Licence Sciences et Technologies	**ANNEE : 2005/2006** **SESSION DE SEPTEMBRE 2006** **ETAPE :** INF2 MAT2 MAI4 **UE :** INF102 **Epreuve :** Initiation à l'algorithmique **Date :** 6 Septembre 2006 **Heure :** 11 H 00 **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 : 3 points
- Question 3 – Compréhension d'algorithme récursif : 2 points
- Question 4 – Ecriture de fonctions : 3 points
- Question 5 – Algorithme de tri : 4 Points
- Question 6 – 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
☒ Pour un tableau, l'accès aux éléments s'effectue directement par la clé.
☒ Les méthodes diviser pour régner consistent à appliquer le même algorithme récursivement.
☐ La complexité maximale d'un algorithme de tri est en $O(n^2)$
☐ Dans le tri fusion, les k premiers éléments du tableau sont à leur place après k appels récursifs
☒ Une liste est une table d'association de taille non constante.
☐ Dans une liste doublement chaînée de n éléments, l'accès à l'avant-dernier élément se fait en $O(n)$

Question 2. Soit la fonction suivante :

```
fonction mystere(ref T:tableau[1..N] d'entier,val N :entier) : entier ;
    var k,itmp,max,lmax :entier ;
    début
        max=1 ;
        itmp=1 ;
        lmax=1 ;
        pour k allant de 2 à N faire
            si T[k]== T[k-1] alors
                lmax=lmax+1
            sinon
                si lmax>max alors
                    itmp=k-1 ;
                    max=lmax
                finsi
                lmax=1 ;
            finsi
        finpour
        si lmax>max alors
            T[1]=T[N] ;retourner(lmax)
        sinon
            T[1]=T[itmp] ;retourner(max)
        finsi
    fin ;
finfonction
```

1. On considère le tableau A de dimension 6 contenant la séquence 3,2,5,4,6,1. Quel est le résultat de l'appel mystere(A,6) ? Quel est l'effet de cet appel sur le tableau A ?

La fonction mystère retourne la valeur 1. Cet appel ne modifie pas le tableau. En effet, cette fonction détecte une séquence de nombres identiques et stocke la première valeur la plus fréquente dans T[1]. Ici, tous les nombres sont différents donc la première valeur la plus fréquente est 3 qui sera stockée dans T[1].

2. Quelle sera la valeur entière maximale retournée par la fonction si le tableau lors de l'appel a pour taille N? Illustrez par un exemple. Le tableau est-il être modifié dans ce cas? Donnez un exemple pour lequel le tableau serait modifié.

La valeur maximale est N. Par exemple si N=6, on peut choisir T=[3,3,3,3,3,3]. Le tableau dans ce cas n'est pas modifié. Le tableau est modifié si la valeur la plus fréquente n'est pas dans T[1]. Par exemple pour N=6 et T=[4,3,3,3,3,3], le résultat est 5 et le tableau final T=[3,3,3,3,3,3].

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

Dans ce cas, le tableau T n'est jamais modifié puisque la fonction travaille sur une copie de T.

4. Donnez la complexité de cette fonction en nombre de tests ? ……O(N)…….

Question 3. Soit la fonction suivante définie pour N>2:

```
fonction recurse(ref T:tableau[1..N] d'entier ; val N,p entier) :  entier ;
    fonction recurseRec(ref T:tableau[1..N] d'entier,val k :entier) : entier ;
        début
            si (k==N+1) alors
                retourner(0) ;
            sinon
                si T[k]==p*T[k-1] +T[k-2] alors
                    retourner(1+recurseRec(T,k+1));
                sinon
                    retourner(recurseRec(T,k+1))
                finsi
            finsi
        fin
    finfonction ;
    début
        retourner (recurseRec(T,3)) ;
    fin ;
finfonction
```

1. On considère le tableau A de dimension 5 contenant la séquence 1,1,2,3,5. Quel est le résultat de l'appel recurse(A,5,1) ? ……3……

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	*k*	*Val. Fonct*
recurseRec(T,3)	*@A*	*3*	
recurseRec(T,4)	*@A,@A*	*3,4*	
recurseRec(T,5)	*@A,@A,@A*	*3,4,5*	
recurseRec(T,6)	*@A,@A,@A,@A*	*3,4,5,6*	*0*
→0	*@A,@A,@A*	*3,4,5*	*1*
→1	*@A,@A*	*3,4*	*2*
→2	*@A*	*3*	*3*
→3			

