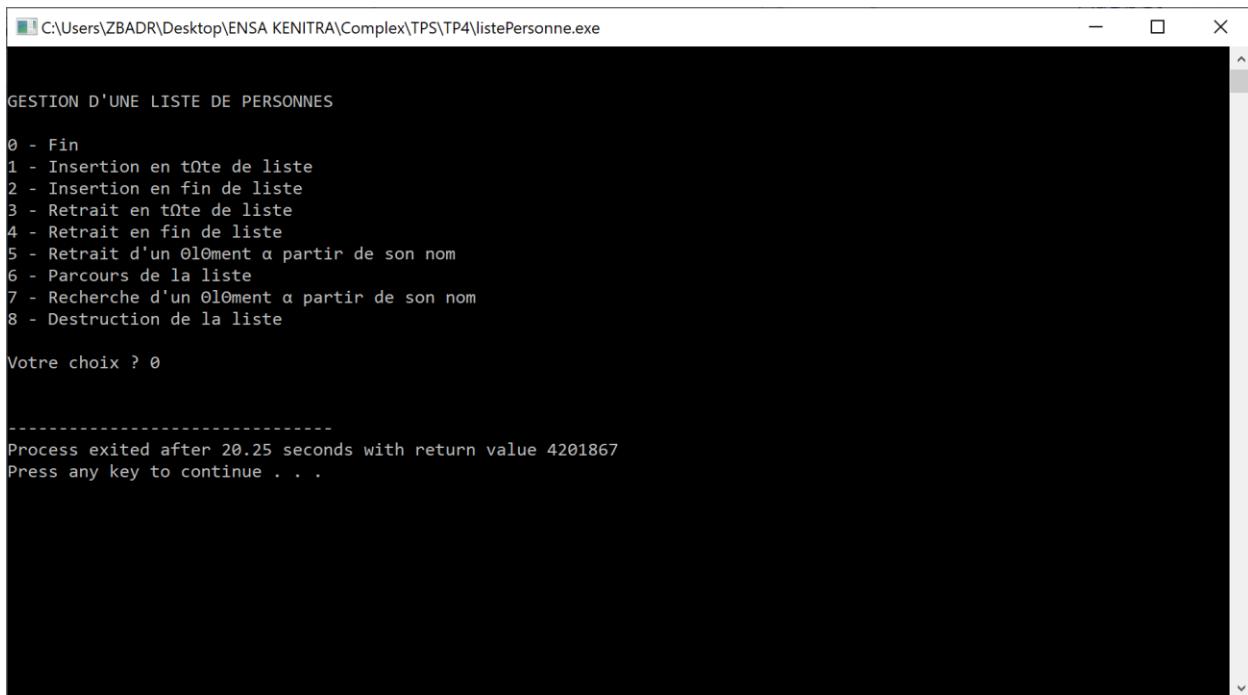
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP4\listePersonne.exe

GESTION D'UNE LISTE DE PERSONNES

```
0 - Fin
1 - Insertion en tØte de liste
2 - Insertion en fin de liste
3 - Retrait en tØte de liste
4 - Retrait en fin de liste
5 - Retrait d'un élément à partir de son nom
6 - Parcours de la liste
7 - Recherche d'un élément à partir de son nom
8 - Destruction de la liste
```

Votre choix ?

0 – Fin



C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP4\listePersonne.exe

GESTION D'UNE LISTE DE PERSONNES

```
0 - Fin
1 - Insertion en tØte de liste
2 - Insertion en fin de liste
3 - Retrait en tØte de liste
4 - Retrait en fin de liste
5 - Retrait d'un élément à partir de son nom
6 - Parcours de la liste
7 - Recherche d'un élément à partir de son nom
8 - Destruction de la liste
```

Votre choix ? 0

Process exited after 20.25 seconds with return value 4201867
Press any key to continue . . .

1 - Insertion en tête de liste

```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP4\listePersonne.exe

0 - Fin
1 - Insertion en tête de liste
2 - Insertion en fin de liste
3 - Retrait en tête de liste
4 - Retrait en fin de liste
5 - Retrait d'un élément à partir de son nom
6 - Parcours de la liste
7 - Recherche d'un élément à partir de son nom
8 - Destruction de la liste

Votre choix ? 1

nom de la personne à créer?BADRY
prénom de la personne à créer?ZAKARIA

GESTION D'UNE LISTE DE PERSONNES

0 - Fin
1 - Insertion en tête de liste
2 - Insertion en fin de liste
3 - Retrait en tête de liste
4 - Retrait en fin de liste
5 - Retrait d'un élément à partir de son nom
6 - Parcours de la liste
7 - Recherche d'un élément à partir de son nom
8 - Destruction de la liste

Votre choix ?
```

2 - Insertion en fin de liste

```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP4\listePersonne.exe

0 - Fin
1 - Insertion en tête de liste
2 - Insertion en fin de liste
3 - Retrait en tête de liste
4 - Retrait en fin de liste
5 - Retrait d'un élément à partir de son nom
6 - Parcours de la liste
7 - Recherche d'un élément à partir de son nom
8 - Destruction de la liste

Votre choix ? 2

nom de la personne à créer?RESSANOSS
prénom de la personne à créer?SPONCHE

GESTION D'UNE LISTE DE PERSONNES

0 - Fin
1 - Insertion en tête de liste
2 - Insertion en fin de liste
3 - Retrait en tête de liste
4 - Retrait en fin de liste
5 - Retrait d'un élément à partir de son nom
6 - Parcours de la liste
7 - Recherche d'un élément à partir de son nom
8 - Destruction de la liste

Votre choix ?
```

3 - Retrait en tête de liste

```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP4\listePersonne.exe

GESTION D'UNE LISTE DE PERSONNES

0 - Fin
1 - Insertion en tØte de liste
2 - Insertion en fin de liste
3 - Retrait en tØte de liste
4 - Retrait en fin de liste
5 - Retrait d'un élément à partir de son nom
6 - Parcours de la liste
7 - Recherche d'un élément à partir de son nom
8 - Destruction de la liste

Votre choix ? 3

élément BADRY ZAKARIA extrait en tête de liste

GESTION D'UNE LISTE DE PERSONNES

0 - Fin
1 - Insertion en tØte de liste
2 - Insertion en fin de liste
3 - Retrait en tØte de liste
4 - Retrait en fin de liste
5 - Retrait d'un élément à partir de son nom
6 - Parcours de la liste
7 - Recherche d'un élément à partir de son nom
8 - Destruction de la liste

Votre choix ?
```

4 - Retrait en fin de liste

```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP4\listePersonne.exe

GESTION D'UNE LISTE DE PERSONNES

0 - Fin
1 - Insertion en tØte de liste
2 - Insertion en fin de liste
3 - Retrait en tØte de liste
4 - Retrait en fin de liste
5 - Retrait d'un élément à partir de son nom
6 - Parcours de la liste
7 - Recherche d'un élément à partir de son nom
8 - Destruction de la liste

Votre choix ? 4

élément RESSANOSS SPONCHE extrait en fin de liste

GESTION D'UNE LISTE DE PERSONNES

0 - Fin
1 - Insertion en tØte de liste
2 - Insertion en fin de liste
3 - Retrait en tØte de liste
4 - Retrait en fin de liste
5 - Retrait d'un élément à partir de son nom
6 - Parcours de la liste
7 - Recherche d'un élément à partir de son nom
8 - Destruction de la liste

Votre choix ?
```

5 - Retrait d'un élément à partir de son nom

```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP4\listePersonne.exe
GESTION D'UNE LISTE DE PERSONNES

0 - Fin
1 - Insertion en tØte de liste
2 - Insertion en fin de liste
3 - Retrait en tØte de liste
4 - Retrait en fin de liste
5 - Retrait d'un ØlØment a partir de son nom
6 - Parcours de la liste
7 - Recherche d'un ØlØment a partir de son nom
8 - Destruction de la liste

Votre choix ? 5

nom de la personne a extraire?ZIKOU
element ZIKOU BAD extrait de la liste

GESTION D'UNE LISTE DE PERSONNES

0 - Fin
1 - Insertion en tØte de liste
2 - Insertion en fin de liste
3 - Retrait en tØte de liste
4 - Retrait en fin de liste
5 - Retrait d'un ØlØment a partir de son nom
6 - Parcours de la liste
7 - Recherche d'un ØlØment a partir de son nom
8 - Destruction de la liste

Votre choix ?
```

