void defil	er ( struct file * f) { /*2 points*/	
		•••
		•••
	* 7	
}		
nt premie	er (struct file f) { /*2 points*/	
1		
50		50

```
/****** Partie de test : test.c (2 Points) ******/
```

```
#include <stdio.h>
#include "file.h"
void main() {
 /* Déclaration d'une structure de file */
 struct file fl:
 int elem;
  char choix:
 /* Initialisation de la file */
 creerFile (.....);
do{
  printf("Tapez E ou e pour Enfiler, D ou d pour Défiler et une autre touche pour terminer \n");
  scanf(" %c", &choix);
 if ((choix=='e')) { (choix=='e')) {
     printf("donnez l'element a enfiler \n");
     scanf ("%d", &elem);
     enfiler (elem, .....);
  else if ((choix='D') || (choix=='d')){
    printf ("defilement de la valeur %d \n ", premier (.....));
    defiler (.....);
 } while ((choix='E')||(choix='e')||(choix='D')||(choix='d'));
```

## 

.....

## Exercice 1: (2+3\*2 Points)

 Sachant que les deux implémentations ci-dessous ont le même rôle, en utilisant le paradigme « diviser pour résoudre », proposez une troisième implémentation pour cette fonction.

```
int Fonction2 (int * t, unsigned n) {
int Fonction1 (int * t, unsigned n) {
  unsigned i:
                                                         int x:
                                                         if (n==1)
  int m= *t;
                                                            return *t;
  for (i=1; i<n; i++)
                                                          else {
     if (m < *(t+i))
                                                            x = Fonction 2 (t, n-1);
      m=*(t+i);
                                                            if (x > *(t+n-1))
  return m;
                                                                  return x;
                                                            else
                                                                  return *(t+n-1);
```

int Fonction	on3 (int'	t, unsign	ed n) {				
				 		 •••••	 
				 		 •••••	 
				 	********	 	 
3							

2. Calculer la complexité de chacune de ces trois implémentations.

Complexité de la Fonction1	Complexité de la Fonction	Complexité de la Fonction3
O ()	O ()	O ()



## Exercice 2: (12 points)

En se basant sur la structure de données Pile déjà vue en cours (voir interface Pile.h), nous souhaitons implémenter la structure de données File en utilisant deux piles. Pour ce faire, nous vous demandons de compléter le fichier source file.c ainsi que le fichier de test test.c

```
/****** Partie interface : pile.h *****/

struct element {
  int cle;
    struct element *suivant;
};

/*opérations exportées*/
  void creerPile (struct element**);
  unsigned pileVide (struct element *);
  int dernier (struct element *);
  void empiler (int, struct element**);
  void depiler (struct element **);
```

/\*\*\*\*\*\* Partie interface : file.h \*\*\*\*\*\*/

```
#include "pile.h"
struct file {
    struct element * P_In;
    struct element * P_Out;
};
/*opérations exportées*/
void creerFile ( struct file *f);
unsigned fileVide ( struct file f);
void enfiler ( int x , struct file * f);
void defiler ( struct file * f);
int premier ( struct file f);
```

/\*\*\*\*\*\* Partie source : file.c \*\*\*\*\*\*/

	_
#include <stdio.h></stdio.h>	1
#include <stdlib.h></stdlib.h>	
#include <assert.h></assert.h>	
#include "file.h"	J
	1
	1
void creerFile (struct file * f) { /*1 point*/	1
Part Could The Country	İ
}	
unsigned fileVide (struct file f) { /*1 point*/	Ì
man Succession and Courses were a) ( , a known .	
}	
,	
void enfiler (int x, struct file * f) {/*1 point*/	
void diffici (int x, state the 1) () I point /	
}	
2	
static void transvider () {	
/* Transvider les éléments de la pile source vers la pile de destination */ /* 3 points*/	
1	
}	