USTHB
Faculté d'Electronique & d'Informatique - Département d'Informatique

Algorithmique et Structures de Données 2ème Année ACAD, Section : B

Contrôle S1 - 2018/2019

Exercice 1 :

1- Ecrire une fonction récursive **Puissance (x,n)** qui calcule la valeur de x^n, avec *n* entier naturel, en utilisant l'algorithme de **Lucas** :

$$x^0 = 1 \ , \qquad \forall\, n \in N^*, x^n = \left(x^{\frac{n}{2}}\right)^2 \qquad si\ n\ est\ pair\ \ et$$
$$x^n = x\left(x^{\frac{n-1}{2}}\right)^2 \qquad si\ n\ est\ impair$$

2- Donner la complexité de cet algorithme.

Exemple : $2^{20} = \left(2^{10}\right)^2 = \left(\left(2^5\right)^2\right)^2 = \left(\left(2\left(2^2\right)^2\right)^2\right)^2 = \left(\left(2\left(\left(2^1\right)^2\right)^2\right)^2\right)^2 = \left(\left(2\left(\left(2\left(2^0\right)^2\right)^2\right)^2\right)^2\right)^2$

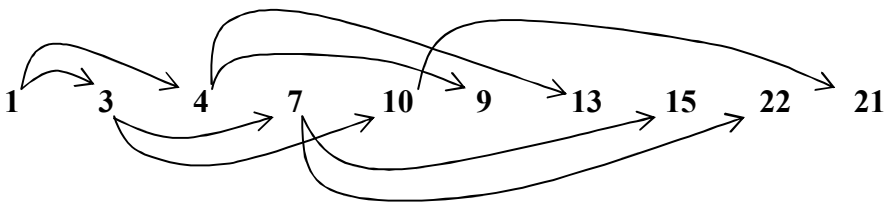
Exercice 2 :

Soit une suite d'entier **Sn** définie comme suit :
Si **A** est élément de la suite Alors les deux éléments : **2A+1** et **3A+1** sont aussi des éléments de la suite.
(Chaque élément **produit** deux autres éléments).
Ecrire un algorithme permettant d'afficher **N** éléments de cette suite à partir d'un élément initial **A dans l'ordre de production**.

Exemple :

Pour **A=1** et **N=10**

On aura la suite :



Bonne Chance

Correction

Exercice 1 :

1- L'algorithme de Lucas est défini d'une manière récursif :
Cas de base n=0, Puissance=1
Sinon, suivant la parité de n, on a deux formules récurrentes de calcul :

```
Fonction Puissance(x,n :entier) :entier ;
Var P :entier ;
Debut
        Si n=0 Alors P←1
                Sinon  Si n MOD 2 =0
                        Alors  P←Puissance(x,n DIV 2) ;
                               P←P*P
                        Sinon  P←Puissance(x,(n-1) DIV 2) ;
                               P←x*P*P
                        Fsi
        Fsi ;
        Puissance←P ;
Fin ;
```

Implémentation en C :

```
double Pow(double x, int n)
{ int P;
  if (n==0) return 1;
  else if (n%2==0) {P=Pow(x,n/2); return P*P;}
       else {P=Pow(x,(n-1)/2); return x*P*P;}
}
```

Implémentation en Pascal :

```
Function Pow(x,n:integer):integer;
var P:integer;
begin
    if n=0 then P:=1
    else if n mod 2 =0
         then begin p:=pow(x,n div 2); p:=p*p; end
         else begin p:=pow(x,(n-1) div 2); p:=x*p*p; end;
    pow:=P;
end;
```

2- Complexité :

D'près l'algorithme, on remarque qu'à chaque appel de la fonction **n** est devisé par deux.
A chaque appel, on fait 1 ou 2 multiplications (suivant la parité de **n**).
Dans le meilleur cas (**n** est une puissance de 2, $n=2^k$), on fait **k+1** multiplications.
Dans le pire des cas (la décomposition de **n** en binaire ne contient que des **1**), on fait **2k+1** multiplications.
D'où la complexité est $O(Log(n))$.