6 - Parcours de la liste

```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP4\listePersonne.exe
1 - Insertion en tØte de liste
2 - Insertion en fin de liste
3 - Retrait en tØte de liste
4 - Retrait en fin de liste
5 - Retrait d'un ØlØment a partir de son nom
6 - Parcours de la liste
7 - Recherche d'un ØlØment a partir de son nom
8 - Destruction de la liste

Votre choix ? 6

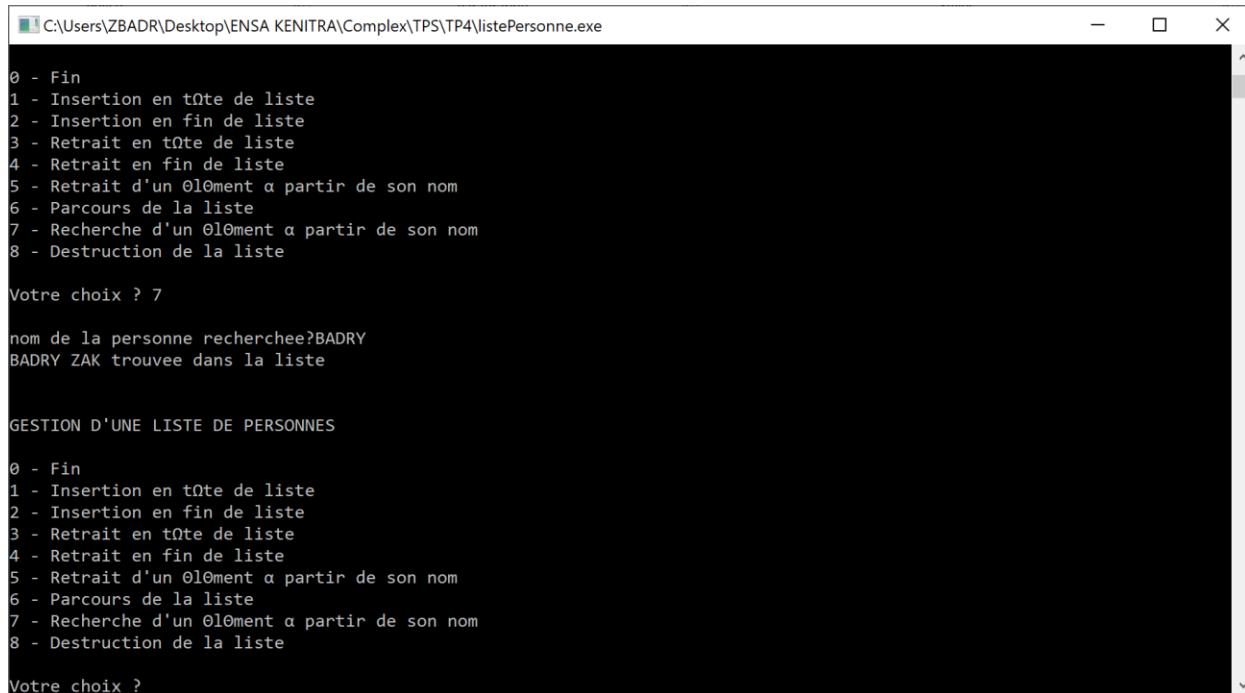
BADRY ZAK
SPONCHE BBOB

GESTION D'UNE LISTE DE PERSONNES

0 - Fin
1 - Insertion en tØte de liste
2 - Insertion en fin de liste
3 - Retrait en tØte de liste
4 - Retrait en fin de liste
5 - Retrait d'un ØlØment a partir de son nom
6 - Parcours de la liste
7 - Recherche d'un ØlØment a partir de son nom
8 - Destruction de la liste

Votre choix ?
```

7 - Recherche d'un élément à partir de son nom



```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP4\listePersonne.exe

0 - Fin
1 - Insertion en t̄t̄e de liste
2 - Insertion en fin de liste
3 - Retrait en t̄t̄e de liste
4 - Retrait en fin de liste
5 - Retrait d'un élément à partir de son nom
6 - Parcours de la liste
7 - Recherche d'un élément à partir de son nom
8 - Destruction de la liste

Votre choix ? 7

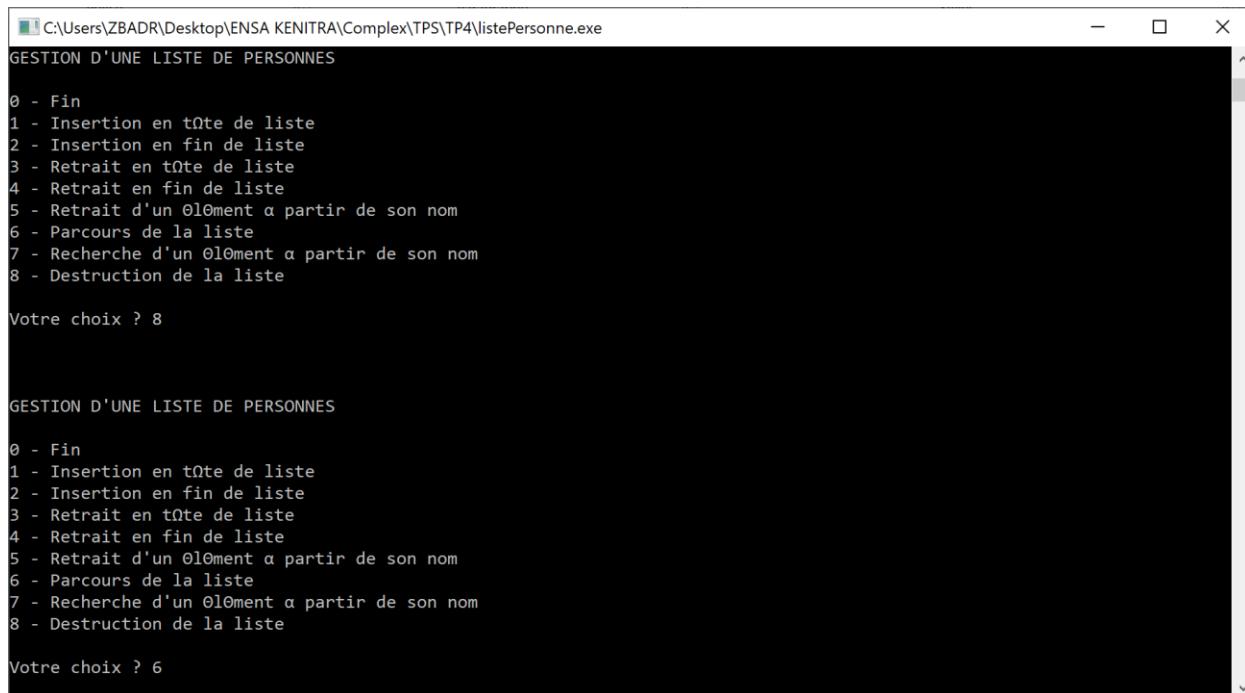
nom de la personne recherchée?BADRY
BADRY ZAK trouvée dans la liste

GESTION D'UNE LISTE DE PERSONNES

0 - Fin
1 - Insertion en t̄t̄e de liste
2 - Insertion en fin de liste
3 - Retrait en t̄t̄e de liste
4 - Retrait en fin de liste
5 - Retrait d'un élément à partir de son nom
6 - Parcours de la liste
7 - Recherche d'un élément à partir de son nom
8 - Destruction de la liste

Votre choix ?
```

