**TELECOM Lille 1** **9 Avril 2009**

Nom : Ben Amor Prénom : Boulbaba N° : Durée : 2 heures

**QROC**

**Module A31TTN**

Structures de données

*Aucun document n'est autorisé, les réponses sont à donner sur le présent document.*

## Exercice 1 – Récursivité

On considère la fonction ***Mystere*** définie par :

```
int Mystere(int X, int N)
int a;
{
        if (N= =0) return 1 ;
        else
        {
                a=Mystere(X,N/2);
                if (N%2 ==0) /* l'opérateur % retourne le reste de la division entière*/
                        return a*a;
                else
                        return X*a*a;
        }
}
```

**1)** Que vaut ***Mystere***(4,2) ? donnez une trace d'exécution de cette fonction sous la forme d'un arbre. Que fait cette fonction ?

Mystere (4,2) [X=4, N=2] → 4*4 = 16

a = Mystere (4, 1) = 4

Mystere (4,1) → return 4 * 1 * 1 ;

a = Mystere (4, 0) = 1

Mystere (4, 0)

return 1 ;

cette fonction Mystère (X, Y) retourne $X^Y$.

**2)** Sans faire de calcul de complexité, pouvez vous donner une idée sur la complexité de cette fonction. Expliquez ?

Cette fonction présente l'inconvénient de générer une explosion combinatoire quand N devient grand. Le phénomène est le même observé dans d'autres fonctions récursives (Fibonnacci, Hanoï, etc.)

**Exercice 2 – Calcul de complexité**

On considère l'algorithme suivant :

```
Algorithme AfficheTemp (t : tableau d'entiers ; n : entier)
i, temp : entier
Début
  temp := t[1]
  Pour i := 2 à n faire
    Si (t[i] > temp)
          Alors temp := t[i]
    Fin si
  Fin Pour
  Afficher (temp)
Fin
```

**1)** Expliquez que fait cet algorithme. Donnez un tournage à la main sur l'exemple 4 2 5 1 3.

Cet algorithme trouve et affiche le maximum d'un tableau donné en entré

AfficheTemp(t,5) Affiche 5.

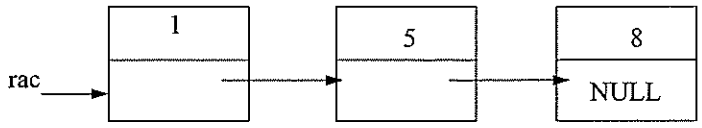
- Dans le tournage à la main, il faut donner les valeurs de i et de temp à chaque itération.

**2)** Calculez sa complexité. Expliquez votre raisonnement (au mieux, en moyenne et au pire des cas).

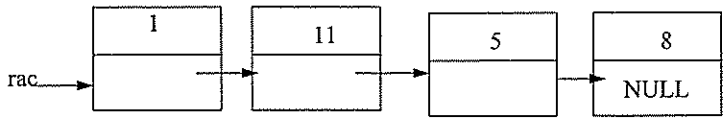
- Au pire des cas, l'algo. trouve le maximum à la dernière case. Sa complexité est de $O(n)$
- Au mieux, l'algo. trouve le maximum à la 1ère case, c'est de complexité $O(1)$
- En moyenne, il est de complexité $O(\frac{n}{2}) \equiv O(n)$

**Exercice 3 – Listes chaînées**

On considère la structure chaînée initiale suivante :



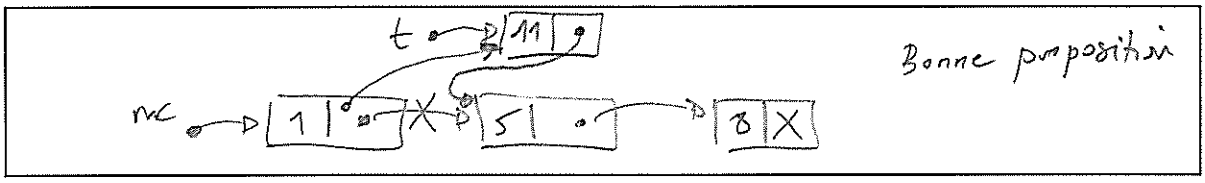

Où rac désigne le pointeur de début de cette liste. On veut créer la liste suivante :



