Université Mohamed Khider-Biskra
Faculté des Sciences Exactes et Sciences de la Nature et de la Vie
Département d'informatique

$2^{ème}$ année LMD — Algorithmique et structures de données 3
30 Jan 2022 — 08:00-09:00, Amphis K,H

Examen

Exercice 1 Listes linéaires chaînées (1 + 2 + 1 + 2 + 3 + 4)
La représentation des nombres entiers dans les langages de programmation est toujours limitée à quelques octets. On souhaite surmonter ce problème en proposant une représentation des entiers quelque soit leur taille en utilisant les listes linéaires chainées. Un nombre entier est représenté par une liste linéaire chaînée où chaque chiffre du nombre est représentée par un maillon de la liste. Le chiffre du poids faible est rangé à la tête.

Exemple : Le nombre 236799 est représenté par la liste linéaire chaînée suivante :

Nil ⟵ [2] ⟵ [3] ⟵ [6] ⟵ [7] ⟵ [9] ⟵ [9] ⟵ **Tete**

1. Donner la déclaration permettant de représenter en mémoire de telles listes.
2. Soit la procédure f suivante :

```
Procédure f(var L : Poniteur(TMaillon));
var p : Poniteur(TMaillon) ;
Début
    Si (L=Nil) Alors
        Allouer(p) ;
        Aff_val(p,1) ;
        Aff_adr(p, L) ;
        L ← p ;
    Sinon
        Si (Valeur(L)=9) Alors
            Aff_val(L,0) ;
            f(Suivant(L)) ;
        Sinon
            Aff_val(L,Valeur(L)+1) ;
        Fin Si;
    Fin Si;
Fin;
```

   (a) Donner la liste de l'exemple ci-dessus après l'appel de la procédure f(Tete).
   (b) Que fait la procédure f sur un nombre entier représenté par une liste linéaire chaînée ?
   (c) Calculer la complexité de la procédure f en fonction de la longueur n de la liste passée en paramètre.
3. Écrire la procédure Div10 suivante :

   Procédure Div10(var L : pointeur(TMaillon)) ;

   permettant de diviser le nombre représenté dans L par 10.

4. Écrire la fonction **récursive** Multi3 suivante :

   Fonction Multi3(*L* : pointeur(TMaillon)) :booléen ;

   permettant de retourner vrai si le nombre représenté par L est un multiple de 3, sachant qu'un nombre entier est multiple de 3 si la somme des chiffres le représentant est un multiple de 3. Par exemple, le nombre 236799 de l'exemple précédent est un multiple de 3 puisque 2 + 3 + 6 + 7 + 9 + 9 = 36 et 36 est un multiple de 3.

Exercice 2 Arbres (2 + 2 + 2 + 1)
Soit les nombre entiers suivants donnés dans l'ordre :

13, 22, 6, 15, 4, 2, 11, 17

1. Donner l'arbre binaire de recherche obtenu en insérant ces nombres dans l'ordre.
2. Soit la procédure *g* suivante :

```
Procédure g( N : Poniteur(TNoeud));
Début
    Si (N=Nil) Alors
        Ecrire("l'arbre est vide") ;
    Sinon
        Tant que (FG(N)≠ Nil ) faire
            N ←FG(N) ;
        Fin TQ;
        Ecrire(Valeur(N)) ;
    Fin Si;
Fin;
```

   Qu'affiche cette procédure en l'executant sur la racine de l'arbre binaire précédent ?
3. Donner le tas dynamique construit en insérant dans l'ordre ces nombre.
4. En déduire le tas statique.

★★★ Bonne chance ★★★

Corrigé type

Exercice 1 Listes linéaires chaînées (1 + 2 + 1 + 2 + 3 + 4)

1. Déclarations permettant d'utiliser de telles listes.

```
Type  TMaillon = Structure
    Valeur : entier ;
    Suivant : Pointeur(TMaillon) ;
Fin ;
Var  Tete : Pointeur(TMaillon ;
```

1 pt

2. Procédure
   (a) Liste de l'exemple ci-dessus après l'appel de la procédure f(Tete).

$$\textbf{Nil} \longleftarrow [2] \longleftarrow [3] \longleftarrow [6] \longleftarrow [8] \longleftarrow [0] \longleftarrow [0] \longleftarrow \textbf{Tete}$$

2 pts

   (b) La procédure f ajoute 1 au nombre représenté pal la liste. **1 pt**

   (c) Le temps T(n) d'exécution de la fonction :

$$T(n) = \begin{cases} a & si \quad n = 0 \ (L = Nil) \\ b & si \quad Valeur(L) \neq 9 \\ T(n-1) + c & sinon \end{cases}$$