8 - Destruction de la liste



```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP4\listePersonne.exe

GESTION D'UNE LISTE DE PERSONNES

0 - Fin
1 - Insertion en t̄t̄e de liste
2 - Insertion en fin de liste
3 - Retrait en t̄t̄e de liste
4 - Retrait en fin de liste
5 - Retrait d'un élément à partir de son nom
6 - Parcours de la liste
7 - Recherche d'un élément à partir de son nom
8 - Destruction de la liste

Votre choix ? 8

GESTION D'UNE LISTE DE PERSONNES

0 - Fin
1 - Insertion en t̄t̄e de liste
2 - Insertion en fin de liste
3 - Retrait en t̄t̄e de liste
4 - Retrait en fin de liste
5 - Retrait d'un élément à partir de son nom
6 - Parcours de la liste
7 - Recherche d'un élément à partir de son nom
8 - Destruction de la liste

Votre choix ? 6
```

TP 5

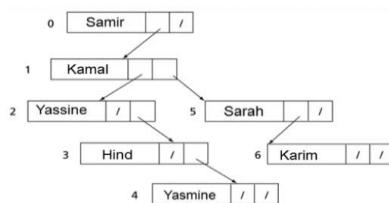
Exercice 1 :

En vous inspirant du TP précédent, implémentez :

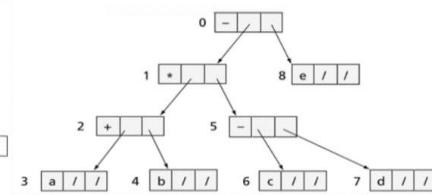
- arbre.h : qui contient la déclaration du type arbre et les prototypes des fonctions de gestion d'un arbre.
- arbre.cpp : qui contient le corps des fonctions dont le prototype est défini dans arbre.h.
- mainarbre.cpp : qui est le programme principal des arbres. Il définit un menu comme suit :

```
int menu () {
    printf ("\n\n GESTION D'ARBRES \n\n");
    printf ("\n\n ARBRES BINAIRES \n\n");
    printf (" 0 - Fin du programme\n");
    printf ("\n");
    printf (" 1 - Cration de l'arbre gnalogique\n");
    printf (" 2 - Cration de l'arbre de l'expression arithmtique\n");
    printf ("\n");
    printf (" 3 - Parcours prfix\n");
    printf (" 4 - Parcours infix\n");
    printf (" 5 - Parcours postfix\n");
    printf (" 6 - Trouver Noeud \n");
    printf (" 7 - Taille \n");
    printf (" 8 - Hauteur \n");
    printf (" 9 - Parcours en Largeur \n");
    printf ("\n");
    printf ("Votre choix ? ");
    int cod; scanf ("%d", &cod); getchar();
    printf ("\n");
    return cod;
}
```

Arbre gnalogique  utiliser :



Arbre de l'expression arithmtique  utiliser :



ARBRE .H

```
#define faux 0
#define vrai 1
typedef int booleen;
typedef void Objet;

typedef int booleen;
///////////
typedef void objet;
/////// MEMORISATION DUN ARBRE BINAIRE
typedef struct noeud {
    objet* reference;
    struct noeud* gauche;
    struct noeud* droite;
}Noeud;

typedef struct {
    Noeud* racine;
    char* (*afficher) (objet* );
    int (*comparer) (objet*,objet* );
}Arbre;

Noeud* cNd(objet* objet,Noeud* gauche,Noeud* droite);

Noeud* cF (objet* objet);

void initArbre (Arbre* arbre,Noeud* racine,char* (*afficher) (objet* ),int
(*comparer) (objet*,objet* ));

Arbre* creeArbre (Noeud* racine, char* (*afficher) (objet* ),int
(*comparer) (objet*,objet* ));

void prefixe (Noeud* racine,char* (*afficher) (objet* ));

void infixe (Noeud* racine,char* (*afficher) (objet* ));

void postfixe(Noeud* racine , char* (*afficher) (objet* ));
Noeud* trouverNoeud (Noeud* racine, Objet*
objet,int(*comparer) (Objet*,Objet* ));

void enLargeur(Noeud* racine,char* (*afficher) (objet* ));

int taille(Noeud* racine);
```

```
booleen estFeuille (Noeud* racine);

int nbFeuilles (Noeud* racine);

void listerFeuilles (Noeud* racine,char* (*afficher) (objet*));

int hauteur (Noeud* racine);

booleen egaliteArbre (Noeud* racine1,Noeud* racine2,int
(*comparer) (objet*,objet*));
```

arbre.cpp

```
#include<stdio.h>
#include<stdlib.h>
#include <math.h>
#include "arbre.h"
#include "liste.h"

Noeud* cNd(objet* objet,Noeud* gauche,Noeud* droite){
    Noeud* nouveau= (Noeud*) malloc(sizeof(Noeud));
    nouveau->reference=objet;
    nouveau->gauche=gauche;
    nouveau->droite=droite;
    return nouveau;
}

Noeud* cF (objet* objet){
    return cNd(objet,NULL,NULL);
}

void initArbre (Arbre* arbre,Noeud* racine,char* (*afficher) (objet*),int
(*comparer) (objet*,objet*)) {
    arbre->racine=racine;
    arbre->afficher=afficher;
    arbre->comparer=comparer;
}

Arbre* creeArbre (Noeud* racine, char* (*afficher)(objet*),int
(*comparer)(objet*,objet*)) {
    Arbre* arbre=(Arbre*) malloc(sizeof(Arbre));
    initArbre(arbre, racine, afficher, comparer);
    return arbre;
}

void prefixe (Noeud* racine,char* (*afficher)(objet*)) {
    if(racine != NULL) {
        printf("%s",afficher(racine->reference));
        prefixe(racine->gauche,afficher);
        prefixe(racine->droite,afficher);
    }
}
```

```

void infixe (Noeud* racine,char* (*afficher) (objet*)) {
    if(racine != NULL){
        infixe(racine->gauche,afficher);
        printf("%s",afficher(racine->reference));
        infixe(racine->droite,afficher);
    }
}

-----
void postfixe(Noeud* racine , char* (*afficher) (objet*)) {
    if(racine != NULL){
        postfixe(racine->gauche,afficher);
        postfixe(racine->droite,afficher);
        printf("%s",afficher(racine->reference));
    }
}

-----
Noeud* trouverNoeud (Noeud* racine, Objet*
objet,int(*comparer) (Objet*,Objet*))
{
    Noeud* pNom;
    if (racine==NULL){
        pNom=NULL;
    }
    else if(comparer(racine->reference,objet)==0){
        pNom=racine;
    }
    else{
        pNom=trouverNoeud(racine->droite,objet,comparer);
    }
    return pNom;
}

-----
void enLargeur(Noeud* racine,char* (*afficher) (objet*)) {
    Liste* li=creerListe(0,NULL,NULL);
    insererEnFinDeListe(li,racine);
    while(!listevide(li)){
        Noeud* extrait=(Noeud*)extraireEnTeteDeListe(li);
        printf("%s",afficher(extrait->reference));
        if(extrait->gauche!=NULL) insererEnFinDeListe(li,extrait-
>gauche);
        if(extrait->droite!=NULL) insererEnFinDeListe(li,extrait-
>droite);
    }
}

-----
int taille(Noeud* racine){
    if(racine==NULL){
        return 0;
    }
}

```

```

        else{
            return 1+taille(racine->gauche)+taille(racine-
>droite);
        }
    }

booleen estFeuille (Noeud* racine){
    return (racine->gauche==NULL) && (racine->droite==NULL);
}

int nbFeuilles (Noeud* racine){
    if(racine==NULL){
        return 0;
    }
    else if(estFeuille(racine)){
        return 1;
    }
    else {
        return nbFeuilles (racine->gauche)+nbFeuilles (racine-
>droite);
    }
}

void listerFeuilles (Noeud* racine,char* (*afficher) (objet*)) {
    if (estFeuille (racine)){
        printf("%s",afficher(racine->reference));
    }
    else{
        listerFeuilles(racine->gauche,afficher);
        listerFeuilles(racine->droite,afficher);

    }
}

int hauteur (Noeud* racine){
    if(racine==NULL){
        return 0;
    }
    else{
        return 1+fmax(hauteur(racine->gauche),hauteur(racine-
>droite));
    }
}

