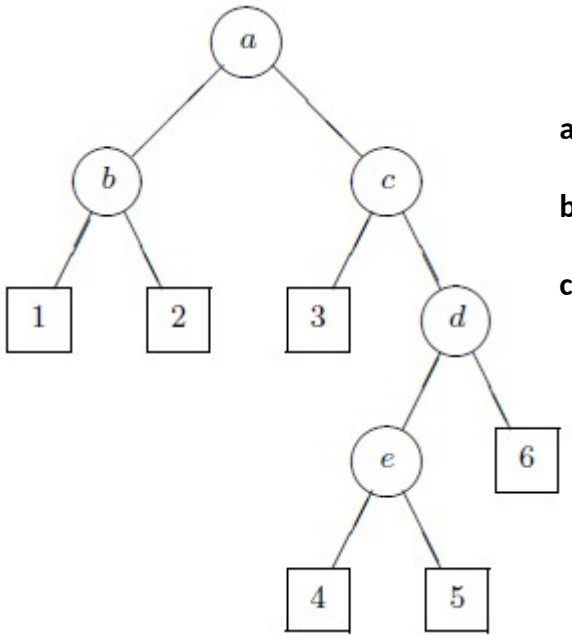
Université Sétif 1 Mardi 12 Janvier 2016
Faculté des Sciences
Département d'informatique **Module : Algorithmique et structure de données**
Filière : 2ème A. Licence Académique

**EXAMEN (**durée :1h30m)

**Questions de cours (3+4 = 7Pts)**

**Q1 (1+1+1 = 3Pts)**



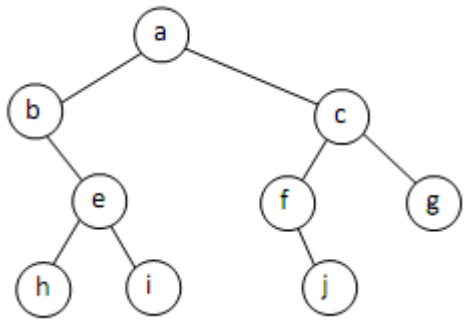

Donner le **résultat** (Rendu) des **parcours** :

a) **a- préfixé (pré-ordre)**

b) **b- infixé (symétrique)**

c) **c- postfixé (post-ordre)**

**Q2 (2+2 = 4Pts)**




a- Représenter l'arbre à gauche dans un tableau

b- Dite si c'est un **TAS** oui ou non et pourquoi ?

**Exercice (2+2+3+2+3+1 = 13Pts)**

Afin de me permettre de gérer la **LISTE** (Liste simplement chainée) de mes étudiants(es) inscrits(es) au module ASD je vous propose la structure suivante :
Chaque élément (maillon) de la liste est composé des champs suivants :
**Num_Insc** (9 caractères), ( N° d'inscription de l'étudiant(e))
**Nom** (20 car),
**Prénom** (20 car),
**Sexe** (1 car), ( M :masculin ; F : féminin)
**Note_TD** (Réel), (Note de TD comprise entre 0 et 20 de **coefficient 1.00**)
**Note_TP** (Réel), (Note de TP comprise entre 0 et 20 de **coefficient 1.25**)
**Note_Examen** (Réel) (Note d'EXAMEN comprise entre 0 et 20 de **coefficient 2.00**)
**Moyenne** (Réel) (la moyenne de l'étudiant calculée en fonction des 3 notes et leurs coefficients)
**Répétitif** (entier) ( 0 :Nouveau ; 1 :Répétitif pour la 1ère fois ; 2 : Répétitif plus de 2 fois !)