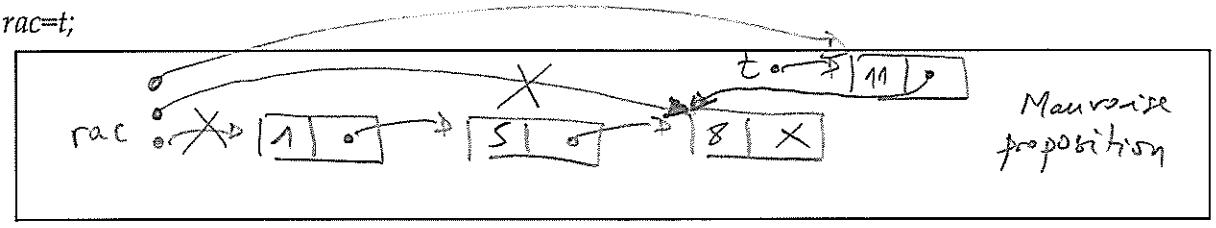

Parmi les cinq portions de codes proposées une seule réalise cette transformation (répondre et expliquez dans le cadre en dessous de chaque proposition). Toutes ont en commun les déclarations suivantes :

```
typedef struct list_noeud *LISTE;
typedef struct list_noeud {
        int valeur;
        struct list_noeud *suiv;
        };
LISTE rac;
```

```
/******************Proposition 1******************/
LISTE t;
t=(LISTE)malloc(sizeof(list_noeud));
t->suiv=rac->suiv;
t->valeur=11;
rac->suiv=t;
```

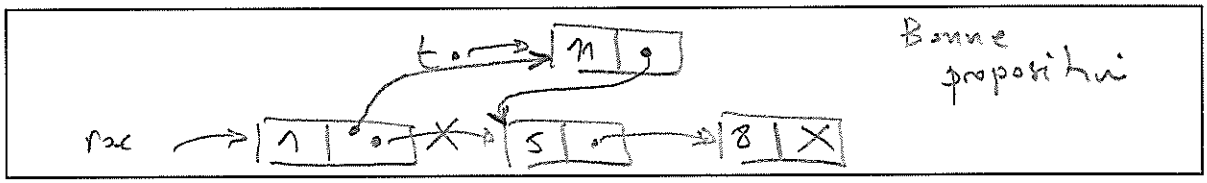



```
/*****************Proposition 2 ********************/
LISTE t;
rac=rac->suiv->suiv;
t=(LISTE)malloc(sizeof(list_noeud));
t->suiv=rac;
t->valeur=11;
rac=t;
```

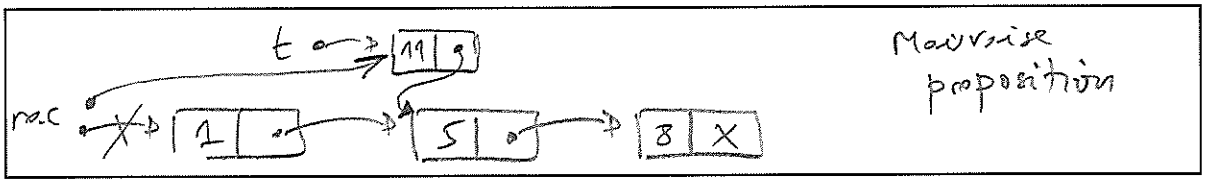



```
/******************proposition 3 ******************/
LISTE t;
t=(LISTE)malloc(sizeof(list_noeud));
t->suiv=rac->suiv;
```

