

Nom :

Prénom :

N° :

Durée : 2 heures

QROC Module A31TTN - Structures de données

Aucun document n'est autorisé, les réponses sont à donner sur la présente feuille.

Exercice 1 – Récursivité

On considère les trois fonctions f , g et h suivantes :

<pre>int f(int a, int b) { int r; while (1) { r = a % b; if (r == 0) return b; a = b; b = r; } }</pre>	<pre>int g(int a, int b) { int r; while (b > 0) { r = a % b; a = b; b = r; } return a; }</pre>	<pre>int h(int a, int b) { int r; r = a % b; if (r == 0) return b; else return h(b, r); }</pre>
--	---	---

1) Donnez leurs résultats retournés par ces fonctions pour $a=96$ et $b=81$. Que pouvez-vous remarquer? Expliquez, en particulier, la trace d'exécution de la fonction h ?

2) Que font alors les fonctions f , g et h ci-dessus ? Donnez des noms significatifs à ces fonctions?

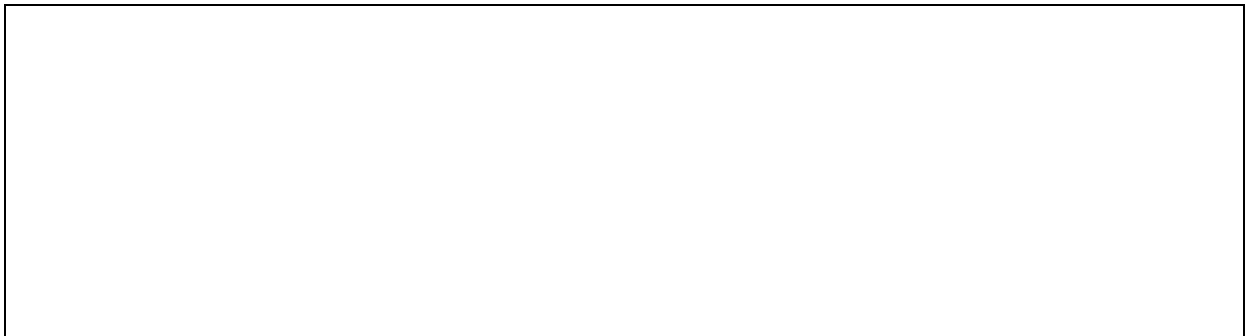
Exercice 2 – Listes chaînées

On considère la déclaration de la structure de données ainsi que les fonctions **X**, **Y** et **Z** suivantes :

```
struct element
{
    int val;
    struct element *nxt;
};
typedef element* llist;
```

```
llist X(llist liste)
{
    if(liste != NULL)
    {
        element* aRenvoyer = liste->nxt;
        free(liste);
        return aRenvoyer;
    }
    else
        return NULL;
}
```

1) Illustrer à l'aide d'un exemple (et un schéma) ce que fait cette fonction **X**. Donnez un non significatif.

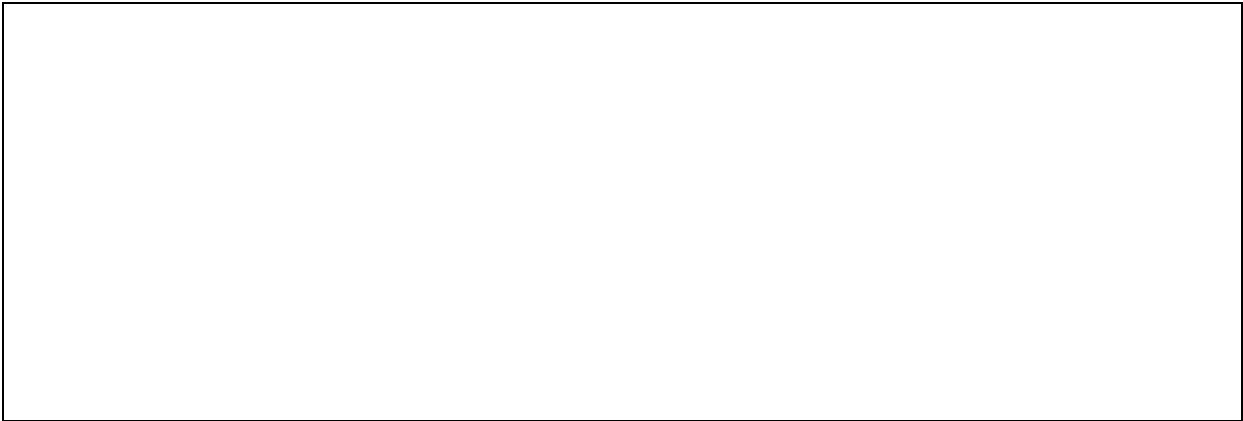


```
llist Y(llist liste)
{
    if(liste == NULL)
        return NULL;

    if(liste->nxt == NULL)
    {
        free(liste);
        return NULL;
    }
    element* tmp = liste;
    element* ptmp = liste;
```

```
while(tmp->nxt != NULL)
{
    ptmp = tmp;
    tmp = tmp->nxt;
}
ptmp->nxt = NULL;
free(tmp);
return liste;
}
```