**Je vous demande de m'écrire les algorithmes (Fonctions/Procédures)** qui me permettent de :

a- Initialisation de la liste (déclaration de la structure)
b- insertion d'un étudiant (les étudiants sont insérer l'un à la suite de l'autre)
c- suppression d'un étudiant connaissant son **Num_Insc**
d- rechercher un étudiant connaissant son **Nom et Prénom**
e- Afficher la liste des étudiants comme suit
   Num_Insc – Nom – Prénom – Note_Examen - Note_TD – Note_TP – Moyenne – **Résultat**

   Nombre Total d'étudiants ; Moyenne module ; meilleure moyenne ; mauvaise moyenne
   **NB : Résultat** = Admis(es) si Moyenne >= 10 ou Ajourné(ée) sinon

f- Afficher la liste des étudiants qui peuvent passés l'examen de rattrapage (7.00 <= Moyenne < 10.00) ainsi que leur nombre

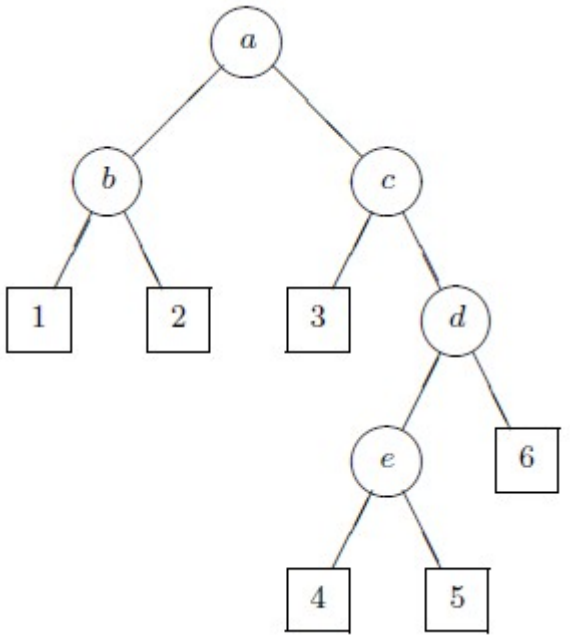
**NB :** Afin de développer des algorithmes sur les LLCs, on construit une machine abstraite avec les opérations suivantes : Allouer(P) et Libérer(P) et Suivant(P) définies comme suit :
– **Allouer(P)** : allocation d'un espace de taille spécifiée par le type de P. L'adresse de cet espace est rendue dans la variable de type Pointeur P.
– **Libérer(P)** : libération de l'espace pointé par P.
– **Suivant(P)** : consultation du champ Suivant du maillon pointé par P.

**Bon courage**

**Solution type pour l'examen ASD**

**Questions de cours**

**Q1 (1+1+1 = 3Pts)**




le résultat (Rendu) des parcours :

a- Préfixe←RGD←.←1Pts←

ab1 2 c3 de4 5 6.

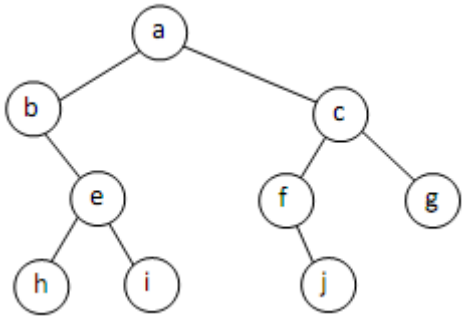
b- Infixe ←GRD←: ←1Pts←

1 b2 a3 c4 e5 d6

c- Postfixe ←GDR← : ←1Pts←

1 2 b3 4 5 e5 dca

**Q2 (2+2 = 4 Pts)**




a- Représentation de l'arbre dans un tableau ←2Pts←

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | B | C | | E | F | G | | | h | i | | J | |

**b- Ce n'est pas un TAS** puisque Un arbre binaire parfait a une représentation compacte **sans trous** dans le tableau ←2Pts←