```

```
booleen egaliteArbre (Noeud* racine1,Noeud* racine2,int
(*comparer)(objet*,objet*)) {
    booleen resu=faux;
    if((racine1==NULL) && (racine2==NULL)){
        resu=vrai;
    }
    else if((racine1!=NULL)&&(racine2!=NULL)){
        if(comparer(racine1->reference, racine2->reference)==0) {
            if(egaliteArbre(racine1->gauche, racine2-
>gauche,comparer)) {
                resu=egaliteArbre(racine1->droite, racine2-
>droite,comparer);
            }
        }
    }
    return resu;
}
```

mainarbre.cpp

```
#include<stdio.h>
#include<stdlib.h>
#include <string.h>
#include <math.h>
#include "arbre.h"
#include "liste.h"
typedef void objet;
typedef int boolean;
///////////
char* afficher (objet* objet){
    return (char*)objet;
}

int comparer(Objet* objet1, Objet* objet2 )
{
    char* p1=(char*) objet1;
    char* p2=(char*) objet2;
    return strcmp (p1,p2);
}

Arbre* creeArbre (Noeud* racine, char* (*afficher) (objet*),int
(*comparer) (objet*,objet*)) {
    Arbre* arbre=(Arbre*) malloc(sizeof(Arbre));
    initArbre(arbre,racine,afficher,comparer);
    return arbre;
}

void initArbre (Arbre* arbre,Noeud* racine,char* (*afficher) (objet*),int
(*comparer) (objet*,objet*)) {
    arbre->racine=racine;
    arbre->afficher=afficher;
    arbre->comparer=comparer;
}

Noeud* cF (objet* objet) {
    return cNd(objet,NULL,NULL);
}

Noeud* cNd(objet* objet,Noeud* gauche,Noeud* droite) {
    Noeud* nouveau= (Noeud*) malloc(sizeof(Noeud));
    nouveau->reference=objet;
    nouveau->gauche=gauche;
    nouveau->droite=droite;
    return nouveau;
}

void prefixe (Noeud* racine,char* (*afficher) (objet*)) {
    if(racine != NULL){
```

```

        printf("%s", afficher(racine->reference));
        prefixe(racine->gauche, afficher);
        prefixe(racine->droite, afficher);
    }
}

void infixe (Noeud* racine, char* (*afficher) (objet*)) {
    if(racine != NULL){
        infixe(racine->gauche, afficher);
        printf("%s", afficher(racine->reference));
        infixe(racine->droite, afficher);
    }
}

void postfixe(Noeud* racine , char* (*afficher) (objet*)) {
    if(racine != NULL){
        postfixe(racine->gauche, afficher);
        postfixe(racine->droite, afficher);
        printf("%s", afficher(racine->reference));
    }
}

Noeud* trouverNoeud (Noeud* racine, Objet*
objet,int(*comparer) (Objet*,Objet*))
{
    Noeud* pNom;
    if (racine==NULL){
        pNom=NULL;
    }
    else if(comparer(racine->reference,objet)==0){
        pNom=racine;
    }
    else{
        pNom=trouverNoeud(racine->droite,objet,comparer);
    }
    return pNom;
}

int taille(Noeud* racine) {
    if(racine==NULL) {
        return 0;
    }

    else{
        return 1+taille(racine->gauche)+taille(racine-
>droite);
    }
}

int hauteur (Noeud* racine){

```

```

        if(racine==NULL) {
            return 0;
        }
        else{
            return 1+fmax(hauteur(racine->gauche),hauteur(racine-
>droite));
        }
    }

void enLargeur(Noeud* racine,char* (*afficher) (objet*)) {
    Liste* li=creerListe(0,NULL,NULL);
    insererEnFinDeListe(li,racine);
    while(!listevide(li)){
        Noeud* extrait=(Noeud*)extraireEnTeteDeListe(li);
        printf("%s",afficher (extrait->reference));
        if(extrait->gauche!=NULL) insererEnFinDeListe(li,extrait-
>gauche);
        if(extrait->droite!=NULL) insererEnFinDeListe(li,extrait-
>droite);
    }
}

Liste* creerListe(int type,char* (*afficher) (objet*),int
(*comparer) (objet*,objet*)) {
    Liste* li=(Liste*)malloc(sizeof(Liste));
    initListe(li,type,afficher,comparer);
    return li;
}

void insererEnFinDeListe(Liste *li,void *obj){
    insererApres(li,li->dernier,obj);
}

static Element* creerElement(){
    return (Element *)malloc(sizeof(Element));
}

void insererEnTeteDeListe(Liste * li,objet * objet){
    Element * nouveau=creerElement();
    nouveau->reference=objet;
    nouveau->suivant=li->premier;
    li->premier=nouveau;
    if(li->dernier==NULL) li->dernier=nouveau;
    li->nbElt++;
}

static void insererApres(Liste *li,Element *precedent,objet* objet){
    if(precedent==NULL){
        insererEnTeteDeListe(li,objet);
    }
    else{

```

```

        Element* nouveau=creerElement();
        nouveau->reference=objet;
        nouveau->suivant=precedent->suivant;
        precedent->suivant=nouveau;
        if(precedent==li->dernier) li->dernier=nouveau;
        li->nbElt++;
    }
}

booleen listevide(Liste* li){
return li-> nbElt == 0;
}

objet* extraireEnTeteDeListe (Liste* li){
    Element* extrait=li->premier;
    if(!listevide(li)){
        li->premier=li->premier->suivant;
        if(li->premier==NULL) li->dernier=NULL;
        li->nbElt--;
    }
    return extrait !=NULL ? extrait->reference : NULL;
}

void initListe (Liste * li,int type,char*
(*afficher)(objet*),int(*comparer)(objet*,objet*)) {
    li->premier=NULL;
    li->dernier=NULL;
    li->courant=NULL;
    li->nbElt=0;
    li->type=type;
    li->afficher=afficher;
    li->comparer=comparer;
}

///////////////////
int menu () {
printf ("\n\n GESTION D'ARBRES \n\n");
printf ("\n\n ARBRES BINAIRES \n\n");
printf (" 0 - Fin du programme\n");
printf ("\n");
printf (" 1 - Cr ation de l'arbre g n alogique\n");
printf (" 2 - Cr ation de l'arbre de l'expression arithm tique\n");
printf ("\n");
printf (" 3 - Parcours pr fix \n");
printf (" 4 - Parcours infix \n");

```

```

printf (" 5 - Parcours postfixé\n");
printf (" 6 - Trouver Noeud \n");
printf (" 7 - Taille \n");
printf (" 8 - Hauteur \n");
printf (" 9 - Parcours en Largeur \n");
printf ("\n");
printf ("Votre choix ? ");
int cod; scanf ("%d", &cod); getchar();
printf ("\n");
return cod;
}

int main(){
    //////////////////////////////

    Arbre *arbregen,*arbreari;
    Noeud*rgen=cNd((char*)"samir\n",cNd((char*)"kamal\n",cNd((char*)"yassi
ne\n",NULL,cNd((char*)"hind\n",NULL,cF((char*)"yasmine\n"))),cNd((char*)"sara
h\n",cF((char*)"karim\n"),NULL)),NULL);
    Noeud*rari=cNd((char*)" \n",cNd((char*)"\*\n",cNd((char*)"+\n",cF((char*
) "a\n"),cF((char*)"b\n")),cNd((char*)"- \n",cF((char*)"c\n"),cF((char*)"d\n"))),cF((char*)"e\n"));

    //////////////////////////////

    int cod;
    do{
        cod=menu();
        switch(cod){
            case 0:
                break ;
            case 1 ://creation de l'arbre genealogique
                arbregen=creeArbre(rgen,afficher,comparer);
                break;
            case 2://Création de l'arbre de l'expression
                    arithmétique
                arbreari=creeArbre(rari,afficher,comparer);
                break;
            case 3 ://préfixé
                printf("1-pour afficher arbre genealogique /
                    2-pour afficher arbre arithmétique ");
                int cpr; scanf("%d",&cpr);
                switch(cpr){
                    case 1:
                        prefixe(rgen,afficher);
                        break;
                    case 2:
                        prefixe(rari,afficher);
                        break; }

            case 4 ://infixé
        }
    }
}