2) Illustrer à l'aide d'un exemple (et un schéma) ce que fait cette fonction **Y**. Donnez un non significatif.



```
llist Z(llist liste, int valeur)
{
    element *tmp=liste;
    while(tmp != NULL)
    {
        if(tmp->val == valeur)
            return tmp;
        tmp = tmp->nxt;
    }
    return NULL;
}
```

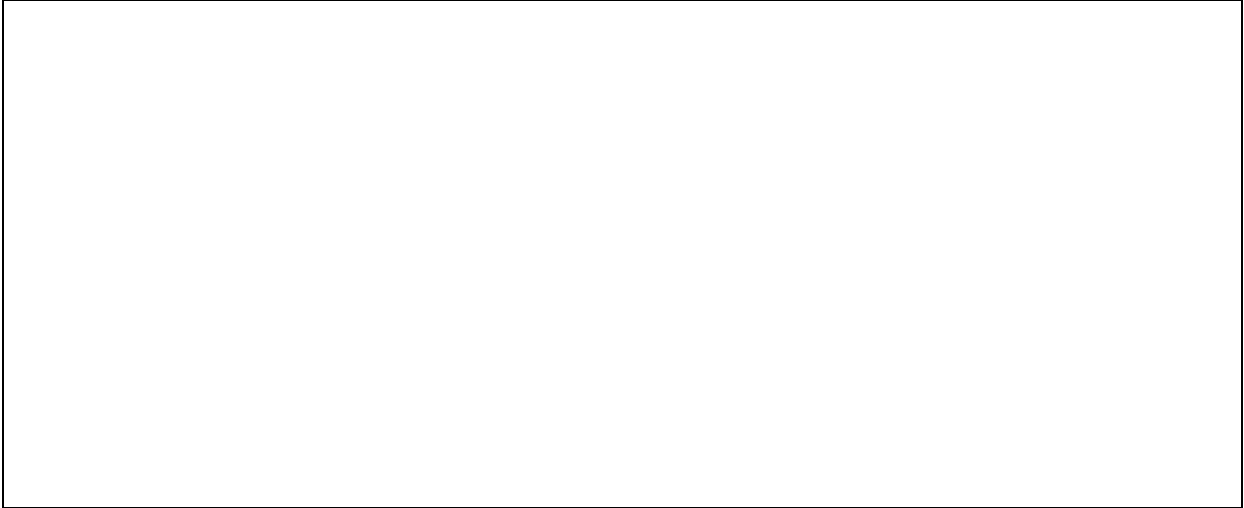
3) Illustrer à l'aide d'un exemple (et un schéma) ce que fait cette fonction **Z**. Donnez alors un non significatif.