Exemple :
n=8, ($n=2^3$) et x=3

Appel principal Puissance(3,8)	81x81		
1er appel Puissance(3,4)	9x9		
2eme appel Puissance(3,2)	3x3	4 multiplications (3+1)	meilleur cas
3eme appel Puissance(3,1)	3x1		
4eme appel Puissance(3,0)	1		

n=12, ($n=2^2+2^3$) et x=3

Appel principal Puissance(3,12)	729x729		
1er appel Puissance(3,6)	27x27		
2eme appel Puissance(3,3)	3x3x3	5 multiplications	cas intermédiaire
3eme appel Puissance(3,1)	3x1		
4eme appel Puissance(3,0)	1		

n=15, ($n=2^0+2^1+2^2+2^3$) et x=3

Appel principal Puissance(3,15)	3x2187x2187		
1er appel Puissance(3,7)	3x27x27		
2eme appel Puissance(3,3)	3x3x3	7 multiplications (2*3+1)	pire des cas
3eme appel Puissance(3,1)	3x1		
4eme appel Puissance(3,0)	1		

Exercice 2 :

Le nombre d'éléments n'étant pas connu (N), il faut choisir donc une structure de donnée dynamique. Afin d'afficher les éléments dans l'ordre de production, on doit les mettre dans une file.

```
Algorithme Suite ;
Var   F :File ;
      I,J,N,A :entier ;
Debut
      Ecrire('Donner une valeur de départ de la suite :') ;
      Lire(A) ;
      Ecrire('Donner le nombre d''éléments à afficher :') ;
      Lire(N) ;
      InitFile(A) ;
      Enfiler(F,A) ; //enfiler le premier élément
      I←1 ; J←0 ;
      Tantque I≤N
      Faire  Défiler(F,A) ;
             Ecrire(A) ; J←J+1 ; //J : nombre d'éléments affichés
             Enfiler(F,2*A+1) ;
             Enfiler(F,3*A+1) ; //production de deux éléments
             I ← I+2 ; //I : le nombre d'éléments produits
      Fait ;
      //afficher le reste des éléments dans la file, non encore affichés
      Tantque J<N
      Faire  Défiler(F,A) ;
             Ecrire(A) ; J←J+1 ;
      Fait ;
      //Vider la file, car elle peut contenir 1 ou 2 éléments en plus qu'on ne doit pas afficher
      Tantque Non FileVide(F)  Faire   Défiler(F,A) ;  Fait ;
Fin.
```

Exemple

Pour A=1 et N=10

I	J	Affichage	Etat de la File					
1	0		1					
3	1	**1**	3	4				
5	2	**1 3**	4	7	10			
7	3	**1 3 4**	7	10	9	13		
9	4	**1 3 4 7**	10	9	13	15	22	
11	5	**1 3 4 7 10**	9	13	15	22	21	31
Fin 1ere boucle								
	6	**1 3 4 7 10 9**	13	15	22	21	31	
	7	**1 3 4 7 10 9 13**	15	22	21	31		
	8	**1 3 4 7 10 9 13 15**	22	21	31			
	9	**1 3 4 7 10 9 13 15 22**	21	31				
	10	**1 3 4 7 10 9 13 15 22 21**	31					
	Fin 2 eme boucle							
	Fin 3 eme boucle							

Solution 2 :

Algorithme Suite ;
Var F :File ;
I,N,A :entier ;
Debut
Ecrire('Donner une valeur de départ de la suite :') ;
Lire(A) ;
Ecrire('Donner le nombre d''éléments à afficher :') ;
Lire(N) ;
InitFile(A) ;
Enfiler(F,A) ; *//enfiler le premier élément*
I←1 ;
Tantque I≤N
Faire Défiler(F,A) ;
Ecrire(A) ;
Enfiler(F,2*A+1) ;
Enfiler(F,3*A+1) ; *//production de deux éléments*
I ← I+1 ;
Fait ;
//Vider la file, car elle peut contenir des éléments en plus qu'on ne doit pas afficher
Tantque Non FileVide(F) **Faire** Défiler(F,A) ; **Fait** ;
Fin.

