Université de Batna 2
Département d'informatique
Licence 2 : informatique académique

Contrôle Final

Durée 1h30

Exo1 (4 pts) : la complexité

- La fonction ci-dessous permet de calculer X^n « **puissance (x,n)** ».

```
double puissance (double x, int n){

    double resultat=1 ;

    if (n==0){
            resultat=1 ;
    }else{
            int i ;

            for (i=0 ;i<n ;i++){
                    resultat = resultat*x ;
            }
    }

return resultat ;

}
```

Q1) Calculer la **complexité** de cette fonction.

- L'algorithme ci-dessous permet calculer la valeur d'un polynôme P(x) de degré N pour une valeur X donnée.

$P(x) = 3X^2 - 7X + 1$ ➔ **N=2**
➔ **Les coefficients de ce polynôme sont: P [0]=1, P[1] = -7, P[2]= 3**
➔ Pour X= 5 ➔ evaluer (2, 5, P) retourne 41.

```
double evaluer (int N, double X, double P []){
        int resultat = 0;
        int i;
        for (i=0 ;i <=N ;i++){

                resultat= resultat + P[i]* puissance (X,i) ;
        }

        return resultat ;
}
```

Q2) Quelle est la complexité de la fonction : **evaluer** ?

Exo2 (12 pts) : liste doublement chainée

Un étudiant est caractérisé par : son numéro d'inscription, son nom (20 caractères), son prénom (20 caractères), et sa moyenne générale.

Q1) Proposer une structure de donnée chainée « **étudiants »** qui permet de gérer les étudiants d'une manière **bidirectionnelle**.

Q2) Si un pointeur occupe un espace mémoire de 4 octets. Donner le résultat de la fonction :
sizeof (étudiant) ;

Q3) Présenter avec un schéma de pointeurs **les 3 cas de suppression** d'un élément d'une **liste doublement chainée** qui contient 4 éléments

Q4) Ecrire une fonction qui permet **d'insérer** un étudiant dans une **liste doublement chaînée triée**.

Remarque 1 : La liste **doit restée triée** par ordre croissant du *N*.
Remarque 2 : on pourra pas y avoir 2 étudiants qui ont le même numéro.

Exo3 (4 pts): Piles + Arbre binaire de recherche

On veut trier un tableau d'entier et le placer dans une pile **en utilisant un arbre binaire de recherche**.

Q1) Décrire les étapes à suivre afin de réaliser cet objectif.

Q2) Quel type de parcours devons nous utiliser ? Et pourquoi ?

Q2) Ecrire **les fonctions** nécessaires qui permettent de trier les éléments d'un tableau et les placer dans une pile (toujours en utilisant un arbre binaire de recherche).

Remarque : dans l'exercice 3, vous pouvez utiliser directement les fonctions vues dans le cours :

- **InsererABR (A, X) :** insérer l'élément X dans l'arbre binaire de recherche A

- **Pile_vide (P1)** =1 si la pile vide, 0 si la pile contient au moins un élément.
- **X= Dépiler (P1).**
- **Empiler (P1, X)**.

- **File_vide (F1)** =1 si la file vide, 0 si la File contient au moins un élément.
- **X= Défiler (F1).**
- **Enfiler (F1, X).**

- **Etc...**

Bon courage seulement aux étudiants qui travaillent dur

Corrigé type :

Exo1 :

Q1) la fonction **puissance** dépend de N

Solution 1 : calcule rapide

- **Calculer C(N) dans le pire des cas ➔ N != 0** — 1

  Resultat = 1 ➔ 1 instruction
  Int i ; ➔ 1 instruction
  Boucle : ➔ N fois (multiplication, affectation) = N * 2
  Return resultat ➔ 1 instruction

  C(N)= 1+1+ 2*N + 1 = **2N +3**

2 pts :
- 1,5 pts : méthode
- 0,5 pts : *O* (N)
- - 0,5 pts : calcule incorrecte

- **Calculer C(N+1) :** — 0,25

  C(N+1) = 2*(N+1) + 3 = **2N+5**

- **Calculer C(N+1) – C(N)** — 0,25

  C(N+1)-C(N) = 2 << N

  ➔ **complexité d'ordre N ➔ *O*(N)** — 0,5

Solution 2 : calcule détaillé

- **Calculer C(N) dans le pire des cas ➔ N != 0**

  Resultat = 1 ➔ 1 instruction
  Test n==0 ➔ 1 instruction
  Int i ; ➔ 1 instruction
  Boucle : ➔ i=0 ➔ 1 instruction