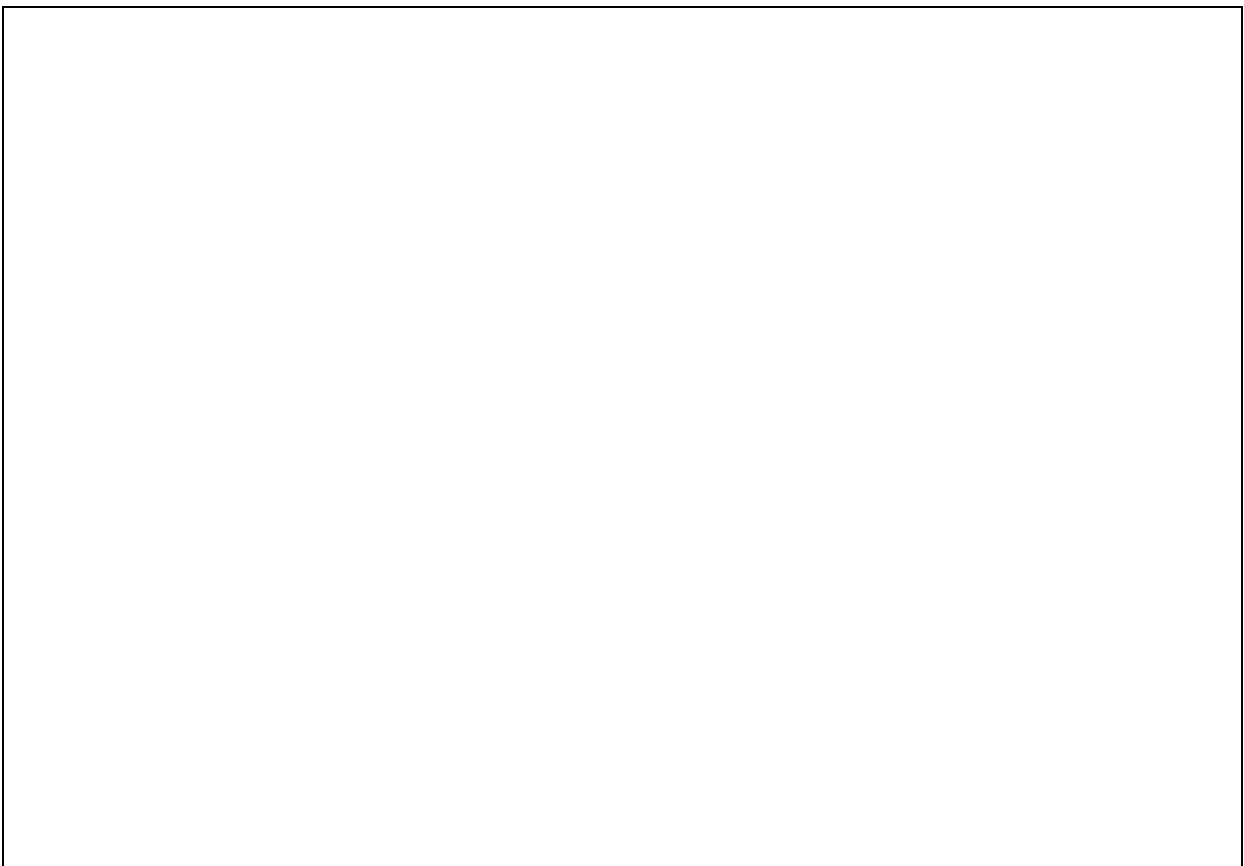
Exercice 3 – Les arbres binaires

On considère la suite de nombres suivante : 96, 99, 46, 64, 58, 50, 29, 30, 27, 84, 97, 26.

1) Partant d'un arbre binaire de recherche vide, on ajoute successivement les valeurs de la suite ci-dessus. Donnez en justifiant votre raisonnement l'arbre binaire résultat (*il ne s'agit pas de donner la version équilibrée de l'arbre*).



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



3) On considère la structure de données suivante pour représenter cet arbre :

```
typedef struct Arbre {
    int Noeud;
    struct Arbre * SAG;
    struct Arbre * SAD;
} Arbre;
```

On considère aussi la fonction suivante Mystere:

```
Mystere (Arbre * Racine) {
    if (Racine!=NULL)
    {
        Mystere (Racine->SAG);
        printf("%d ",Racine->Noeud);
        Mystere (Racine->SAD);
    }
}
```

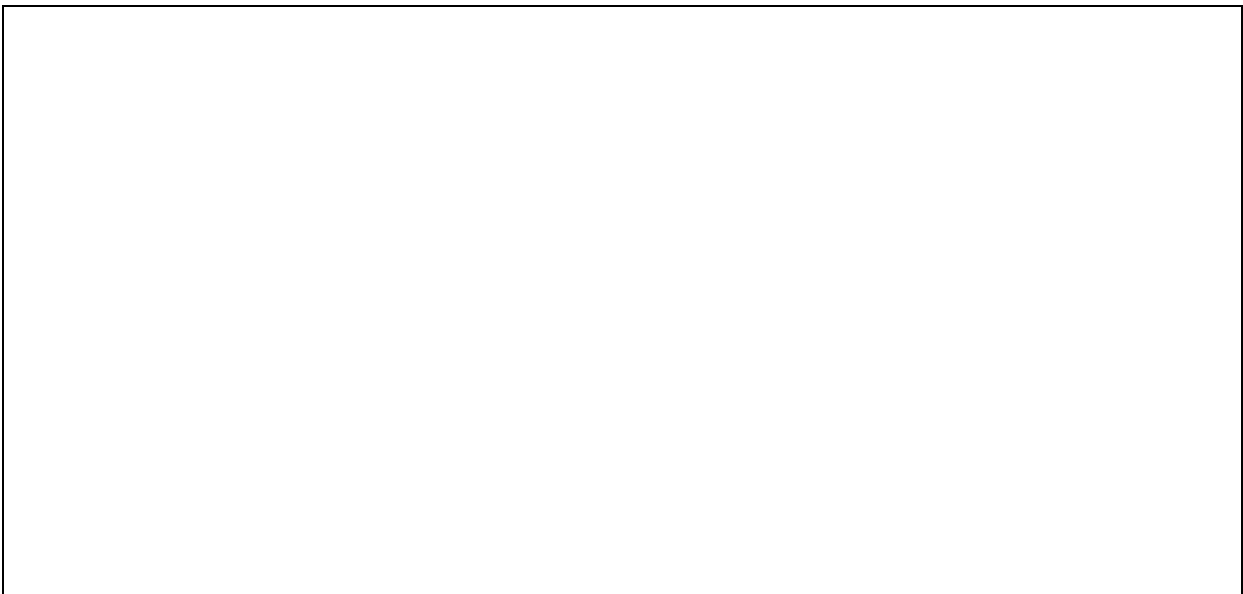
a) Expliquez ce que fait cette fonction Mystere? Donnez alors un non significatif à cette fonction. Donnez le résultat de cette fonction sur l'arbre originale (non équilibré) obtenu dans 1).

b) Donnez le résultat de cette fonction sur l'arbre équilibré que vous avez obtenu dans 2). Comparez les deux listes obtenues dans a) et b). Expliquez l'intérêt d'équilibrage des arbres.

4) Proposez une fonction récursive qui permet de calculer la somme des feuilles d'un arbre.



5) Proposez une fonction récursive qui permet de calculer la hauteur d'un arbre.



Bonne chance.