```

```

    {
        printf("1-pour afficher arbre genealogique /
               2-pour afficher arbre arithmétique");
        int cin;scanf("%d",&cin);
        switch(cin){
            case 1:
                infixe(rgen,afficher);
            case 2:
                infixe(rari,afficher);
            }
        }
        break;
    case 5:
    {
        printf("1-pour afficher arbre genealogique /
               2-pour afficher arbre arithmétique");
        int cpo;scanf("%d",&cpo);
        switch(cpo){
            case 1:
                postfixe(rgen,afficher);
            case 2:
                postfixe(rari,afficher);

        }
        break;
    }

    case 6:{

        printf("1-rechercher un noeud ds arbre
               genealogique/2-rechercher un noeud ds arbre
               arithmétique");
        int cre;scanf("%d",&cre);
        switch(cre){
            case 1:
            {
                printf("donner le noeud
                       chercher");
                char* ng=(char*)malloc(sizeof(char));scanf("%s",ng);
                Noeud* trouve;
                trouve=trouverNoeud(rgen,ng,comparer);
                if(trouve==NULL){
                    printf("noeud n existe pas");
                }
                else {
                    printf("noeud trouver   ");
                }
                break;
            }
            case 2:{

                printf("donner le noeud chercher");
                char* na=(char*)malloc(sizeof(char));scanf("%s",na);
                Noeud* trouve;

```

```

trouve=trouverNoeud(rari,na,comparer);
        if(trouve==NULL){
            printf("noeud n existe pas");
        }
        else {
            printf("noeud trouver   ");
        }

        break;
    }
}
break;
}

case 7://taille
{
printf("1-TAILLE arbre genealogique/
       2-TAILLE arbre arithmétique");
int ct;scanf("%d",&ct);
switch(ct){
    case 1:
printf("la taille est %d",taille(rgen));
        break;
    case 2:
printf("la taille est %d",taille(rari));
        break;
}

}break;

case 8:{

printf("1-Hauteur arbre genealogique/
       2-Hauteur arbre arithmétique");
int ch;scanf("%d",&ch);
switch(ch){
    case 1:
printf("la hauteur est %d",hauteur(rgen));
        break;
    case 2:
printf("la hauteur est %d",hauteur(rari));
        break;
}

}break;
case 9://Largeur
{
printf("1-parcours arbre genealogique en largeur/
       2-parcours arbre arithmétique en largeur");
int cl;scanf("%d",&cl);
switch(cl){
    case 1: {
        enLargeur(rgen,afficher);
    }
}
}

```

```
        break;
    }

    case 2 :{
        enLargeur(rari,afficher);
        break;
    }

}

break;
}

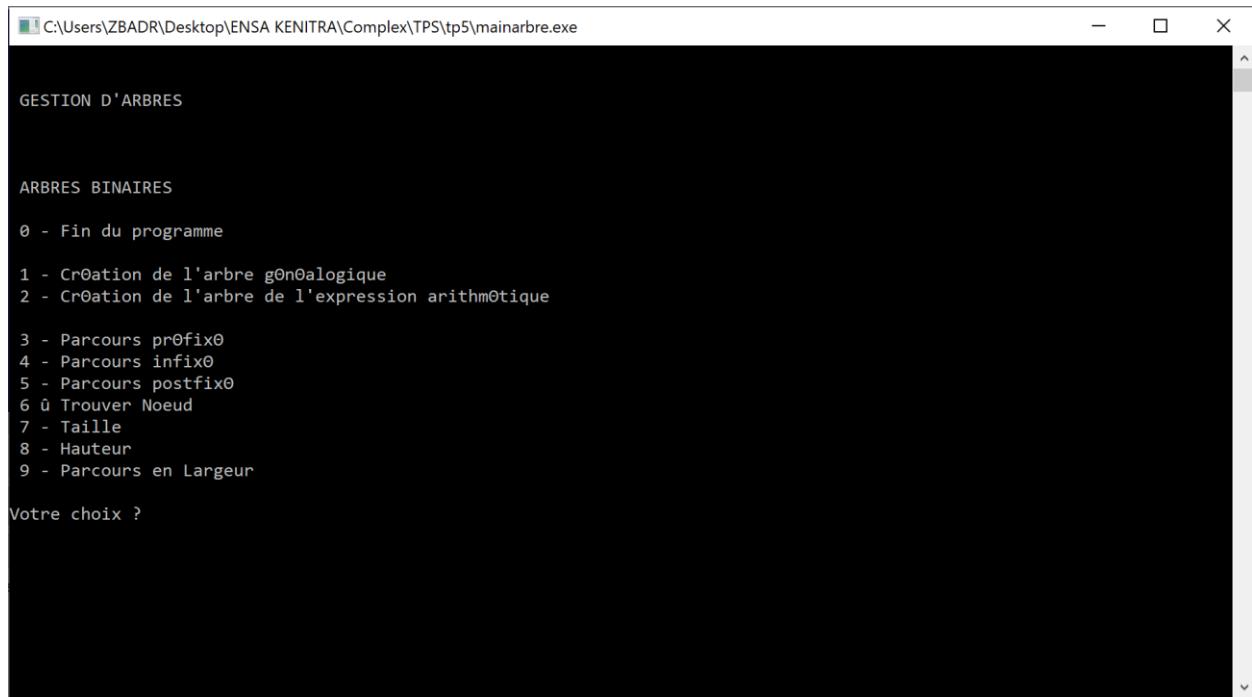
}

}

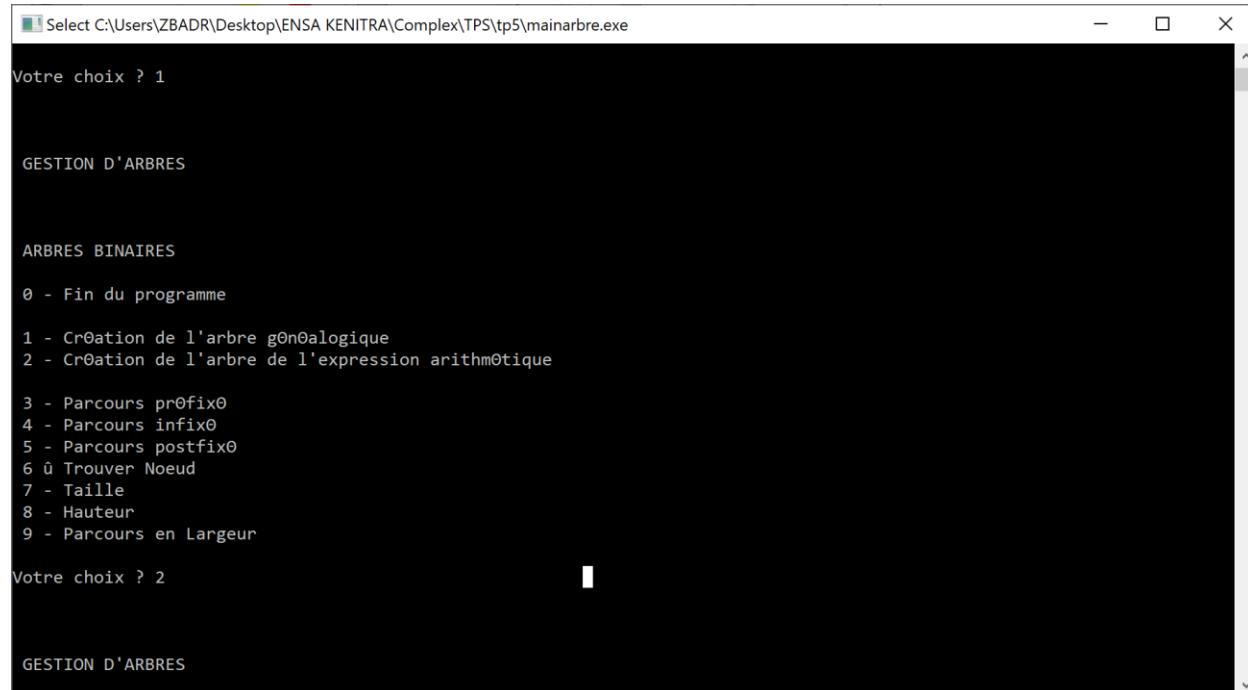
while(cod!=0);

}
```

