INF 1001 L

LIF 1 : Algorithmique et Programmation Impérative, Initiation Durée : 2h

Contrôle Continu Terminal : Mardi 15 juin 2010

Les documents, calculatrice, téléphone portable sont interdits.

Vous veillerez à respecter les notations et les règles d'écriture des algorithmes vues en cours et en TD. Le barème est donné à titre indicatif.

Partie A : QCM (4 pts)

La feuille jointe en annexe devra être complétée et rendue avec votre copie.
N'oubliez pas d'y inscrire le **numéro d'anonymat** fourni sur la copie.

Partie B : Algorithmique (7 pts)

1. Un nombre parfait est un nombre naturel n non nul qui est égal à la somme de ses diviseurs stricts (n exclus).
   Exemple : 6 = 1 + 2 + 3
   a. Écrire en langage algorithmique une fonction booléenne qui retourne vrai si un entier n passé en paramètre est un nombre parfait, faux sinon.
   b. Écrire en langage algorithmique le programme principal permettant d'afficher la liste des nombres parfaits compris entre 1 et 10000. On utilisera le résultat renvoyé par la fonction précédente.
2. Soit M un tableau 2 dimensions de taille 3 * 5 contenant des entiers. Écrire en langage algorithmique un sous-programme permettant de remplir un tableau 1D avec la somme des colonnes de M.

1	5	6	4	3
8	9	0	6	4
3	2	7	1	8

12	16	13	11	15

Partie C : Langage C (9 pts)

1. Écrire en langage C/C++ un sous programme permettant de compter et renvoyer au programme appelant le nombre de majuscules, de minuscules et de voyelles dans une chaîne de caractères passée en paramètre.
2. **Match Nul**. On souhaite recueillir un certain nombre d'informations concernant les joueurs des équipes actuellement en lice pour la coupe du monde de football.
   Pour chaque joueur, on mémorisera son nom, son prénom, son poids, et une variable permettant de savoir s'il est titulaire ou non. Une équipe est connue par sa nationalité (France, Argentine, ...) ainsi qu'un tableau des 23 joueurs (11 titulaires + 12 remplaçants)
   a. Déclarez en langage C/C++ deux constantes TAILLE_EQUIPE et TITULAIRE ayant pour valeurs respectives 23 et 11.
   b. Donnez en langage C/C++ la définition des structures JOUEUR et EQUIPE précédemment décrites.
   c. Écrire en langage C/C++ une fonction permettant de remplir la structure JOUEUR.
   d. Écrire en langage C/C++ une fonction permettant de remplir une EQUIPE.
   e. Écrire en langage C/C++ une fonction permettant d'afficher les noms des 11 titulaires d'une équipe.
   f. Écrire en langage C/C++ une fonction permettant de calculer et de renvoyer la moyenne des poids des joueurs d'une équipe

N° Anonymat :
(figurant sur la copie double)

LIF1 : Contrôle Continu Terminal

Mardi 15 juin 2010

Partie A : Questions de cours

1. Dans la portion d'algorithme suivant, A, B, C et D sont des entiers. Si A vaut 1 avant l'exécution de la première ligne, combien vaut D après l'exécution de ces 3 lignes ?

```
B ← 2 * A - 4
C ← -3 * A  - 2* B
D ← 15 * C + 21 * B / 3
```

☐ D vaut -1 ☐ D vaut 0
☐ D vaut 1 ☐ D vaut autre chose

2. Dans la portion de code suivante, n est un entier. Que vaut n après l'exécution de ces lignes ?

```
n← 0
tant que n  > 10 faire
    n ← n + 2
fin tant que
```

☐ n vaut 0 ☐ n vaut 22
☐ n vaut 20

3. Dans la portion de code suivante, m est un entier ayant pour valeur 7. Que vaut m lorsque l'exécution de ces lignes est terminée ?

```
faire
    si m modulo 2 = 1 alors
        m ← 3 * m + 1
    sinon
        m ← m / 2
    fin si
Tant que m ≠ 1
```

☐ m vaut 0 ☐ on a une boucle infinie
☐ m vaut 1 ☐ m vaut 2

4. En programmation en langage C, quel signe utilise-t-on pour l'affectation ?
☐ = ☐ = =
☐ := ☐ ! =

5. En programmation en langage C, quel signe utilise-t-on pour le test d'égalité ?
☐ = ☐ = =
☐ := ☐ !=

6. L'instruction «switch» sert à éviter des instructions :
☐ while... imbriquées. ☐ if... else... imbriquées.
☐ do... while imbriquées. ☐ for... imbriquées.