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

1. Ecrire la fonction *poc* qui prend pour paramètre un index *K* et le tableau *B* et dont le résultat est *vrai* si *B[K] = B[K-1]+B[k-2]* ou si *B[K] = B[K-1]* .

```
fonction poc(ref B :tableau[1..N]d'entiers ;val K :entier) :booléen ;
  début
    retourner((K>1) et (K<=N) et ((B[K] = = B[K-1]+B[K-2]) ou (B[K]= = B[K-1])))
  fin
finfonction
```

2. Ecrire une fonction *comptePoc* qui utilise la fonction *poc* et dont le résultat est le nombre de fois où un entier est égal à la somme des deux entiers qui le précèdent ou à l'entier qui le précède dans le tableau. Par exemple si le tableau contient (0,1,0,0,0,0,1,1,0,1), le résultat est 5.

```
fonction comptePoc(ref B :tableau[1..N]d'entiers) :entier ;
  var i,c :entiers ;
  début
    si B[1]==B[2] alors
      c =1 ;
    sinon
      c=0
    finsi
    pour i allant de 3 à N faire
      si poc(B,i) alors c=c+1 finsi;
    finpour
    retourner(c)
  fin
finfonction
```

Question 5. Soit un tableau T de dimension N de structures *Info* définies de la manière suivante :

```
Info :structure
        Taille :entier ;
        Poids :entier ;
finstructure
```

Par exemple, pour N=3, un exemple de tableau T est :

	Taille	Poids
T[1]=	150	60
T[2]=	185	95
T[3]=	150	45

Donner les modifications à apporter à l'une des fonctions de tri décrite dans le cours, pour qu'elle reçoive en entrée un tel tableau et dont le résultat sera ce même tableau trié suivant le champ *Taille* et en cas d'égalité trié suivant les champs *Poids*.

Dans tout algorithme de tri, il existe un test permettant de comparer deux éléments. Par exemple, pour le tri à sélection, dans la fonction minimumSoustableau, l'instruction

« si T[i]<T[sauv] alors »

permet cette comparaison. Il suffit donc d'écrire une fonction booléenne « compare » qui permettra la comparaison de deux éléments dans ce contexte. L'instruction précédente devient

« si compare(T,i, sauv) alors ».

La fonction compare est écrite ci-dessous.

```
fonction compare(ref B :tableau[1..N]d'Info ;val i,sauv : entier) :booléen ;
  début
    retourner((T[i].Taille< T[sauv].Taille)ou(T[i].Taille==T[sauv].Taille et(T[i].Poids< T[sauv].Poids))
  fin
finfonction
```

Il faut de plus modifier les en-têtes des fonctions !

```
fonction minimumSoustableau(ref T:tableau[1..N] d'Info, val Imin,Imax:entier):entier;
fonction triSelection(ref T:tableau[1..N] d'Info):vide;
```

Question 6.

1. Soit L une liste doublement chaînée d'entiers.

   - Que contient L et P après la séquence d'instructions suivante

```
var L :listeSC d'entier ;
var P :^entier ;
creerListe(L) ;
insererEnTete(6,L) ;
insererEnTete(8,L) ;
P=premier(L) ;
insererAprès (7,L,P) ;
P=premier(L) ;
insererAprès(4,L,P) ;
P=suivant(L,P) ;
supprimerAprès(L,P) ; ;
P=premier(L) ;
supprimerAprès(L,P) ;
P=suivant(L,premier(L)) ;
```

P=……@6……
Contenu(P)=…6……
L=8,6……

   - Ecrire une fonction *comptepocListe* dont le résultat est le nombre de fois où un entier est égal à la somme des deux entiers qui le précèdent ou à l'entier qui le précède dans la liste L. On s'inspirera de la question 4.

```
fonction poc(ref B : ListeDC d'entiers ;val K :^entier) :booléen ;
    var p :^entier ;
   début
      p :=precedent(B,K) ;
      retourner((contenu(K)= =contenu(p)+contenu(precedent(B,p))ou(contenu(K)= =contenu(p)))
    fin
finfonction
fonction comptePoc(ref B :ListeDCd'entiers) :entier ;
    var i,c :entiers ;
    var p:^entier ;
    début
      c=0 ;
      p :=premier(B);
      si  non( estDernier(B,p)) alors
      si contenu(p)==contenu(suivant(B,p)) alors
          c =1 ;
      finsi ;
      p=suivant(B,p) ;
      si  non(estDernier(B,p)) alors
          p=suivant(B,p) ;
          tant que p<>NIL faire
              si poc(B,p) alors c=c+1 finsi;
           fintantque
       finsi
      retourner(c)
    fin
finfonction
```

FIN