Menu



Création de l'arbre généalogique et Création de l'arbre de l'expression arithmétique



Parcours préfixé

C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\tp5\mainarbre.exe

```
5 - Parcours postfix0
6 Ù Trouver Noeud
7 - Taille
8 - Hauteur
9 - Parcours en Largeur

Votre choix ? 3

1-pour afficher arbre genealogique / 2-pour afficher arbre arithm0tique 1
samir
kamal
yassine
hind
yasmine
sarah
karim
1-pour afficher arbre genealogique / 2-pour afficher arbre arithm0tique2
a
+
b
*
c
-
d
-
e

GESTION D'ARBRES
```

Parcours infixé

C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\tp5\mainarbre.exe

```
GESTION D'ARBRES

ARBRES BINAIRES

0 - Fin du programme
1 - Cr0ation de l'arbre g0n0alogique
2 - Cr0ation de l'arbre de l'expression arithm0tique

3 - Parcours pr0fix0
4 - Parcours infix0
5 - Parcours postfix0
6 Ù Trouver Noeud
7 - Taille
8 - Hauteur
9 - Parcours en Largeur

Votre choix ? 4

1-pour afficher arbre genealogique / 2-pour afficher arbre arithm0tique1
yassine
hind
yasmine
kamal
karim
sarah
samir
```

```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\tp5\mainarbre.exe
3 - Parcours pr0fix0
4 - Parcours infix0
5 - Parcours postfix0
6 Ù Trouver Noeud
7 - Taille
8 - Hauteur
9 - Parcours en Largeur

Votre choix ? 4

1-pour afficher arbre genealogique / 2-pour afficher arbre arithm0tique2
a
+
b
*
c
-
d
-
e

GESTION D'ARBRES

ARBRES BINAIRES

0 - Fin du programme
```

Parcours postfixé

```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\tp5\mainarbre.exe
GESTION D'ARBRES

ARBRES BINAIRES

0 - Fin du programme

1 - Cr0ation de l'arbre g0n0alogique
2 - Cr0ation de l'arbre de l'expression arithm0tique

3 - Parcours pr0fix0
4 - Parcours infix0
5 - Parcours postfix0
6 Ù Trouver Noeud
7 - Taille
8 - Hauteur
9 - Parcours en Largeur

Votre choix ? 5

1-pour afficher arbre genealogique / 2-pour afficher arbre arithm0tique1
yasmine
hind
yassine
karim
sarah
kamal
samir
```

```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\tp5\mainarbre.exe
1 - Cr0ation de l'arbre g0n0alogique
2 - Cr0ation de l'arbre de l'expression arithm0tique

3 - Parcours pr0fix0
4 - Parcours infix0
5 - Parcours postfix0
6 Ù Trouver Noeud
7 - Taille
8 - Hauteur
9 - Parcours en Largeur

Votre choix ? 5

1-pour afficher arbre genealogique / 2-pour afficher arbre arithm0tique2
a
b
+
c
d
-
*
e
-
GESTION D'ARBRES

ARBRES BINAIRES
```

Trouver Noeud

```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\tp5\mainarbre.exe
2 - Cr0ation de l'arbre de l'expression arithm0tique

3 - Parcours pr0fix0
4 - Parcours infix0
5 - Parcours postfix0
6 Ù Trouver Noeud
7 - Taille
8 - Hauteur
9 - Parcours en Largeur

Votre choix ? 6

1-rechercher un noeud ds arbre genealogique/2-rechercher un noeud ds arbre arithm0tique1
donner le noeud chercher ZAKARIA
noeud n existe pas

GESTION D'ARBRES

ARBRES BINAIRES

0 - Fin du programme

1 - Cr0ation de l'arbre g0n0alogique
2 - Cr0ation de l'arbre de l'expression arithm0tique

3 - Parcours pr0fix0
4 - Parcours infix0
5 - Parcours postfix0
```

Taille

```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\tp5\mainarbre.exe
1-rechercher un noeud ds arbre genealogique/2-rechercher un noeud ds arbre arithm0tique1
donner le noeud chercher ZAKARIA
noeud n existe pas

GESTION D'ARBRES

ARBRES BINAIRES

0 - Fin du programme

1 - Cr0ation de l'arbre g0n0alogique
2 - Cr0ation de l'arbre de l'expression arithm0tique

3 - Parcours pr0fix0
4 - Parcours infix0
5 - Parcours postfix0
6 ù Trouver Noeud
7 - Taille
8 - Hauteur
9 - Parcours en Largeur

Votre choix ? 7

1-TAILLE arbre genealogique/2-TAILLE arbre arithm0tique1
la taille est 7

GESTION D'ARBRES
```

```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\tp5\mainarbre.exe
1-TAILLE arbre genealogique/2-TAILLE arbre arithm0tique1
la taille est 7

GESTION D'ARBRES

ARBRES BINAIRES

0 - Fin du programme

1 - Cr0ation de l'arbre g0n0alogique
2 - Cr0ation de l'arbre de l'expression arithm0tique

3 - Parcours pr0fix0
4 - Parcours infix0
5 - Parcours postfix0
6 ù Trouver Noeud
7 - Taille
8 - Hauteur
9 - Parcours en Largeur

Votre choix ? 7

1-TAILLE arbre genealogique/2-TAILLE arbre arithm0tique2
la taille est 9

GESTION D'ARBRES
```

Hauteur

```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\tp5\mainarbre.exe

GESTION D'ARBRES

ARBRES BINAIRES

0 - Fin du programme
1 - Cr0ation de l'arbre g0n0alogique
2 - Cr0ation de l'arbre de l'expression arithm0tique

3 - Parcours pr0fix0
4 - Parcours infix0
5 - Parcours postfix0
6 Ù Trouver Noeud
7 - Taille
8 - Hauteur
9 - Parcours en Largeur

Votre choix ? 8

1-Hauteur arbre genealogique/2-Hauteur arbre arithm0tique1
la hauteur est 5

GESTION D'ARBRES

ARBRES BINAIRES
```

```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\tp5\mainarbre.exe

1-Hauteur arbre genealogique/2-Hauteur arbre arithm0tique1
la hauteur est 5

GESTION D'ARBRES

ARBRES BINAIRES

0 - Fin du programme
1 - Cr0ation de l'arbre g0n0alogique
2 - Cr0ation de l'arbre de l'expression arithm0tique

3 - Parcours pr0fix0
4 - Parcours infix0
5 - Parcours postfix0
6 Ù Trouver Noeud
7 - Taille
8 - Hauteur
9 - Parcours en Largeur

Votre choix ? 8

1-Hauteur arbre genealogique/2-Hauteur arbre arithm0tique2
la hauteur est 4

GESTION D'ARBRES
```

Parcours en Largeur

```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\tp5\mainarbre.exe
7 - Taille
8 - Hauteur
9 - Parcours en Largeur

Votre choix ? 9

1-parcours arbre genealogique en largeur/2-parcours arbre arithm0tique en largeur1
samir
kamal
yassine
sarah
hind
karim
yasmine

GESTION D'ARBRES

ARBRES BINAIRES

0 - Fin du programme

1 - Cr0ation de l'arbre g0n0alogique
2 - Cr0ation de l'arbre de l'expression arithm0tique

3 - Parcours pr0fix0
4 - Parcours infix0
5 - Parcours postfix0
```

```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\tp5\mainarbre.exe
6 Ù Trouver Noeud
7 - Taille
8 - Hauteur
9 - Parcours en Largeur

Votre choix ? 9

1-parcours arbre genealogique en largeur/2-parcours arbre arithm0tique en largeur2
-
*
+
-
a
b
c
d

GESTION D'ARBRES

ARBRES BINAIRES

0 - Fin du programme

1 - Cr0ation de l'arbre g0n0alogique
2 - Cr0ation de l'arbre de l'expression arithm0tique
```

TP 6

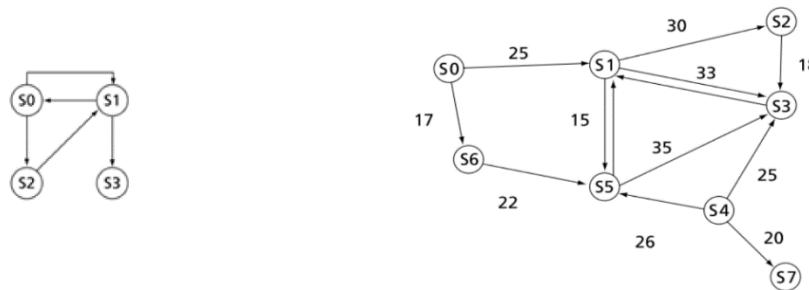
Exercice :

Implémentez les fonctions déclarées dans le fichier graphemat.h présenté ci-dessous. Ces fonctions

doivent permettre de gérer des graphes mémorisés sous forme de matrices.