**Exercice** **(2+2+3+2+3+1 = 13Pts)**

a- **Fonction/Procedure** initialisation() //Déclaration de la structure ←2Pts→

```
Type Etudiant = Structure
        Num_Insc : chaine_car(9) ;
        Nom :chaine_car(20) ;
        Prénom :chaine_car(20) ;
        Sexe : caractere(1) ;
        Note_TD : Réel ;
        Note_Tp : Réel ;
        Note_Examen : Réel ;
        Moyenne :Réel ;
        Répétitif : Entier ;
        Suivant : Pointeur(Etudiant) ;
Fin ;

Variables  Tete : Pointeur(Etudiant)
```

b- Fonction insertion() //Insertion d'un nouveau étudiant(au début et en fin) ←2Pts→

```
Fonction /Procédure Insertion(tète)
Var P, Q : Pointeur(Etudiant) ;
Début
Allouer(Q) ;
Lire(Num_Insc(Q)) ;
Lire(Nom(Q)) ;
Lire(Prénom(Q)) ;
Lire(Sexe(Q)) ;
Lire(Note_TD(Q)) ;
Lire(Note_TP(Q)) ;
Lire(Note_Examen(Q)) ;
Suivant(Q) ← Nil ;

Si (Tete = Nil) Alors  /* ajouter en tete */
   Tete ← Q
Sinon
    P ← Tete ;
    Tant que suivant(P) != Nil Faire
            P ←  suivant(p)
    Fin Tant que
    Suivant(P) ← Q
Fin Si;
Fin
```

c- Fonction suppression() //Suppression d'un étudiant(début-milieu-fin) ←3Pts→

```
Fonction/Procedure SupprimerElement ( Tete , Val )
/* Supprime l'élément dont la valeur est passée en paramètre */
Variables
P : pointeur(Etudiant)    /* pointeur sur l'élément à supprimer */
Prec : pointeur(Etudiant)    /* pointeur sur l'élément précédant l'élément à supprimer */
Trouvé : Booléen     /* indique si l'élément à supprimer a été trouvé */
DEBUT
SI Tete <> Nil ALORS /* la liste n'est pas vide on peut donc y chercher une valeur à supprimer */
    SI Num_Insc(Tete) = Val ALORS   /* l'élément à supprimer est le premier */
            P ← Tete
            Tete ←Suivant(Tete)
            Liberer(P)
```

```
        SINON   /* l'élément à supprimer n'est pas le premier */
                Trouve ← ⊏Faux
                Prec ← ⊏Tete   /* pointeur précédent */
                P ← ⊏Suivant(Tete)   /* pointeur courant */
                TANTQUE P <> Nil  ET Non Trouve Faire
                        SI Num_Insc(P) = Val ALORS   /* L'élément recherché est l'élément courant */
                                Trouve ← ⊏Vrai
                        SINON   /* L'élément courant n'est pas l'élément cherché */
                                Prec ← ⊏P   /* on garde la position du précédent */
                                P ← ⊏Suivant(P)   /* on passe à l'élément suivant dans la liste */
                        FINSI
                FIN TANT QUE
                SI Trouve ALORS
                        Suivant(Prec) ←Suivant(P)   /* on "saute" l'élément à supprimer "/
                        Liberer(P)
                SINON
                        Ecrire ("La valeur ", Val, " n'est pas dans la liste")
                FINSI
        FINSI
SINON
        Ecrire("La liste est vide")
FINSI
FIN
```

d- Rechercher une valeur ←2Pts←

```
Fonction/Procedure RechercherValeurListe (Tete , Val)
/* Rechercher si une valeur donnée en paramètre est présente dans la liste passée en paramètre */
Variables
P : pointeur(Etudiant)    /* pointeur de parcours de la liste */
Trouve : booléen /* indicateur de succès de la recherche */
DEBUT
        SI Tete <> Nil ALORS  /* la liste n'est pas vide on peut donc y chercher une valeur */
                P ←Tete
                Trouve ← ⊏Faux
                TANTQUE P <> Nil ET Non Trouve faire
                        SI Num_Insc(P) = Val ALORS   /* L'élément recherché est l'élément courant */
                                Trouve ← ⊏Vrai
                        SINON   /* L'élément courant n'est pas l'élément recherché */
                                P←Suivant(P)    /* on passe à l'élément suivant dans la liste */
                        FINSI
                FIN TANT QUE
                SI Trouve ALORS
                        Ecrire (" La valeur ", Val, " est dans la liste")
                SINON
                        Ecrire (" La valeur ", Val, " n'est pas dans la liste")
                FINSI
        SINON
                Ecrire("La liste est vide")
        FINSI
FIN
```