  ➔N fois (i<n, i++, multiplication , affectation) = N * 4
  Return resultat ➔ 1 instruction

  C(N)= 1+1+ 1+1+ 4*N + 1 = **4N +5**

- **Calculer C(N+1) :**

  C(N+1) = 4*(N+1) + 5 = **4N+9**

- **Calculer C(N+1) – C(N) :**

  C(N+1) – C(N) = 4 << N ➔ complexité d'ordre N ➔ *O*(N)

2 pts :
- 0,75 pts : méthode.
- 1 pts C(puissance) dépend de i
- 0,25 pts : $O(N^2)$
- - 0,5 pts : calcule incorrecte

Q2) la fonction **évaluer** dépend de N.

Solution 1 : calcule rapide

- **Calculer C(N) dans le pire des cas** 0,25

Resultat = 0 ➔ 1 instruction
Int i ; ➔ 1 instruction
Boucle : ➔ (N +1) fois

➔ Affectation ➔ 1 instruction
➔ Addition ➔ 1 instruction
➔ Multiplication ➔ 1 instruction
➔ Puissance (X, i) ➔ C (puissance) = **2*i +3** 1

return resultat ; ➔ 1 instruction

C(N) = 3 + C(boucle)

- C (boucle) ???

Supposons N= 5

- I=0
  - 3 + (2*0 + 3) = 6+0 =6
- I =1
  - 3 + (2*1+3) = 6+2 =8
- I =2
  - 3 + (2*2+3) = 6+4 =10
- I =3
  - 3 + (2*3+3) = 6+6 =12
- I =4
  - 3 + (2*4+3) = 6+8 =14
- I =5
  - 3 + (2*5+3) = 6+10 =16

C(boucle)= 6*N + (0+2+4+6+8+10) = 6*N + 2* (1+2+3+4+5)
= 6*N + 2* (1+2+3+....+N)
= 6*N + 2* [(1+N) (N) /2]
= 6*N + (N+1)N
= 6N + N^2+N
= N^2 + 7N

$C(N) = 3 + C(\text{boucle}) = N^2 + 7N + 3$

- **Calculer C (N+1) :** 0,25

$$C(N+1) = (N+1)^2 + 7(N+1) + 3$$
$$= N^2 + 2N + 1 + 7N + 7 + 3$$
$$= N^2 + 9N + 11$$

- **Calculer C (N+1) – C(N)** 0,25

$C(N+1) - C(N) = 2N + 8 << N^2 \rightarrow O(N^2)$ 0,25

Solution 2 : calcule plus rapide

La fonction puissance dépend de la boucle **« les autres instruction seront éliminé lors de la soustraction : C (N+1) – C (N) »**

Donc il suffit de calculer C (boucle).

Calcule de C_N (boucle) :

$C_N(\text{boucle}) = N^2 + 7N$

Calcule de C_{N+1} (boucle) :

$$C_{N+1}(\text{boucle}) = (N+1)^2 + 7(N+1)$$
$$= N^2 + 2N + 1 + 7N + 7$$
$$= N^2 + 9N + 8$$

Calcule de C_{N+1}(boucle) – C_N (boucle) :

$C_{N+1}(\text{boucle}) - C_N(\text{boucle}) = 2N + 8 << N^2 \rightarrow O(N^2)$

Exo3 :

Q1)

- Etape 1 : il faut insérer l'élément du tableau dans un arbre binaire de recherche.
- Etape 2 : On fait un traitement infixé de l'arbre en insérant les nœuds dans une pile

Ou

- Explication avec des schémas

1

Q2) un parcours infixé. Le parcours infixé d'un arbre binaire de recherche produit la suite ordonnée des éléments

0,5

Q3) On accepte même le passage par valeur, car le module est algorithmique :

- **<u>Etape1 : remplir l'arbre binaire de recherche :</u>**

1

```
Main (){
...
...
 Pile P ;
 Arbre A ;
 For (int i = 0 ; i< taille ; i++){
        insererABR (A, T[i]) ;
 }

TraitementInfixé (A, P) ;


}
```