7. Dans une structure if...
☐ Les parenthèses encadrant la condition logique sont obligatoires.
☐ Le mot clé « else » est obligatoire.
☐ La condition, énoncée juste après if, est suivie d'un point virgule.

8. Où sont déclarées les fonctions?
☐ Après la fonction principale « main », si les prototypes sont définis avant.
☐ Au cours du programme principal.
☐ Avant le programme principal et la règle "une fonction doit être déclarée avant usage" doit être respectée.

9. Une fonction renvoie toujours une valeur. Comment cela se programme-t-il, à l'intérieur de la fonction ?
   - ☐ En précédant la valeur à retourner par le mot-clé return.
   - ☐ En affectant à l'identificateur de fonction la valeur à retourner.
   - ☐ Le langage C renvoie toujours la dernière valeur calculée dans la fonction.

10. Lesquelles de ces définitions de fonctions sont correctes si elles sont sensées calculer « x » élevé à la puissance « N » entière. ?
   - ☐ float puissance (float X , int N)
   - ☐ puissance (float X ; int N)
   - ☐ puissance (float x , int N) float
   - ☐ puissance() ;
   - ☐ puissance : float ;
   - ☐ void puissance (float X , int N)

11. On considère l'appel suivant : echange(A, B). Pour qu'il y ait inversion des valeurs des variables A et B dans le programme principal, on utilise :
   - ☐ Un passage par valeur.
   - ☐ Un passage par adresse.

12. Pour accéder à la troisième case du vecteur Carte, on utilise l'instruction :
   - ☐ Carte [3] ;
   - ☐ Carte [2] ;
   - ☐ Carte {2} ;
   - ☐ Carte {3} ;
   - ☐ Carte (2) ;

13. Quelle(s) déclaration(s) correspond(ent) à une matrice de N lignes et M colonnes ?
   - ☐ float Identificateur[N][M] ;
   - ☐ float Identificateur [M-1][N-1] ;
   - ☐ float Identificateur1 [M-1] Identificateur2 [N-1] ;
   - ☐ float Identificateur1 [N-1] Identificateur2 [M-1] ;

14. Pour accéder à la case située à la 2éme ligne et la 3éme colonne de la matrice T, quelle est la bonne syntaxe?
   - ☐ T [2,1] ;
   - ☐ T [1,2] ;
   - ☐ T [2] [1] ;
   - ☐ T [1] [2] ;
   - ☐ T (2, 1) ;

15. Soient les déclarations suivantes :

```
struct LIVRE
{
    int nb_pages ;
    char langue[20], auteur[20], titre[20] ;
    float prix;
} ;
struct LIVRE L ;
```

Comment accéder à un champ d'un enregistrement ?
   - ☐ auteur.L
   - ☐ L.auteur
   - ☐ L : auteur
   - ☐ auteur.livre
   - ☐ livre.auteur

16. On considère la déclaration suivante :

```
#define MAX 50
struct TIMBRE {
int prix ;
char origine[20] ;
int annee ;
char image[20] ;
char couleur[20];
} ;
struct TIMBRE COLLECTION [MAX];// Une collection est un tableau de timbres
```

Comment accède-t-on à l'année du 3ème timbre de la collection?
   - ☐ COLLECTION [2, 2]
   - ☐ COLLECTION [2]. annee
   - ☐ COLLECTION [2, annee]
   - ☐ TIMBRE[3].annee
   - ☐ COLLECTION. annee [2]
   - ☐ COLLECTION. annee

INf 1002 L

Licence Sciences et Technologie
Mention Mathématiques, Informatique : U.E. : LIF2
Contrôle continu: Bases Physiques de l'Informatique
(Tout document interdit)

Lyon 1

P. Moretti – 45 mm

répondre sur les feuilles

Date : 22/01/2010

Numéro d'Anonymat : ***Groupe***

I Considérons (Fig.1a,b,c) un câble coaxial (infini). Le conducteur interne a un rayon $\mathbf{R_1}$ et porte une charge **+Q** par unité de longueur (h = 1), le conducteur externe a pour rayons interne $\mathbf{R_2}$ et externe $\mathbf{R_3}$ ($\mathbf{R_2} < \mathbf{R_3}$).