Implémentez maingraphemat.c qui permettra à l'utilisateur d'appeler ces fonctions.

Les graphes à utiliser pour le test sont les suivants :



graphemat.h

```
#include <stdio.h>
#define faux 0
#define vrai 1
#define INFINI INT_MAX
typedef int booleen;
typedef char NomSom[20]; // nom d'un sommet
typedef int* Matrice;
typedef struct {
    int n; // nombre de sommets
    int nMax; // nombre max de sommets
    booleen value; // graphe valué ou non
    NomSom* nomS; // noms des sommets
    Matrice element; // existence d'un arc (i, j)
    Matrice valeur; // cout de l'arc (i, j)
    booleen* marque; // sommet marqué (visité) ou non
} GrapheMat;
GrapheMat* creerGrapheMat (int nMax, int value);
void detruireGraphe (GrapheMat* graphe);
void ajouterUnSommet (GrapheMat* graphe, NomSom nom);
void ajouterUnArc (GrapheMat* graphe, NomSom somD, NomSom somA, int cout);
void ecrireGraphe (GrapheMat* graphe);
void parcoursProfond (GrapheMat* graphe)
```

graphemat.c

```
#include <stdio.h>
#include <stdlib.h>
#include<string.h>
#include "graphemat.h"
int i,j;

static void razMarque (GrapheMat* graphe){
    for( i=0;i<graphe->n;i++)  graphe->marque[i]=faux;
}

GrapheMat* creerGrapheMat(int nMax,int value){
    //allocation de graphe
    GrapheMat* graphe=(GrapheMat*)malloc (sizeof(GrapheMat));
    graphe->n=0;
    graphe->nMax;
    graphe->value;
    graphe->nomS=(NomSom*)malloc(sizeof(NomSom) *nMax);
    graphe->marque=(boolean*)malloc(sizeof(boolean) *nMax);
    graphe->element=(int*)malloc(sizeof(int) *nMax*nMax);
    graphe->valeur=(int*)malloc(sizeof(int) *nMax*nMax);
    //initialisation par defaut

    for ( i=0; i<nMax; i++){
        for( j=0;j<nMax;j++){
            graphe->element[i*nMax+j]=faux;
            graphe->valeur[i*nMax+j]=INFINI;

        }
    }
    razMarque(graphe);
    return graphe;
}

void deteruireGraphe(GrapheMat* graphe){
    free (graphe->nomS);
    free (graphe->marque);
    free (graphe->element);
    free (graphe->valeur);
    free (graphe);
}
```

```

static int rang (GrapheMat* graphe,NomSom nom) {
    int i=0;
    boolean trouve=false;
    while((i<graphe->n) && !trouve){
        trouve=strcmp(graphe->nomS[i],nom)==0;
        if(!trouve) i++;
    }
    return trouve ? i : -1;
}

void ajouterUnSommet(GrapheMat* graphe,NomSom nom) {
    if(rang (graphe,nom)==-1){
        if(graphe->n < graphe->nMax){
            strcpy(graphe->nomS[graphe->n++],nom);

        }else{
            printf("\nNombre de sommets > %d\n",graphe->nMax);
        }
    }else {
        printf("\n%s deja defini\n",nom);
    }
}

void ajouterUnArc(GrapheMat* graphe,NomSom somD,NomSom somA,int cout){
    int nMax=graphe->nMax;
    int rd=rang(graphe,somD);
    int rg=rang(graphe,somA);
    graphe->element[rd*nMax+rg] = vrai;
    graphe->valeur[rd*nMax+rg] = cout;
}

void ecrireGraphe(GrapheMat* graphe){
    int nMax= graphe->nMax;
    for ( i=0;i<graphe->n;i++) printf("%s",graphe->nomS[i]);
    printf(";\n");
    for( i=0;i<graphe->n;i++){
        printf("\n%s:",graphe->nomS[i]);
        for( j=0;j<graphe->n;j++){
            if(graphe->element[i*nMax+j]==vrai){
                printf("%s",graphe-
>nomS[j]);

            if(graphe->value) {

```

```
    } ;
}
printf(";%");
}
```

maingraohemat.c

```
#include <stdio.h>
#include <stdlib.h>
#include<string.h>
#include "graphemat.h"

///////////
int i,j;

static void razMarque (GrapheMat* graphe){
    for( i=0;i<graphe->n;i++)  graphe->marque[i]=faux;
}

GrapheMat* creerGrapheMat(int nMax,int value){
    //allocation de graphe
    GrapheMat* graphe=(GrapheMat*)malloc (sizeof(GrapheMat));
    graphe->n=0;
    graphe->nMax;
    graphe->value;
    graphe->nomS=(NomSom*)malloc(sizeof(NomSom) *nMax);
    graphe->marque=(boolean*)malloc(sizeof(boolean) *nMax);
    graphe->element=(int*)malloc(sizeof(int) *nMax*nMax);
    graphe->valeur=(int*)malloc(sizeof(int) *nMax*nMax);
    //initialisation par defaut

    for ( i=0; i<nMax; i++){
        for( j=0;j<nMax;j++){
            graphe->element[i*nMax+j]=faux;
            graphe->valeur[i*nMax+j]=INFINI;

        }
    }
    razMarque(graphe);
    return graphe;
}

void deteruireGraphe(GrapheMat* graphe){
    free (graphe->nomS);
    free (graphe->marque);
    free (graphe->element);
    free (graphe->valeur);
    free (graphe);
}

static int rang (GrapheMat* graphe,NomSom nom){
    int i=0;
    boolean trouve=false;
```

```

        while((i<graphe->n) && !trouve) {
            trouve=strcmp(graphe->nomS[i], nom)==0;
            if(!trouve) i++;
        }
        return trouve ? i : -1;
    }

void ajouterUnSommet(GrapheMat* graphe, NomSom nom) {
    if(rang(graphe, nom)==-1) {
        if(graphe->n < graphe->nMax) {
            strcpy(graphe->nomS[graphe->n++], nom);

        } else{
            printf("\nNombre de sommets > %d\n",graphe->nMax);
        }
    } else {
        printf("\n%s deja defini\n", nom);
    }
}

void ajouterUnArc(GrapheMat* graphe, NomSom somD, NomSom somA, int cout) {
    int nMax=graphe->nMax;
    int rd=rang(graphe, somD);
    int rg=rang(graphe, somA);
    graphe->element[rd*nMax+rg] = vrai;
    graphe->valeur[rd*nMax+rg] = cout;
}

void ecrireGraphe(GrapheMat* graphe) {
    int nMax= graphe->nMax;
    for ( i=0;i<graphe->n;i++) printf("%s",graphe->nomS[i]);
    printf(";\n");
    for( i=0;i<graphe->n;i++){
        printf("\n%s:",graphe->nomS[i]);
        for( j=0;j<graphe->n;j++){
            if(graphe->element[i*nMax+j]==vrai){
                printf("%s",graphe-
>nomS[j]);
            }
            if(graphe->value){

            } ;
            }
        }
    printf(";");
}
}

