NOM :

PRENOM :

QROC A31 Structure de données et graphes

Juin 1997

Documents distribués et notes personnelles autorisés.

Il faut impérativement répondre sur la copie.

Faire d'abord au brouillon, aucun double de sujet ne pourra être distribué.

• Récursivité (3 points)

Soit la relation de récurrence de la fonction mystere définie par le code en C suivant :

```
void mystere(int N)
{
if (N >0)
     {
     if (N/8 >0)
               mystere(N/8) ;
               printf("%d", N%8) ;
     }
}
```

a) Donner le résultat de mystere(100). En déduire ce que fait cette fonction.

Réponse :

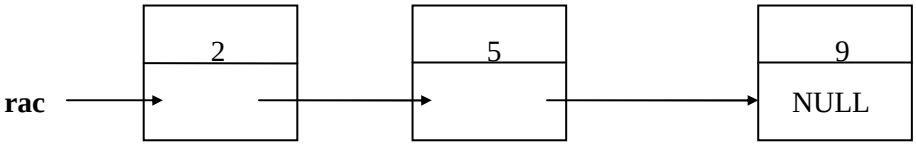
b) **Sans faire de calcul de complexité**, pouvez vous intuitivement donner une idée sur le la complexité de cette fonction.

Réponse :

Liste chaînée (points)

- **Exercice 1 (3 points)**

  On considère la structure chaînée initiale suivante :



  Où rac désigne le pointeur de début de cette liste.

  Donner le résultat des deux portions de codes (répondre dans le tableau réponse).

  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));
rac->suiv=t;
t->suiv=rac->suiv;
t->valeur=11;
```

Recopie des lignes	*Schémas correspondants*

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

Recopie des lignes	*Schémas correspondants*

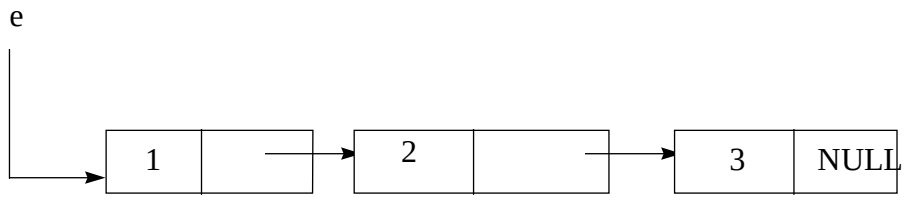
Exercice 2 (4 points)

Dans cet exercice nous souhaitons représenter un ensemble d'entiers par une liste chaînée.

Exemple l'ensemble A définie par :

$$A=\{1, 2, 3\}$$

peut être représentée par la liste chaînée suivante :



1) Proposez une structure de données **ensemble** adéquate qui permet de représenter cette liste chaînée.

Réponse :

2) Ecrire une fonction int NombreElement(ensemble p) qui retourne le nombre d'éléments d'un ensemble.

Exemple : Soit l'ensemble A = {1,2,7} le nombre d'élément de l'ensemble A est égal à 3.

Réponse :

3) Ecrire une fonction **int Intersection(ensemble p, ensemble q)** qui permet de tester si l'intersection de deux ensembles est vide ou non. **Intersection(ensemble p, ensemble q)** retourne 1 si les deux ensembles ont une intersection, 0 sinon.

Exemple1 : les deux ensembles A et B définis respectivement par A = {1,2,7} B={2,1,8} ont une intersection non vide.

Exemple2 : les deux ensembles A et B définis respectivement par A = {1,2,8} B={6,4,9} ont une intersection vide.

Réponse :

4) Sachant que deux ensembles sont égaux si et seulement si ils ont les mêmes éléments écrire une fonction **int Egal(ensemble p,ensemble q)** qui permet de tester si deux ensmbles sont identiques. Egal retourne 1 si les deux ensembles sont identiques, 0 sinon.

Exemple1 : les deux ensembles A et B définis respectivement par A = {1,2,7} B={2,1,7} sont identiques.

Exemple2 : les deux ensembles A et B définis respectivement par A = {1,2,8} B={2,1,7} sont différents.

Conseil : vous pourriez utiliser le résultat de la question numéro 2.

Réponse :

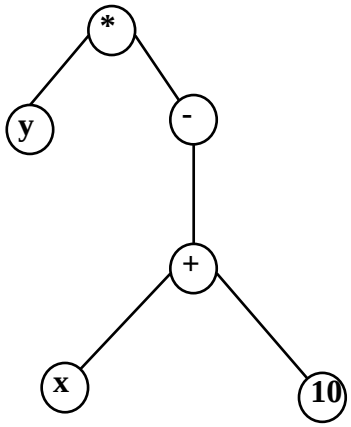
ATTENTION : On ne vous demande pas de créer les listes chaînées, on suppose qu'elles existent.

Arbres (5 points)

Exercice 1

Une expression arithmétique peut être facilement représentée par un arbre. Les arbres d'expression spécifient l'association des opérandes d'une expression et de ses opérateurs d'une manière uniforme, sans qu'il soit nécessaire de se préoccuper du placement des parenthèses.

Exemple : l'expression suivante (y*(-(x+10))) peut être représentée par l'arbre



1) Construisez l'arbre de l'expression suivante : ((x+y)+(5+(x*z)))

Réponse :

2) Une structure de données possible pour représenter les arbres définis précédemment peut être définie de la façon suivante (vu dans le cours):

```
typedef struct node *tree_pointer;
typedef struct node {
        char data;
        tree_pointer fils_gauche, fils_droit;
        };
```

2.1) Dans cette partie nous ne considérons pas comment l'arbre a été crée mais nous supposons qu'il existe.

Soit la fonction mystere définie par :

```
void mystere(tree_pointer ptr){
if (ptr) {
        putchar('(') ;
        mystere(ptr->fils_gauche);
        mystere(ptr->fils_droit);
```

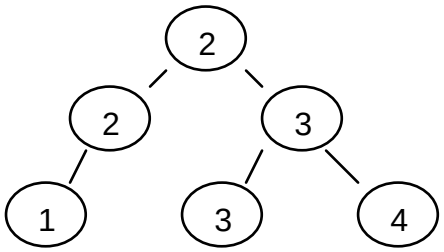
```
        printf("%c",ptr->data);
        putchar(')') ;
        }
}
```

2.2) En appliquant la fonction mystere sur l'arbre de l'expression définie dans la question 1, donner le résultat de l'affichage de la fonction **mystere(racine),** où racine est la racine cet arbre?

Réponse :

Exercice 2 :

Soit l'arbre de **recherche binaire** :



1) Ecrire la fonction qui retourne la valeur minimum des valeurs des noeuds de l'arbre.

Exemple : pour l'arbre précédent la valeur retournée sera 1.

Réponse :

2) Ecrire une fonction qui retourne le produit des valeurs de tous les noeuds de l’arbre.

Exemple : pour l’arbre précédent la valeur retournée sera 72.

Réponse :

NOM :

PRENOM :

- **Graphes (5 points)**
- **Réponse**