➢ **Etape2 : remplir l'arbre binaire de recherche :**

```
TraitementInfixé (Arbre A, Pile P) {

        If (A != NULL){
                TraitementInfixé ( (*A).FG ) ;

                Empiler (P, (*A).info) ;

                TraitementInfixé ( (*A).FD) ;

        }

}
```

1,5

EXO2 :

Q1)

```
Struct etudiant {

Int N ;
Char[20]  nom ;
Char [20]  prenom ;
Double moyenne ;
Etudiant* suivant ;
Etudiant* precedent ;

}
```

2

Q2)

Bloc1 (N)	4 octets occupé
	4 octets perdus
Bloc 2 (nom)	8 octets (nom) : nom [0]➔ nom [7]
Bloc 3 (nom)	8 octets (nom) : nom [8]➔ nom [15]
Bloc 4 (nom+ prenom)	4 octets (nom) : nom [16]➔ nom [19] + 4 octets (prenom) : prenom [0]➔ prenom [3]
Bloc 5 (prenom)	8 octets (prenom) : prenom [4]➔ prenom [11]
Bloc 6 (prenom)	8 octets (prenom) : prenom [12]➔ prenom [19]
Bloc 7 (moyenne)	8 octets
Bloc8 (suivant + précèdent) « même type »	4 octets (pointeur suivant)
	4 octets (pointeur precedent)

Explication :
Le bloc le plus grand est 8 octets (double) ➔ on divise la mémoire en bloc de 8 octets.

Calculer le nombre de blocs

- Un entier occupe 4 octets. **Cependant**, on doit réserver **1 bloc** = **8 octets pour N** (4 octets perdu).
- On a 2 tableaux, chacun de 20 caractères (on peut fusionner les 2 tableaux parce qu'ils ont le même type)
  - Donc total = 40 caractères ➔ nbr de blocs = 40/8 = **5 blocs**
- Moyenne de type double ➔ 8 octets = **1 blocs**
- Dans l'exercice : un pointeur occupe un espace de 4 octets
  - On a 2 pointeurs (suivant et précèdent) ➔ ils ont le même type ➔ on peut les fusionner dans un même bloc ➔ 4+4 = 8Octets = **1bloc**

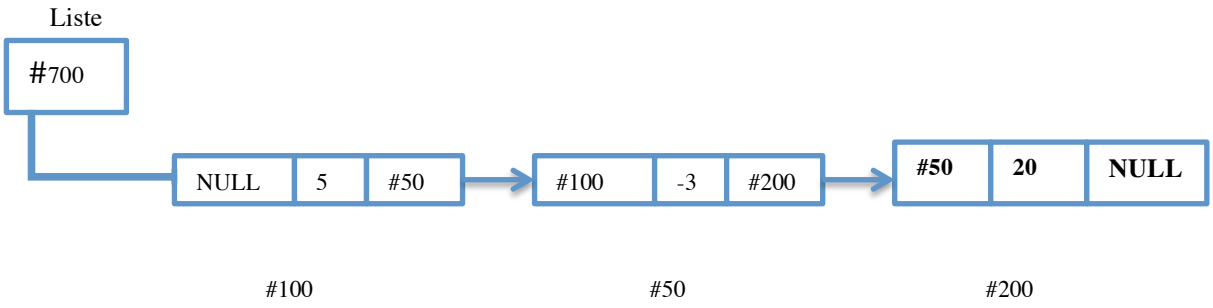
1

Total = 1+5+1+1 = 8 blocs = 8*8 octets = 64 octets
Sizeof(etudiant) = 64 Octet

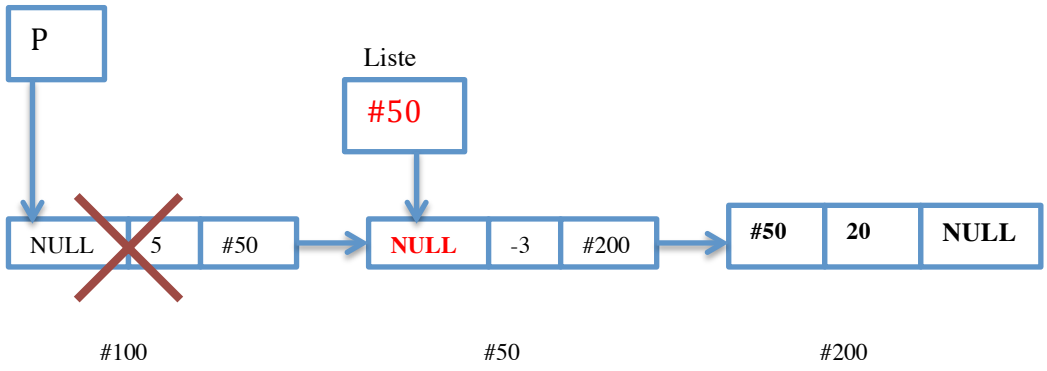
Q3) les 3 schémas

3

Liste initiale (cas de 3 éléments) :