```

```

}

int menu(){
    printf("*****>>>>GRAPHES<<<<*****\n");
    printf ("\n \n");
    printf("0-Fin\n");
    printf("1- Creer graphe\n");
    printf("2-Detruire graphe\n");
    printf("3-Ajouter un sommet\n");
    printf("4-Ajouter un arc\n");
    printf("5-Ecrire graphe\n");
    printf("6-Parcourir en profondeur\n");
    printf ("\n \n");
    printf("votre choix?\n");
    int cod; scanf("%d",&cod);
    return cod;
}

int main(){
    GrapheMat* G1;
    int cod;
    do{
        cod=menu();
        switch(cod){
            case 0:
                break;
            case 1:{
                G1=creerGrapheMat(4,0);

                break;
            }

            case 2:
            {
                detruireGraphe(G1);
                break;
            }
            case 3:{

                printf("donner nom ajoute\n");
                NomSom S;
                scanf("%d",S);
                ajouterUnSommet(G1,S);
                break;
            }

            case 4:{

                NomSom S1,S2;

```

```

        int arc;
        printf("donner som 1==");
        scanf("%d",&S1);
        printf("donner som 2==");
        scanf("%d",&S2);
        printf("donner arc==");
        scanf("%d",&arc);
        ajouterUnArc(G1,S1,S2,arc);
        break;
    }
    case 5:{ 
        ecrireGraphe(G1);
        break;
    }
    case 6:{ 
        printf("fonction n existe pas\n");
//        parcoursProfond(G1);
        break;
    }
}
}while(cod!=0);

}

```

Menu



```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP6\maingraohemat.exe
*****>>>>GRAPHES<<<<*****
0-Fin
1- Creer graphe
2-Detruire graphe
3-Ajouter un sommet
4-Ajouter un arc
5-Ecrire graphe
6-Parcourir en profondeur

votre choix?
```

Creer graphe



```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP6\maingraohemat.exe
*****>>>>GRAPHES<<<<*****
0-Fin
1- Creer graphe
2-Detruire graphe
3-Ajouter un sommet
4-Ajouter un arc
5-Ecrire graphe
6-Parcourir en profondeur

votre choix?1
*****>>>>GRAPHES<<<<*****

0-Fin
1- Creer graphe
2-Detruire graphe
3-Ajouter un sommet
4-Ajouter un arc
5-Ecrire graphe
6-Parcourir en profondeur

votre choix?
```

Détruire graphe

```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP6\maingraohemat.exe
4-Ajouter un arc
5-Ecrire graphe
6-Parcourir en profondeur

votre choix?1
*****>>>>GRAPHES<<<<*****>>>

0-Fin
1- Creer graphe
2-Détruire graphe
3-Ajouter un sommet
4-Ajouter un arc
5-Ecrire graphe
6-Parcourir en profondeur

votre choix?2
*****>>>>GRAPHES<<<<*****>>>

0-Fin
1- Creer graphe
2-Détruire graphe
3-Ajouter un sommet
4-Ajouter un arc
5-Ecrire graphe
6-Parcourir en profondeur

votre choix?
```

Ajouter un sommet

```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP6\maingraohemat.exe
*****>>>>GRAPHES<<<<*****>>>

0-Fin
1- Creer graphe
2-Détruire graphe
3-Ajouter un sommet
4-Ajouter un arc
5-Ecrire graphe
6-Parcourir en profondeur

votre choix?3
donner som ajoute
2

Nombre de sommets > 0
*****>>>>GRAPHES<<<<*****>>>

0-Fin
1- Creer graphe
2-Détruire graphe
3-Ajouter un sommet
4-Ajouter un arc
5-Ecrire graphe
6-Parcourir en profondeur

votre choix?3
donner som ajoute
4
```

Ajouter un arc

```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP6\maingraohemat.exe
*****>>>>GRAPHES<<<<*****
0-Fin
1- Creer graphe
2-Detruire graphe
3-Ajouter un sommet
4-Ajouter un arc
5-Ecrire graphe
6-Parcourir en profondeur

votre choix?1
*****>>>>GRAPHES<<<<*****

0-Fin
1- Creer graphe
2-Detruire graphe
3-Ajouter un sommet
4-Ajouter un arc
5-Ecrire graphe
6-Parcourir en profondeur

votre choix?4
donner som 1==6
donner som 2==2
donner arc==88
*****>>>>GRAPHES<<<<*****

0-Fin
```

Ecrire graphe

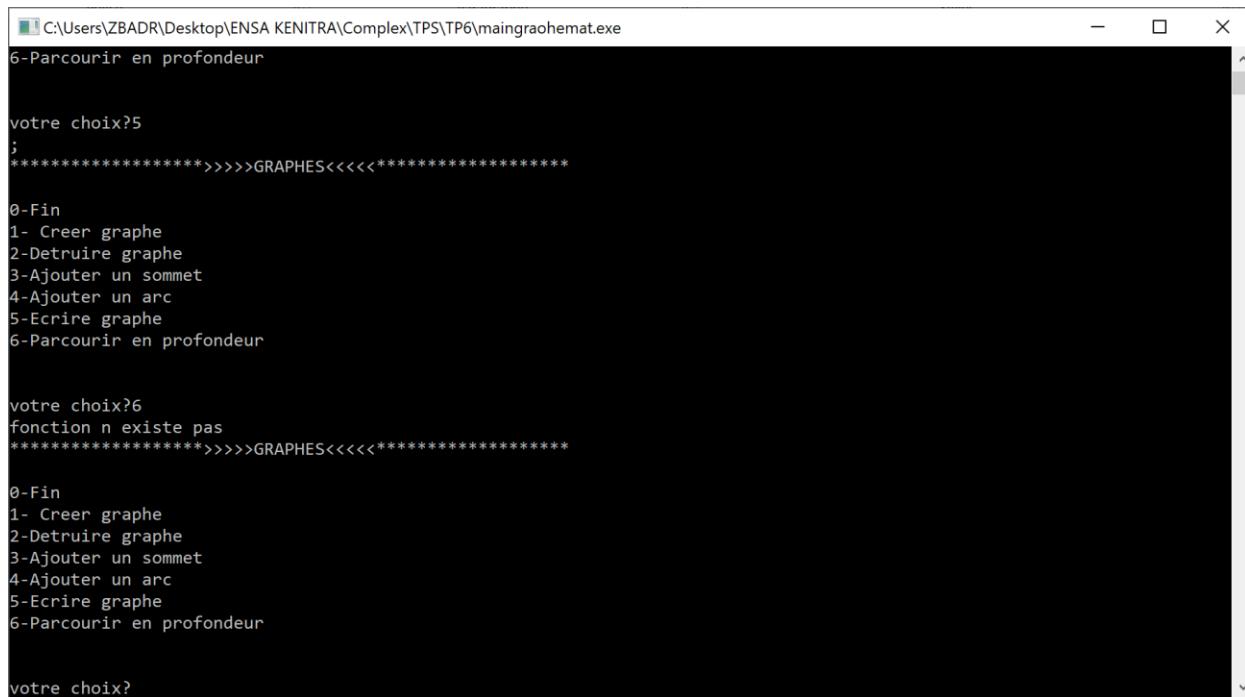
```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP6\maingraohemat.exe
*****>>>>GRAPHES<<<<*****
0-Fin
1- Creer graphe
2-Detruire graphe
3-Ajouter un sommet
4-Ajouter un arc
5-Ecrire graphe
6-Parcourir en profondeur

votre choix?5
;
*****>>>>GRAPHES<<<<*****

0-Fin
1- Creer graphe
2-Detruire graphe
3-Ajouter un sommet
4-Ajouter un arc
5-Ecrire graphe
6-Parcourir en profondeur

votre choix?
```

Parcourir en profondeur



The screenshot shows a terminal window with the following text output:

```
C:\Users\ZBADR\Desktop\ENSA KENITRA\Complex\TPS\TP6\maingraohemat.exe
6-Parcourir en profondeur

votre choix?5
;
*****>>>GRAPHES<<<*****
0-Fin
1- Creer graphe
2-Detruire graphe
3-Ajouter un sommet
4-Ajouter un arc
5-Ecrire graphe
6-Parcourir en profondeur

votre choix?6
fonction n existe pas
*****>>>GRAPHES<<<*****
0-Fin
1- Creer graphe
2-Detruire graphe
3-Ajouter un sommet
4-Ajouter un arc
5-Ecrire graphe
6-Parcourir en profondeur

votre choix?
```