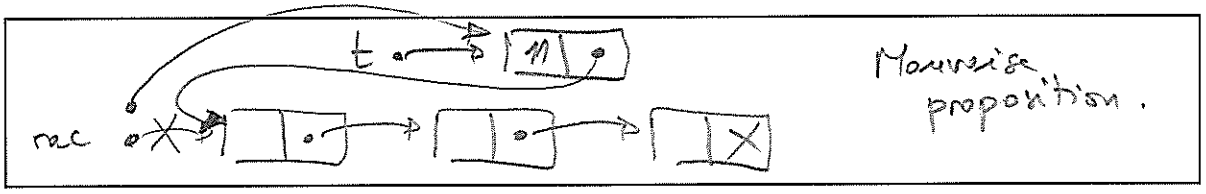
*t->valeur=11;*
*rac->suiv=t;*




*/**************proposition 4************************/*
*LISTE t;*
*t=(LISTE)malloc(sizeof(list_noeud));*
*t->suiv=rac->suiv;*
*t->valeur=11;*
*rac=t;*



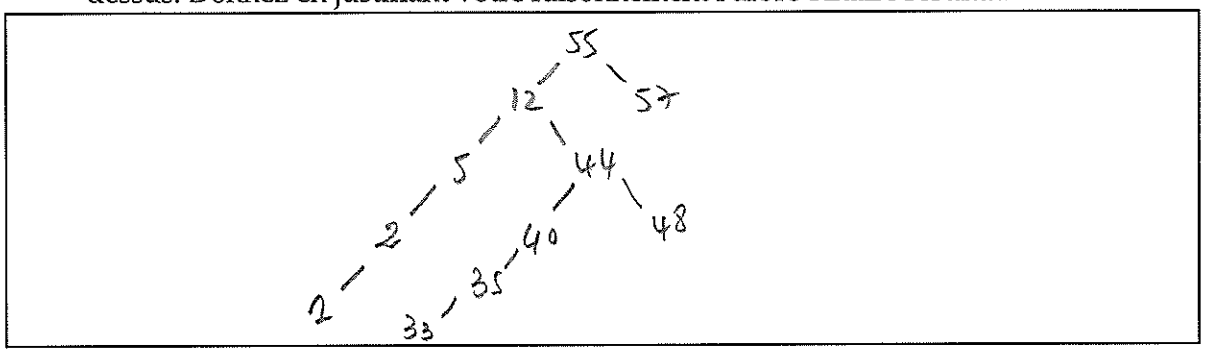

*/************************proposition 5************************/*
*LISTE t;*
*t=(LISTE)malloc(sizeof(list_noeud));*
*t->suiv=rac;*
*t->valeur=11;*
*rac=t;*