1°) Ou sont situées les charges ? Placer les charges sur les Figs. 1b. et 1c.

2°) Sachant que le champ **E(r),** en un point M entre les deux cylindres, est donné en coordonnées cylindriques (cordonnées, r,θ, z, axe du cylindre selon oz, base unitaire u, v, k,) par:

$$\vec{E} = \frac{Q}{2\pi\varepsilon_0 r}\vec{u}$$

Dessiner, sur les trois figures (Fig 1,a,b,c) le champ en M et les lignes de champ sur les Figures **1b** et **1c**.

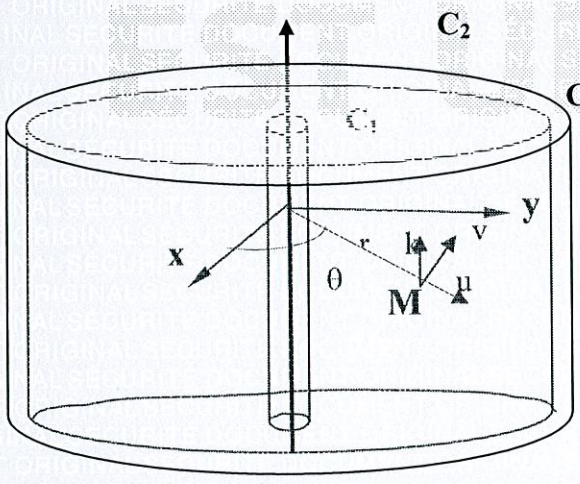


Fig.1 a

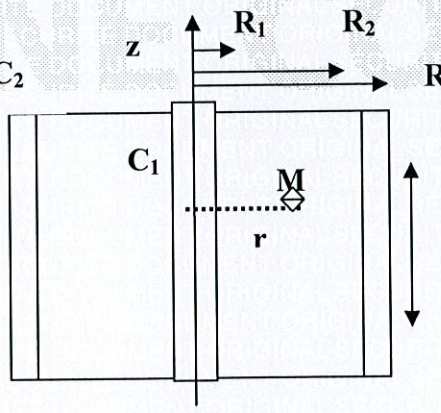


Fig.1 b

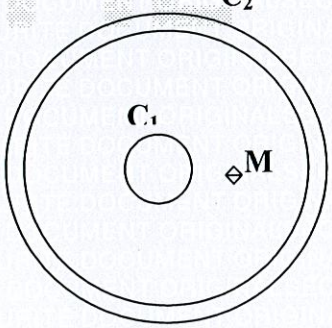


Fig.1 c

3°) Calculer le potentiel **V(M)** au point M (entre les cylindres). Dessiner les équipotentielles, sur les deux figures Fig 1a et 1c, en le justifiant.

4°) Calculer le potentiel du cylindre C_1.