   - La constante a englobe le temps du test ($n \neq Nil$) et le temps des instructions :
     - Allouer(p) ;
     - Aff_val(p,1) ;
     - Aff_adr(p, L) ;
     - $L \leftarrow p$;
   - La constante b le temps du test ($n \neq Nil$), le temps de test (Valeur(L)=9) et le temps de l'instruction Aff_val(L,0) ;
   - La constante c le temps du test ($n \neq Nil$), le temps de test (Valeur(L)=9) et le temps de l'instruction Aff_val(L,Valeur(L)+1) ;

$$\begin{aligned} T(n) &= T(n-1) + b \\ &= [b + T(n-2)] + b \\ &= T(n-2) + 2b \\ &= [b + T(n-3)] + 2b \\ &= \ldots \\ &= T(n-i) + ib \\ &= \ldots \\ &= T(0) + nb = a + nb \\ T(n) &= nb + (aouc) \end{aligned}$$

   Et puisque $a > c$ on prends le pire des cas c'est à dire a,

   Donc la complexité est en $O(nb + a) = O(n)$ car a et b sont des constantes positives. **2 pts**

3. Pocédure Div10 : pour diviser un nombre sur 10, on supprime le premier maillon c'est dire la position des unités

```
Procédure Div10(var L : Poniteur(TMaillon));
var p : Poniteur(TMaillon) ;
Début
    Si (L ≠Nil) Alors
        p ← L ;
        L ← Suivant(L) ;
        Libérer(p) ;
    Fin Si;
Fin;
```

3 pts

4. Écrire la fonction **récursive** Multi3 suivante :

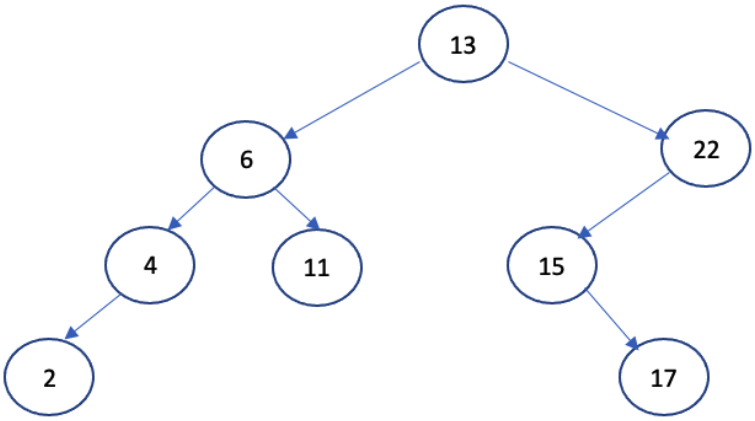
```
Fonction Mult3( L : Poniteur(TMaillon)) : booléen;
    Procédure LibererListe(var L : Poniteur(TMaillon));                    ;
    var q : Poniteur(TMaillon) ;
    Début
        Tant que (L ≠ N) faire
            p ← L ;
            L ← Suivant(L) ;
            Libérer(p) ;
        Fin TQ;
    Fin;
var ListeSomme : Poniteur(TMaillon) ;
Début
    ListeSomme ← Nil ;
    p ← L ;
    Tant que (p ≠ Nil) faire
        Pour i de 1 à Valeur(p) faire
            P(ListeSomme) ;
        Fin Pour;                                                          4
    Fin TQ;
    Si (ListeSomme=Nil) Alors
        Mult3 ← Vrai ;
    Sinon
        Si (Suivant(ListeSomme)=Nil) Alors
            Si (Valeur(ListeSomme) mod 3 = 0) Alors
                Mult3 ← Vrai ;
            Sinon
                Mult3 ←Faux ;
            Fin Si;
        Sinon
            Mult3 ← Mult3(ListeSomme) ;
            LibererListe(ListeSomme)
        Fin Si;
    Fin Si;
Fin;
```

pts

Exercice 2 Arbres (2 + 2 + 2 + 1)
Soit les nombre entiers suivants donnés dans l'ordre :

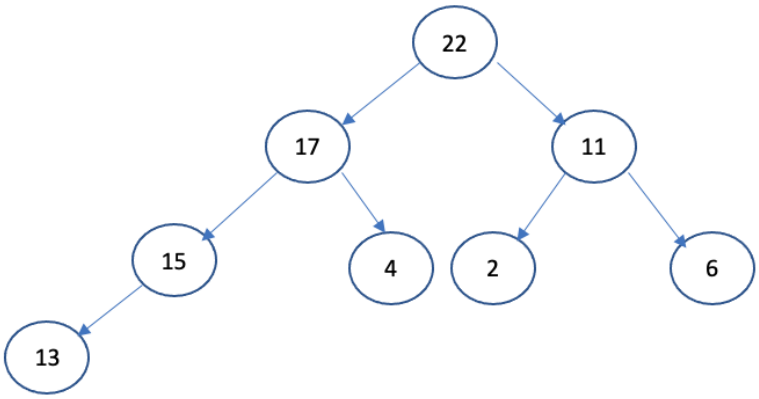
13, 22, 6, 15, 4, 2, 11, 17

1. l'arbre binaire de recherche obtenu en insérant ces nombres dans l'ordre :



2 pts

2. La procédure affiche : 2 **2 pts**
3. Donner le tas dynamique construit en insérant dans l'ordre ces nombre.



2 pts

4. Le tas statique.

22	17	11	15	4	2	6	13

1 pt