**Exercice 3 – Les arbres binaires**

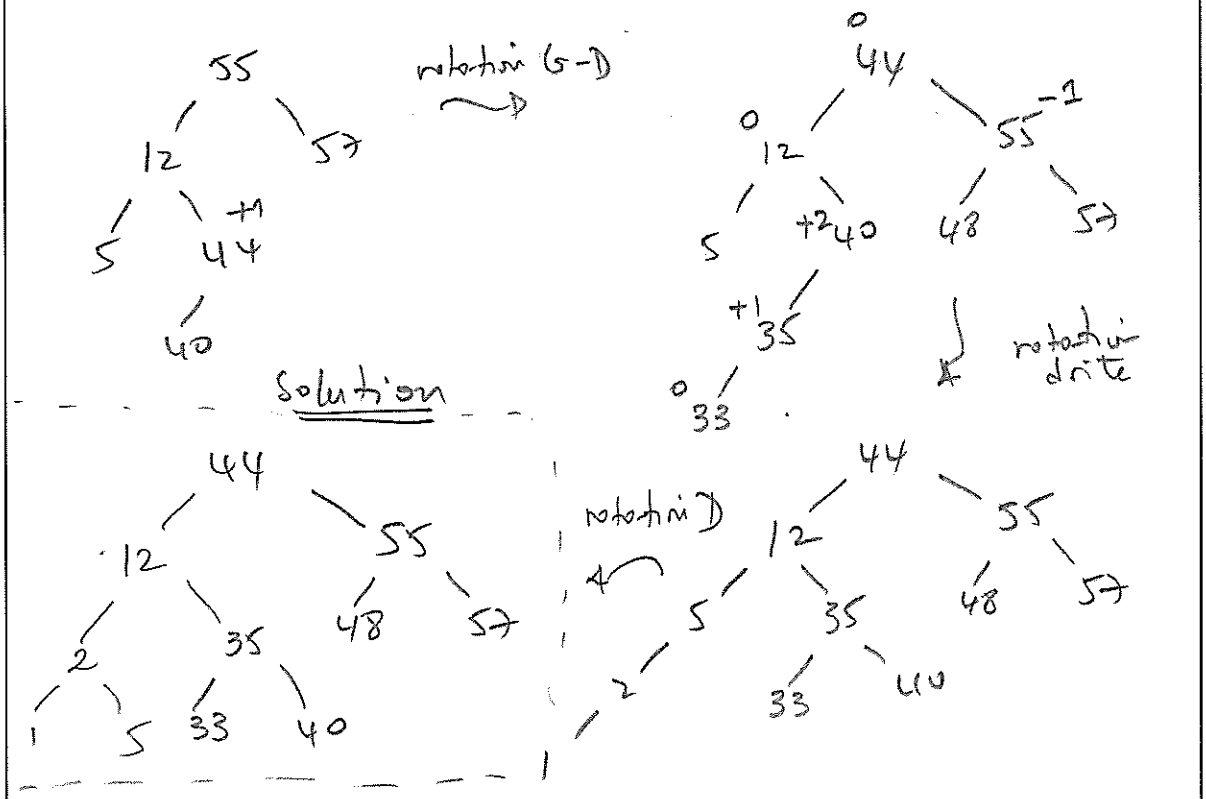
On considère la suite de nombres suivante : 55, 57, 12, 44, 5, 40, 48, 35, 33, 2, 1.

1) Partant d'un arbre binaire vide, on ajoute successivement les valeurs de la suite ci-dessus. Donnez en justifiant votre raisonnement l'arbre binaire résultat.



2) Vérifier si l'arbre obtenu est équilibré, s'il n'est pas équilibré, équilibrez le. Expliquez les opérations de rotation effectuées.

l'arbre binaire obtenu n'est pas equilibré.
La version equilibrée est :




3) Proposez une structure de donnée qui permet une représentation chaînée de cet arbre.

```
typedef struct Noeud {
    int val;
    Noeud *pfg; *pfd; } Noeud;
typedef struct Noeud *Arbre;
```

4) Sur l'arbre obtenu, calculer manuellement sa hauteur. Proposez une fonction qui retourne la hauteur d'un arbre.

La hauteur de l'arbre est 3.

```
int Hauteur (Arbre A) {
    if (A==NULL)
        return 0;
    else return 1+ max (Hauteur (A->pfg), Hauteur (A->pfd));
}
```

5) Proposez une fonction récursive qui permet de calculer la somme des nœuds.

```
int Somme (Arbre A) {
    if (A==NULL)
        return 0;
    else
        return A->val + Somme (A->pfg) + Somme (A->pfd) ;
}
```

6) Donnez les listes infixe et préfixe de cet arbre binaire.

liste infixe : 1, 2, 5, 12, 33, 35, 40, 44, 48, 55, 57.

liste préfixe : 44, 12, 2, 1, 5, 35, 33, 40, 55, 48, 57.

7) Proposez une fonction qui permet de tester si deux arbres sont différents.

```
int Different (Arbre A1, Arbre A2) {
    if (A1==NULL) && (A2==NULL) return 1;
    if (A1==NULL) && (A2!=NULL) return 0;
    if (A1!=NULL) && (A2==NULL) return 0;
    return (A1->val != A2->val) &&
           Different (A1->pfg, A2->pfg) &&
           Different (A1->pfd, A2->pfg) ;
}
// cette fonction retourne 0 si les arbres sont différents
```

**Bonne chance.**