Examen Final - 2018/2019

Exercice 1 : (4 Pts)

1- Soit N un entier positif donné en base 10, écrire une fonction récursive qui convertit N en binaire.
2- Donner la complexité de cet algorithme.

Exercice 2 : (5 Pts)

Soit l'ensemble de valeurs entières {10, 20, 4, 1, 15, 28, 12, 17, 16, 18, 17, 10, 23, 18, 31, 25}
1- Dessiner l'arbre binaire de recherche correspondant (valeurs répétitives dans le SAD).
2- Donner les listes des éléments obtenues par les parcours préfixé, postfixé et en largeur de cet arbre.
3- Dessiner l'arbre après la suppression de l'élément 20.
4- Ecrire une fonction **NBfeuille** qui renvoie le nombre de feuilles dans un arbre binaire A.

Exercice 3 : (11 Pts)

Lors de la saisie d'une suite de caractères en utilisant un éditeur, on dispose de deux touches de contrôle :
'**#**' : Permet d'effacer le dernier caractère tapé, s'il y en a un.
'**%**' : Permet de tout effacer jusqu'au début.
Soit **F** une file de caractères contenant une suite de caractères brute (y compris les caractères de contrôle).
On veut développer une procédure **Nettoyer** permettant de nettoyer la file **F** des caractères de contrôles.

Exemple :

F brute : #jE fat#it part%Je suie#s un e#étudiet##ant en 2ACABD##D B

F nette : Je suis un étudiant en 2ACAD B

1- Donner la déclaration d'une File de caractères et celui d'une Pile de caractères.
2- Ecrire la procédure **Nettoyer** qui réalise la tache décrite précédemment.
3- Afin de gérer des files brutes contenant plusieurs lignes, on rajoute un autre caractère de contrôle :
'**$**' : Indique la fin de la ligne.
Soit **Ftext** une file contenant plusieurs lignes brutes.
   a- Ecrire une procédure **Extraire** permettant d'extraire la première file brute de **Ftext**.
   b- Pour sauvegarder les différentes lignes nettes de **Ftext**, on utilise une structure (**Snet**) de liste chainée, où chaque élément contient un pointeur vers la file de la ligne nette et un pointeur vers la ligne suivante.
      - Donner la déclaration de la structure **Snet.**
      - Ecrire une procédure **Construire** permettant de remplir une structure de type **Snet** à partir de **Ftext.**

NB : Les primitives de base manipulant les piles et les files sont supposées prédéfinies.

Bonne Chance

Correction

Exercice 1 :

