Licence Sciences et Technologies	**ANNEE : 2006/2007** **SESSION DE MAI 2007** **ETAPE :** *INF2 MAT2 MAI4* **UE :** INF102 **Epreuve :** Initiation à l'algorithmique **Date :** *11 Mai 2007* **Heure :** *8 H 30* **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** M^me^ 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 : 5 points
- Question 3 – Compréhension d'algorithme récursif : 4 points
- Question 4 – Ecriture de fonctions : 3 points
- Question 5 – Manipulation de listes : 4 Points

Question 1. Cochez les affirmations qui sont correctes :

☐ Après une boucle *pour*, l'indice de boucle a pour valeur la valeur finale de la boucle.
☐ Un *indice de boucle* est une variable sur laquelle on ne peut qu'additionner la valeur 1
☒ Une variable passée par *valeur* à une fonction est recopiée dans l'environnement de la fonction.
☐ La récursivité est un mécanisme qui permet toujours d'obtenir une meilleure complexité en mémoire
☒ La complexité maximale des algorithmes de tri vu en cours est $O(n^2)$
☒Dans le tri sélection en ordre croissant, k éléments du tableau sont à leur place après k itérations
☐ Les listes doublement chainées ont une complexité mémoire plus élevée que les listes simplement chainées.
☒ L'élément d'un tableau peut être une structure

Question 2. Soit la fonction suivante :

```
fonction mystere(ref T:tableau[1..NMAX] d'entier, val x :entier, ref N : entier) : vide ;
    var iT,cpt, q,r,v :entier ;
    début
        iT=2;v=x ;
        tant que v !=1 faire
            cpt=0 ; q=v ;
            répéter
                v =q ;
                q=QuotientEntier (v,iT) ;
                r =Modulo(v, iT) ;
                cpt=cpt+1 ;
            jusqu'à r!=0;
            T[iT] =cpt-1 ;
            iT=iT+1;
        fintantque ;
        T[1]=iT-1;
        N=iT-1 ;
    fin ;
finfonction
```

où *QuotientEntier* et *Modulo* sont les fonctions vues en TD.

1. On considère le tableau A de dimension 8 contenant la séquence 2,5,7,4,3,6,8,1. Quel est le résultat de l'appel mystere(A, 12, 8) ? Quelle est la valeur de N après cet appel ?
   *On a dans A la séquence 3, 2, 1, 4, 3, 6, 8, 7. Pour information, l'algorithme calcule la décomposition en nombre premier de 12 et la range dans le tableau A puisque A est passé en référence. Après exécution A[1] contient le nombre premier le plus grand de la décomposition (ici 3) et A[i] contient la puissance de i dans 12 (12 = 2^2*3). La valeur de N est 3.*
2. L'appel à la fonction mystere conduit à une erreur dans le cas où x=0, pourquoi ? Modifiez l'entête et le code de la fonction que cela ne se produise pas.
   Si x=0 alors il y a erreur puisque r ne sera jamais non nul, la boucle repeter est donc infinie. Pour corriger ce problème, on modifie l'en tête de la fonction :
   fonction mystere(ref T:tableau[1..NMAX] d'entier,val x :entier, ref N :entier) : booléen ;
   et on ajoute le test si x==0 alors retourner(faux) ; finsi ;
   De plus, il faut ajouter l'instruction retourner(vrai) avant le fin.
3. Pour prendre en compte cette modification, est-il utile ou obligatoire de modifier l'en-tête de la fonction ? Justifier
   Il est utile de modifier l'en-tête de la fonction mais non obligatoire, on pourrait par exemple choisir de mettre la valeur 0 dans T[1]. Pour information, la valeur 0 serait impossible dans une décomposition en nombre premier.
4. Expliquez et donnez la complexité maximale de cette fonction en nombre de tests lorsque x est un nombre premier. On rappelle qu'un nombre premier n'est divisible que par un et par lui-même.
   *La complexité maximale est O(x). En effet, la boucle interne peut ne faire qu'un seul tour (pour information cas où x est premier). La complexité est conditionnée par l'index iT qui prendra les valeurs entre 2 et x. Au total, il y aura donc 3*x+1 d'où la complexité annoncée.*

Question 3. Soit la fonction suivante :

```
fonction recurse(val H,J :entier) :  entier ;
    var r : entier ;
    début
        si (H==1) alors
            retourner(0) ;
        sinon
            r=Modulo(H,J) ;
           si r==0 alors
                retourner(1+recurse(QuotientEntier(H,J),J))
            sinon
                retourner(0) ;
            finsi
        finsi
    fin ;
finfonction
```