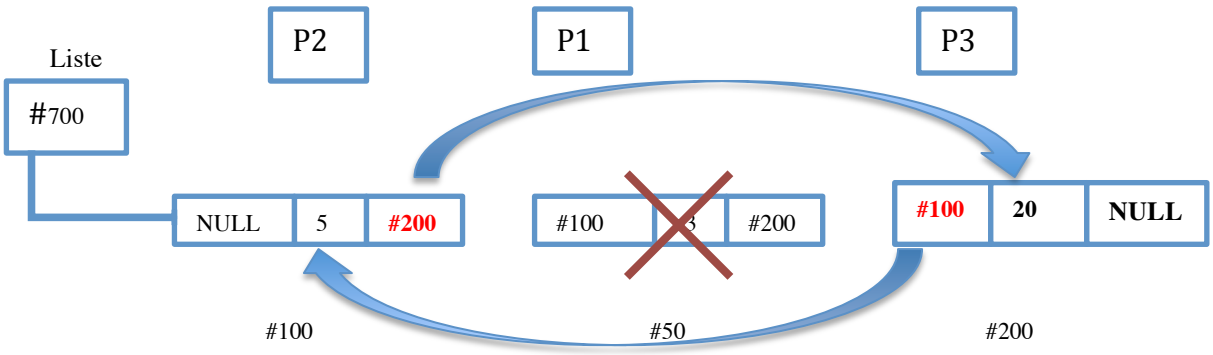


Supprimer le 1er élément




```
P= liste ;
Liste = (*liste).suivant ;
(*liste).precedent = NULL ;
free (p) ;
```

Suppression au milieu

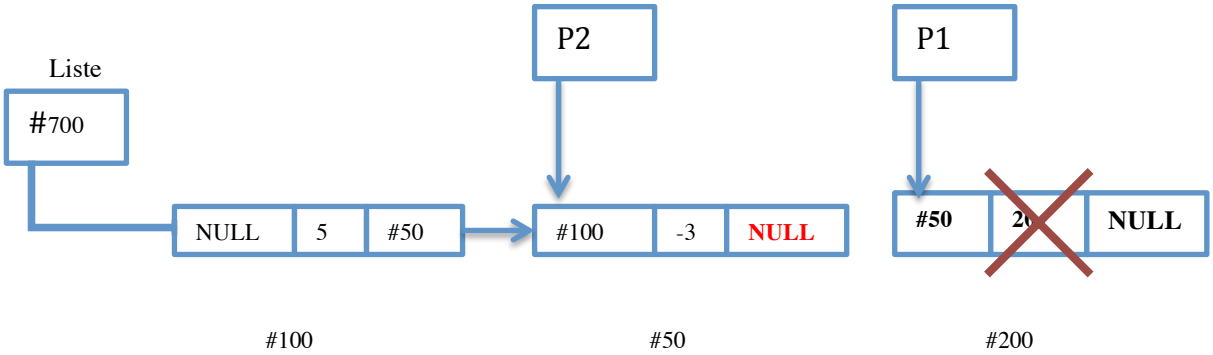



```
P1= Liste ;
While ( (*P1).info != x ){
        P1= (*P1).suivant ;
}

P2= (*P1).precedent ;
P3= (*P1).suivant ;

(*P2).suivant = P3 ;
(*P3).precedent = P2 ;
free (P1) ;
```

Supprimer le dernier élément




```
P1= Liste ;
While ( (*P1).suivant != NULL )  {
        P1= (*P1).suivant ;
}

P2= (*P1).precedent ;
(*P2).suivant = NULL ;
free (P1) ;
```

Q4) l'algorithme : 6 pts

- entête ..... (0,5 pts)
- allocation d'espace mémoire ..... (1pts)
- initialisation des positions ..... (1pts)
- boucle pour trouver la bonne position..... (1pts)
- teste d'égalité..... (0,5 pts)
- les 4 liaisons ..... (2 pts)