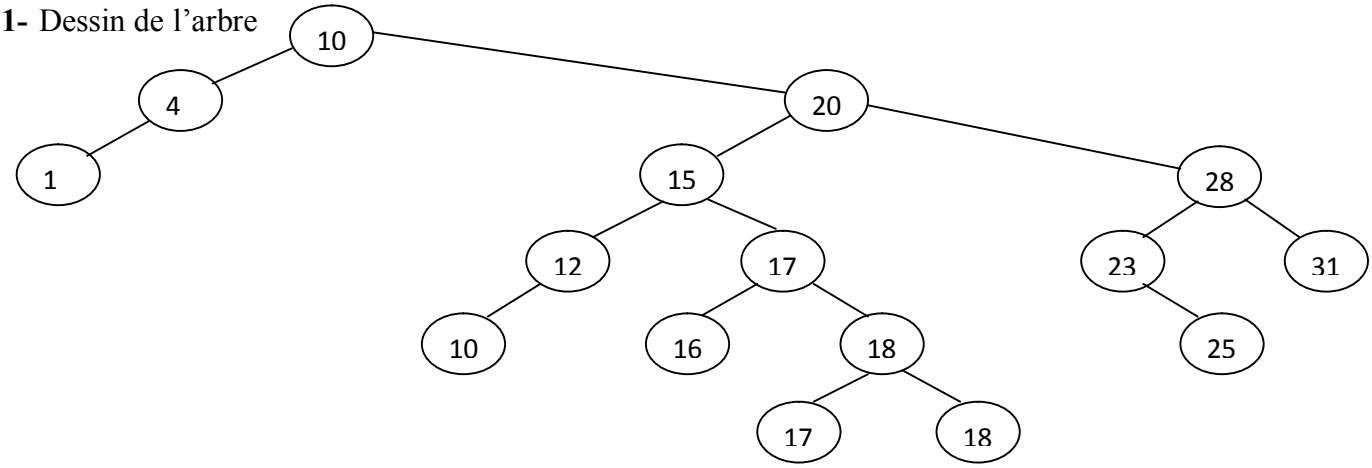
Fonction DecToBin(N :entier) :entier ;
Debut
Si N=0 **Alors** DecToBin ← 0
Sinon DecToBin ← N **MOD** 2 +10*DecToBin(N **DIV** 2);
Fsi;
Fin ;

3- Complexité :

D'près l'algorithme, on remarque qu'à chaque appel de la fonction N est devisé par deux.
A chaque appel, on fait une addition, une multiplication et une division (3 opérations de base)
En posant $N=2^K$, on fait **K** appels. D'où la complexité est $O(Log(N))$.

Exercice 2 : 5 Pts

1- Dessin de l'arbre

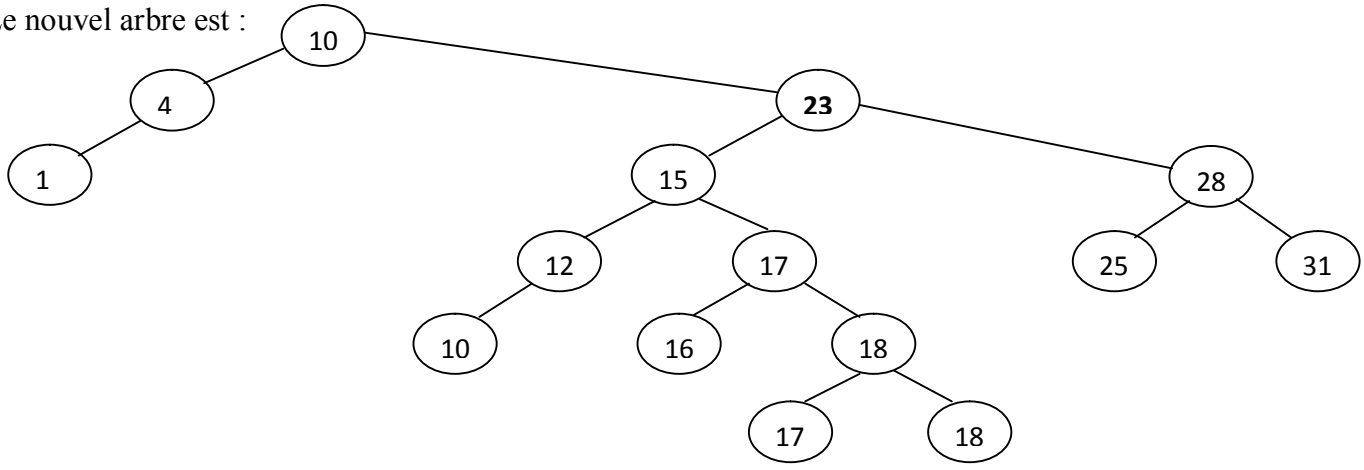


2- Eléments de la liste :
- Parcours Préfixé : 10-4-1-20-15-12-10-17-16-18-17-18-28-23-25-31
- Parcours Postfixé : 1-4-10-12-16-17-18-18-17-15-25-23-31-28-20-10
- Parcours en largeur : 10-4-20-1-15-28-12-17-23-31-10-16-18-25-17-18

3-Suppression de 20 :

On cherche le Min du SAD (Min=23), on remplace 20 par 23, puis on supprime 23 (fils gauche de 28) du SAD de 20. Le fils gauche de 28 devient le fils droit de 23 qui est 25.

