Licence **Sciences et Technologies**	**ANNEE : 2004/2005** **SESSION DE SEPTEMBRE 2005** **ETAPE :** INF2, MAT2 et MAI4 **UE :** INF102 **Epreuve :** Initiation à l'algorithmique **Date :** 2/09/2005 **Heure :** 11 H **Durée :** 1 H 30 **Documents :** Tous documents interdits **Vous devez répondre directement sur le sujet qui comporte 4 pages.** **Insérez ensuite votre réponse dans une copie d'examen comportant tous les renseignements administratifs** **Epreuve de** Mme Delest.

Indiquez votre code **d'anonymat** : N° : .

La notation tiendra compte de la clarté de l'écriture des réponses.

Barème

- Question 1 – Connaissances générales : 4 points
- Question 2 – Compréhension d'algorithme de tris : 3 points
- Question 3 – Compréhension d'algorithme récursif : 3 points
- Question 4 – Manipulation de tableaux : 6 points
- Question 5 – Manipulation de listes : 4 points

Question 1. Cochez les affirmations qui sont correctes :

¤ La boucle *tant que* ne comporte pas de test

¤ Les instructions d'une boucle *pour* sont toujours exécutées

x Un tableau passé par *référence* à une fonction n'est pas recopié dans l'environnement de la fonction.

¤ Une fonction *récursive* comporte au moins une boucle

¤ L'algorithme de tri qui s'exécute toujours en $O(n)$ est le tri à bulle.

x Dans le tri séléction, les k premiers éléments du tableau ne sont pas à leur place définitive après k itérations.

¤ Les méthodes « diviser pour régner » permettent toujours de diminuer la complexité.

x Dans une liste simplement chaînée, l'ajout d'un élément peut se faire en tête

Question 2. Soit la fonction suivante :

```
fonction mystere(ref T:tableau[1..NMax] d'entier,val N :entier) : entier ;
var i, k,ntmp,itmp :entier ;
var tmp :tableau[1..N*N]d'entiers ;
début
   init(tmp,0) ;
   ntmp=0 ;
   pour i allant de 1 à N-1 faire
      pour k allant de i+1 à N faire
         si T[i] >T[k] alors
            ntmp=ntmp+1 ;
            tmp[ntmp]=T[k] ;
         finsi ;
      finpour ;
   finpour;
   copie(tmp,T,1,ntmp ,1);
   retourner(ntmp) ;
fin ;
finfonction
```

où *init* et *copie* sont les fonctions décrites dans le cours.

1. On considère le tableau A de dimension 8 contenant la séquence 3,6,5,1,2,4,7. Quel est le résultat de l'appel mystere(A,7) ? Quel est l'effet de cet appel sur le tableau A ?
   Le résultat de l'appel est 9.
   Le tableau A est modifié car le tableau est passé par référence et on utilise la fonction copie qui copie le tableau tmp dans T. La tableau A contiendra 3,3,6,6,6,6,5,5,5.

2. Quelle sera la valeur entière maximale (en fonction de N) retournée par la fonction mystere ? Illustrez par un exemple.

   La valeur maximale retournée sera N(N-1)/2 quand les valeurs contenues dans le tableau forment une séquence décroissante. Par exemple pour N=5 et la séquence 5,4,3,2,1.

3. Que se passe-t-il si on remplace ref par val dans l'en-tête de la fonction ?

Si on remplace ref par val alors le tableau sera copié dans le contexte d'exécution de la fonction. Par suite la fonction modifiera la copie et non le tableau fournit dans l'appel. Celui-ci ne sera donc pas modifié.

4. Donnez la complexité de cette fonction en nombre de tests ? $C^<(n)=C^>(n)=C(n)=O(n^2)$

Question 3. Soit la fonction suivante :

```
fonction recurse(ref T:tableau[1..N] d'entier) :  entier ;
   fonction recurseRec(ref T:tableau[1..N] d'entier,val i :entier) : entier ;
      début
         si i=N  alors
            retourner(0)
         sinon
            si T[i]>T[N] alors
               retourner(1+recurseRec(T,i+1)) ;
            sinon
               retourner(recurseRec(T,i+1)) ;
            finsi
         finsi
      fin
   finfonction ;
   début
      retourner(recurseRec(T,1)) ;
   fin ;
finfonction
```

1. On considère le tableau A de dimension 4 contenant la séquence 3,4,1,2. Quel est le résultat de l'appel recurse(A) ? ……2……

2. Donner la suite des appels récursifs provoqués par l'appel de la question précédente ainsi que l'état du contexte de la fonction à chaque appel.

	T	*i*		
recurse(T)	*@A*			
recurseRec(T,1)	*@A,@A*	*1*		
recurseRec(T,i+1)	*@A,@A,@A*	*1,2*		
recurseRec(T,i+1)	*@A,@A,@A,@A*	*1,2,3*		
recurseRec(T,i+1)	*@A,@A,@A,@A,@A*	*1,2,3,4*		
0	*@A,@A,@A,@A*	*1,2,3*		
0	*@A,@A,@A*	*1,2*		
1	*@A,@A*	*1*		
2	*@A*			
2				

3. Le tableau est passé par référence. S'il était passé par valeur, quel serait l'effet sur la complexité en nombre d'opérations ? la complexité en mémoire?
Le tableau serait alors recopié à chaque appel récursif. Au niveau de la complexité en temps, cela ajoutera $O(N^2)$ opérations (N fois une copie dont la complexité est en O(N)).Au niveau de la mémoire de la même manière $O(N^2)$ entiers. Donc, l'algorithme qui est dans l'état actuel
   - *en O(N) en temps passerait en $O(N^2)$,*
   - *en O(N) en mémoire passerait en $O(N^2)$.*

La complexité serait donc très dégradée

Question 4. Soit un tableau B d'entiers de taille N.