1. Quel est le résultat de l'appel recurse(108,3) ? *…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.

	r	*H*	*J*	*Val.Fonc*
recurse(108,3)	*0*	*108*	*3*	
recurse(36,3)	*0,0*	*108,36*	*3,3*	
recurse(12,3)	*0,0,0*	*108,36,12*	*3,3,3*	
recurse(4,3)	*0,0,0,1*	*108,36,12,4*	*3,3,3,3*	
->0				*0*
	0,0,0	*108,36,12*	*3,3,3*	
->1				*1*
	0,0	*108,36*	*3,3*	
->2				*2*
	0	*108*	*3*	
->3				*3*

3. Réécrire la fonction de la question 2 en remplaçant la boucle « répéter » par un appel à la fonction recurse.

```
fonction mystere(ref T:tableau[1..NMAX] d'entier,val x :entier, ref N : entier) : vide ;
    var iT,cpt, :entier ;
    début
        iT=2;
        tant que x !=1 faire
            cpt=recurse(x, iT) ;
            x =x/Puissance(iT, cpt);
            T[iT] =cpt;
            iT=iT+1
        fintantque ;
        T[1]=iT-1;
     fin ;
finfonction
```

où *Puissance(a, n) calcule* a^n

4. Comparer la complexité en mémoire de cette nouvelle fonction par rapport à la question 2 ? en nombre de tests ? Justifiez.
   La complexité en nombre de test est inchangé car si la boucle répéter fait k tours il y aura k appels récursifs et donc O(k) tests. Par contre la complexité de la fonction recurse en mémoire est plus importante puisque chaque appel récursif donne lieu à la création d'un environnement.

Question 4.

Soit *T* un tableau de dimension *N* contenant des nombres complexes. Donnez une définition du type complexe qui permet de décrire un nombre complexe. Ecrire une fonction *plusQueDeux* qui modifie le tableau *T* en supprimant les nombres complexes de module strictement supérieur à 2 et retourne le nombre d'éléments ayant cette propriété.

On choisit la représentation d'un nombre complexe par son module et argument, type « complexeExp » vu en TD.

```
type complexeExp=structure
                    rho,teta :réel
                finstructure

fonction plusQueDeux (ref T:tableau[1..NMAX] de complexeExp) : entier ;
    var iT,cpt,j :entier ;
    début
        cpt=0;j=1;
        pour iT allant de 1 à NMAX faire
            si T[iT].rho<=2 alors
                T[j]=T[iT] ;
                j=j+1 ;
            sinon
                cpt=cpt+1;
            Finsi
        Finpour
        retourner(cpt) ;
     fin ;
finfonction
```

Question 5.

1. Quelle est l'utilité des listes pour gérer des polynômes à plusieurs variables ? Est-il nécessaire de choisir des listes doublement chainées.

Pour les polynômes à une seule variable cela permet de ne pas limiter le degré. Il suffit de prévoir des cellules dont le champ structure contient le coefficient et la puissance de la variable. Lorsqu'il s'agit de polynômes à plusieurs variables c'est incontournable car chaque coefficient est lui même un polynôme à une ou plusieurs variables. La structure de donnée est donc récursive et dans ce cas les listes sont particulièrement adaptées. Il n'est pas par contre nécessaire de choisir des listes doublement chainées puisque les opérations de bases ne nécessitent pas des va-et-vient dans la liste.

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

```
var L :listeDC d'entier ;
var P :^entier ;
creerListe(L) ;
insererEnTete(12,L) ;
P=premier(L) ;
insererAvant (4,L,P) ;
insererAprès(6,L,P) ;
P=Dernier(L) ;
insérerAvant(5,L,P) ;
insérerAprès(7,L,P) ;
supprimerAprès(L,P) ;
P=précédent(L,dernier(L)) ;
```

P=……*adresse de l'avant dernier (@5)*….

Contenu(P)= *5*…….

L=…*4,12,5,6*…….

   - Ecrire une fonction *decompose* dont le résultat est une liste contenant les nombres premiers qui divisent un entier *x*. On rappelle qu'un nombre premier n'est divisible que par un et par lui-même. Par exemple si x=20, la liste est (5,2).

Il suffit de s'inspirer de la fonction de l'exercice 2.

```
fonction decompose (val x :entier) :listeSC d'entiers ;
    var L: listeSC d'entier;
    var q,r,v,iT,cpt :entier ;
    début
        iT=2 ;v=x;
        creerListe(L) ;
        tant que v !=1 faire
            cpt=0 ; q=v ;
            répéter
                v =q ;
                q=QuotientEntier(v, iT) ;
                r =Modulo(v, iT);
                cpt=cpt+1 ;
            jusqu'à r!=0;
            si cpt>1 alors
                insererEnTête(iT,L);
            finsi ;
            iT=iT+1
        fintantque ;
        retourner(L);
    fin ;
finfonction                                                    FIN
```