Le nouvel arbre est :



4- Fonction nombre de feuilles

```
Fonction NBFeuille(A : Arbre) :entier ;
Debut
        Si ArbreVide(A)
        Alors  NBFeuille←0
        Sinon Si Feuille(A)   Alors NBFeuill←1
                              Sinon NBFeuille← NBFeuille(FilsGauche(A))+ NBFeuille(FilsDroit(A))
                Fsi
        Fsi;
Fin ;
```

Exercice 3 : 11 Pts.

1- Déclaration de File est Pile

Pile=^Epile ;

Epile=**Enregistrement** Info:caractere ; Suiv : Pile ; **FinEnreg** ;

File= **Enregistrement** Tete,Queue :Efile ; **FinEnreg** ;

Efile=^CelluleF ;

CelluleF= **Enregistrement** Info :caractere ; Suiv :Efile ; **FinEnreg** ;

2-

```
Procedure Nettoyer(E/S F :File) ;
Var     P,R :Pile ;
        X :caractere ;
Debut
        //Nettoyer F  dans la pile P
        InitPile(P) ;
        Tantque Non FileVide(F)
        Faire  Defiler(F,X) ;
                Cas X Vaut
                '# ' : Si Non PileVide(P) Alors Depiler(P,X)  Fsi;
                '%' : Tantque Non PileVide(P) Faire Depiler(P,X) ; Fait ;
                Sinon Empiler(P,X) ;
                FinCas ;
        Fait ;
        InitPile(R) ;
        Tantque Non PileVide(P) Faire Depiler(P,X) ; Empiler(R,X) ; Fait ;   //Inverser la pile P
        Tantque Non PileVide(R) Faire Depiler(R,X) ; Enfiler(F,X) ; Fait ;  //Recharger la file F
Fin ;
```

3-a-

```
Procedure Extraire(E/S  FT,F :File) ;
Var  X :caractere ;
Debut
        InitFile(F) ;
        Si Non FileVide(FT)
        Alors  Defiler(FT,X) ;
                Tantque Non FileVide(FT) et X≠'$'
```

```
            Faire  Enfiler(F,X) ; Defiler(FT,X) ; Fait ;
            Si FileVide(FT) et X≠'$' Alors Enfiler(F,X) Fsi; //traitement dernier caractère
        Fsi ;
Fin ;
```

3-b-

- Structure de Snet

```
Snet=^Liste ;
Liste= Enregistrement   Pfile :File ; Suiv :Snet ; FinEnreg ;
```

- Procedure de construction

```
Procedure Construire(E/S FT :File ; E/S Lnet :Snet) ;
Var     F :File ;
        LN,PN :Snet ;
Debut
        Lnet←Nil ;
        //créer Lnet en FIFO, on doit respecter l'ordre entre les lignes
        Si Non FileVide(FT)
        Alors  Extraire(FT,F) ;
               Nettoyer(F) ;
               Allouer(Lnet) ;  Lnet^.Pfile←F ; LN←Lnet ;   //création de la tête
               //créer les autres éléments
               Tantque Non FileVide(FT)
               Faire
                     Extraire(FT,F) ;
                     Nettoyer(F) ;
                     Allouer(PN) ;
                     PN^.Pfile←F ;
                     LN^.Suiv←PN ;
                     LN←PN ;
               Fait ;
               LN^.Suiv←Nil
        Fsi ;
Fin ;
```