1. Ecrire la fonction *compare* qui prend pour paramètre deux index *I* et *J,* le tableau *B* et dont le résultat est *vrai* si *B[I]>B[J]*.

```
fonction compare(ref B :tableau[1..N] d'entiers, val I,J :entier) :booléen ;
   début
      si I>=N ou J>=N alors
            retourner (NULL)
      sinon
           retourner(B[I]>B[J])
      finsi
   fin
finfonction
```

2. Ecrire une fonction *compteSup* qui utilise la fonction *compare* et dont le résultat est le nombre de fois où dans un tableau B, pour deux indices *I,J* tels que *I<J,* on a *B[I]>B[J]*. Par exemple si le tableau contient (3,4,1,2), le résultat est 4 ((3>1), (3>2), (4>1), (4>2)).

```
fonction compteSup(ref B :tableau[1..N] d'entiers) :entier ;
    var cpt,i,j :entier ;
    début
      cpt=0 ;
      pour i allant de 1 à N-1 faire
           pour j allant de i+1 à N faire
              si compare(B,i,j)  alors cpt=cpt+1 finsi
          finpour
       finpour
       retourner(cpt)
    fin
finfonction
```

3. Ecrire une fonction *inversion* dont le résultat est un tableau de taille maximale N*N, qui contient l'ensemble des valeurs *B[I]* pour lesquelles il existe un indice *J* tel que *I<J* et *B[I]>B[J]*. On utilisera la fonction *compare.* Par exemple la séquence (4,1,6,7,3,8,2,5) conduit au tableau (4,4,4,6,6,6,7,7,7,3,8,8).

```
fonction inversion(ref B :tableau[1..N] d'entiers) : tableau[1..N*N] d'entiers;
   var,itmp,i,j :entier ;
   var tmp :tableau[1..N*N] d'entiers;
   début
     itmp=0 ;
     pour i allant de 1 à N-1 faire
          pour j allant de i+1 à N faire
             si compare(B,i,j)  alors
               itmp=itmp+1 ;tmp[itmp]=B[i]
             finsi
          finpour
      finpour
     retourner(tmp)
     fin
finfonction
```

Question 5.

1. Quelles différences y-a-t-il entre la suppression d'un élément dans une simplement liste chaînée et la suppression dans une liste doublement chaînée ? On illustrera par un exemple.

   Dans les listes simplement chaînée il est possible de supprimer en O(1) l'élément en tête de liste et un élément suivant un élément pointé lors du parcours de la liste. Par exemple si la liste est (2,3,5,1), on peut supprimer l'élément 2 en O(1). Si on parcourt la liste et que l'on est positionné sur l'élément 3 on peut supprimer l'élément 5 en O(1) mais le parcours de la liste est en O(n). Dans les listes doublement chaînées, on peut de plus supprimer le dernier élément en O(1). Par exemple dans la liste précédente on peut supprimer l'élément 1 en O(1). On peut également supprimer l'élément précédent un élément pointé lors du parcours de la liste puisqu'on dispose de la primitive précédent.

2. Soit L une liste doublement chaînée d'entiers. Que contiennent L et P après la séquence d'instructions suivante

```
var L :listeDC d'entier ;
var P :^entier ;
creerListe(L) ;
ajouterEnTete(7,L) ;
ajouterEnTete(16,L) ;
P=suivant(L,premier(L)) ;
ajouterAprès(4,L,P) ;
P=dernier(P);
ajouterAprès(12,L,P) ;
P=precedent(dernier(L)) ;
supprimer(L,P) ;
P=suivant(L,premier(L)) ;
```

P=@7……
L=…16,7,12…..……

3. Ecrire une fonction *compteSupliste* dont le résultat est le nombre de fois où dans une liste L simplement chaînée, le premier élément est supérieur à un de ses suivants. Par exemple si L=(3,4,1,2) alors le résultat est 2 (car 3>1 et 3>2) .

```
fonction compteSupListe(ref L :listeSC d'entiers) :entier ;
   var compte,p :entier ;
   var Q :^entier ;
   fonction compare (ref L :listeSC d'entiers,val p :entier,Q :^entier) :booléen ;
      début
        retourner(p> contenu(L,Q)
      fin
   finfonction
   début
     compte=0 ;
      si non(listeVide(L)) alors
              Q=suivant(premier(L));
              p=contenu(premier(L) ;
              tant que non(Q==NIL) faire
                  si compare(L,p,Q) alors
                    compte=compte+1 ;
                  finsi
                  Q=suivant(L,Q) ;
              fintantque
      finsi
      retourner(compte) ;
   fin
finfonction
```

FIN