e- Fonction affichage_liste_etudiants()  //Affichage de la liste inclus le calcul du résultat(Admis/Ajourné)

```
Fonction / Procedure AfficherListe (Tete)                ←3Pts←
/* Afficher les éléments d'une liste chaînée passée en paramètre */
Variables       Nb_Et :Entier ;
                Moy_Mod, Meilleure_Moy, Mauvaise_Moy : Réel
DEBUT
SI Tete = Nil ALORS
                Ecrire('Liste vide')
```

```
SINON
        P ← ⌴Tete   /* P pointe sur le premier élément de la liste*/
        Nb_Et ←0
        Moy_Mod ← 0 ; Meilleure_Moy← Moyenne(P) ; Mauvaise_Moy← Moyenne(P)
        /* On parcourt la liste tant que l'adresse de l'élément suivant n'est pas Nil */
        TANT QUE P <> NIL FAIRE /* si la liste est vide Tete est à Nil */
                Ecrire(Num_Insc(P))   /* afficher la valeur contenue à l'adresse pointée par P */
                Ecrire(Nom(P))
                Ecrire(Prénom(P))
                Ecrire(Note_Examen(P))
                Ecrire(Note_TD(P))
                Ecrire(Note_TP(P))
                Ecrire(Moyenne(P))
                SI Moyenne(P) >= 10 ALORS
                        Ecrire('Admis')
                SINON
                        Ecrire('Ajournée')
                FIN SI
                SI Moyenne(P) > Meilleure_Moy  ALORS
                        Meilleure_Moy ← ⌴Moyenne(P)
                FIN SI

                SI Moyenne(P) < Mauvaise_Moy  ALORS
                        Mauvaise_Moy ← ⌴Moyenne(P)
                FIN SI

                Nb_Et ← Nb_Et +1
                Moy_Mod ← Moy_Mod  + Moyenne(P)

                P ← ⌴Suivant(P) /* On passe à l'élément suivant */
        FIN TANT QUE
        Ecrire('Nombre d étudiants  = ', Nb_Et)
        Ecrire('Moyenne Module = ', Moy_Mod / Nb_Et)
        Ecrire('Meilleure Moyenne  =  ', Meilleure_Moy)
        Ecrire('Mauvaise Moyenne  =  ', Mauvaise_Moy)
FIN SI
FIN
```

f- Fonction affichage_liste_rattrapage()  //Affichage de la liste de rattrapage( 7 <= Moyenne < 10

```
Fonction/Procedure AfficherListeRattrapage (Tete)
/* Afficher les éléments d'une liste chaînée passée en paramètre */
Var         Nb_Et :Entier ;
P :  pointeur(Etudiant) ;
DEBUT
SI Tete = Nil ALORS
    Ecrire('Liste vide')
SINON
    P ← ⌴Tete   /* P pointe sur le premier élément de la liste*/
    Nb_Et ←0
    /* On parcourt la liste tant que l'adresse de l'élément suivant n'est pas Nil */
    TANT QUE P <> NIL FAIRE /* si la liste est vide Tete est à Nil */
            SI Moyenne(P) >= 7 ET Moyenne(P) < 10  ALORS
                    Ecrire(Num_Insc(P))   /* afficher la valeur contenue à l'adresse pointée par P */
                    Ecrire(Nom(P))
                    Ecrire(Prénom(P))
                    Ecrire(Note_Examen(P))
                    Ecrire(Note_TD(P))
                    Ecrire(Note_TP(P))
                    Ecrire(Moyenne(P))
                    Nb_Et ← Nb_Et +1
            FIN SI
            P ← ⌴Suivant(P) /* On passe à l'élément suivant */
    FIN TANT QUE
```

←1Pts←

Ecrire(‘Nombre d étudiants pour rattrapage  = ’, Nb_Et)
**FIN SI**
**FIN**