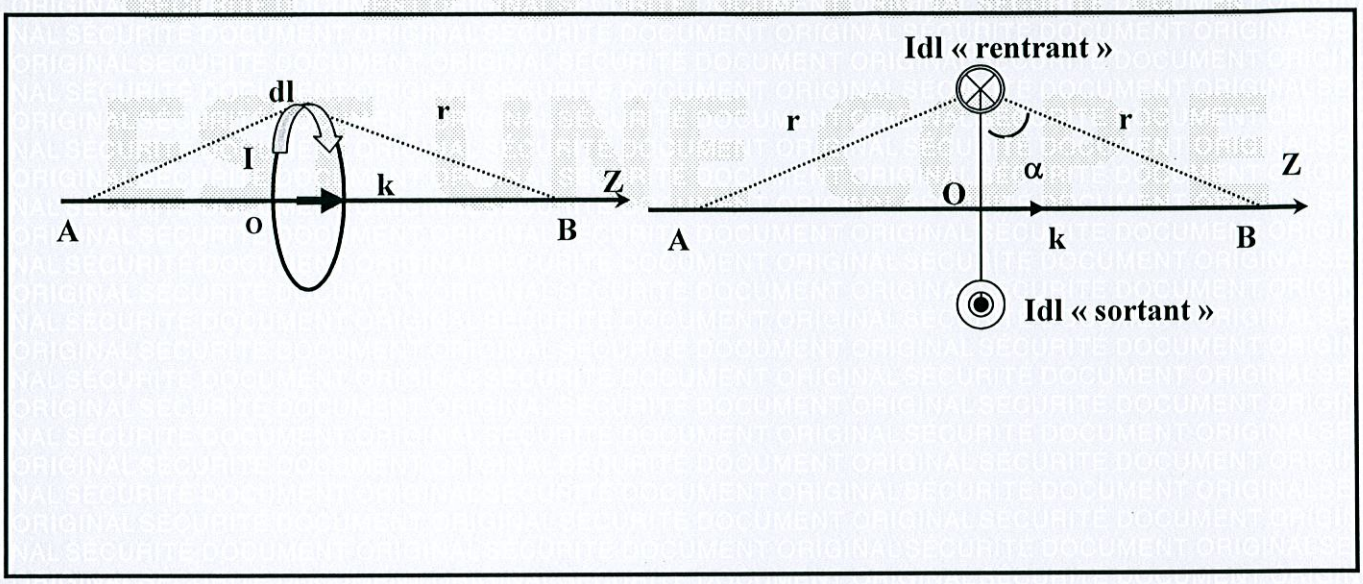
5°) Calculer la capacité **C** par unité de longueur (h =1) de ce câble.

6°) Quelle est la valeur de cette capacité si la permittivité du milieu entre les cylindres est ε_r ?

(II) - Milieux semi-conducteurs.
Expliquer, en illustrant avec des dessins, les deux cas possibles de polarisation d'une jonction PN

(III) - Soit une spire circulaire placée dans l'air, de rayon **a,** de centre O, et parcourue par un courant **I** (voir le sens sur la figure).
1° Rappeler la loi donnant le champ élémentaire **dH** créé par un élément **Idl** à une distance r

2° Dessiner les vecteurs champs magnétiques élémentaires aux points **A** et **B** sur l'axe **Oz** et expliquer qualitativement (**pas de calcul**) comment est le champ résultant en **A** et **B** et **0** (préciser sons sens par rapport au vecteur uitaire **k** de l'axe).



3°)Ecrire l'expression du champ élémentaire au point O, module et vecteur (base **k**), et dessiner le sur les figures.

4° Calculer l'expression du champ **H(o),** au centre **O** de la spire (**uniquement en O**).

INF 1002 L

Licence Sciences et Technologie
Mention Mathématiques, Informatique : U.E. : LIF2
Contrôle continu: Bases Physiques de l'Informatique
(Tout document interdit)

Lyon 1

M. Beuve– 45 mm | ***répondre sur les feuilles*** | ***Date : 22/01/2010***

Numéro d'Anonymat : ***Groupe***

Question de cours (2/10)

1) Qu'est ce que la modularité ? Donner, en électronique, une condition pour que cette modularité soit possible.

Problème A

Soit un clavier à 4 touches. On souhaite réaliser un circuit électronique qui renvoie 1 si et seulement si « une seule » touche est activée et 0 sinon

1) Ecrire la fonction logique de cette sortie (3/10)

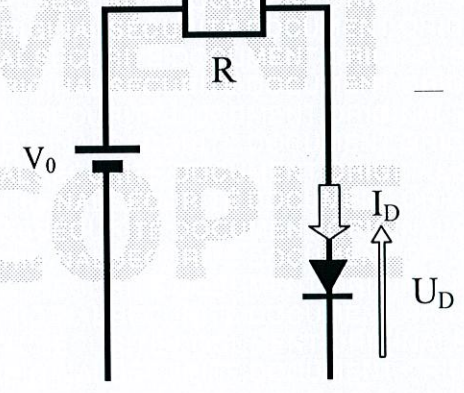
2) Dessiner le circuit associé (2 /10)

Problème B

Le schéma suivant représente une diode alimentée par une source de tension idéale de valeur V_0 (=1V) via une résistance R (=5Ω). La caractéristique de la diode est donnée par la figure 2.

- Déterminer graphiquement le courant I_D et la tension U_D du circuit (2/10)

Figure 1



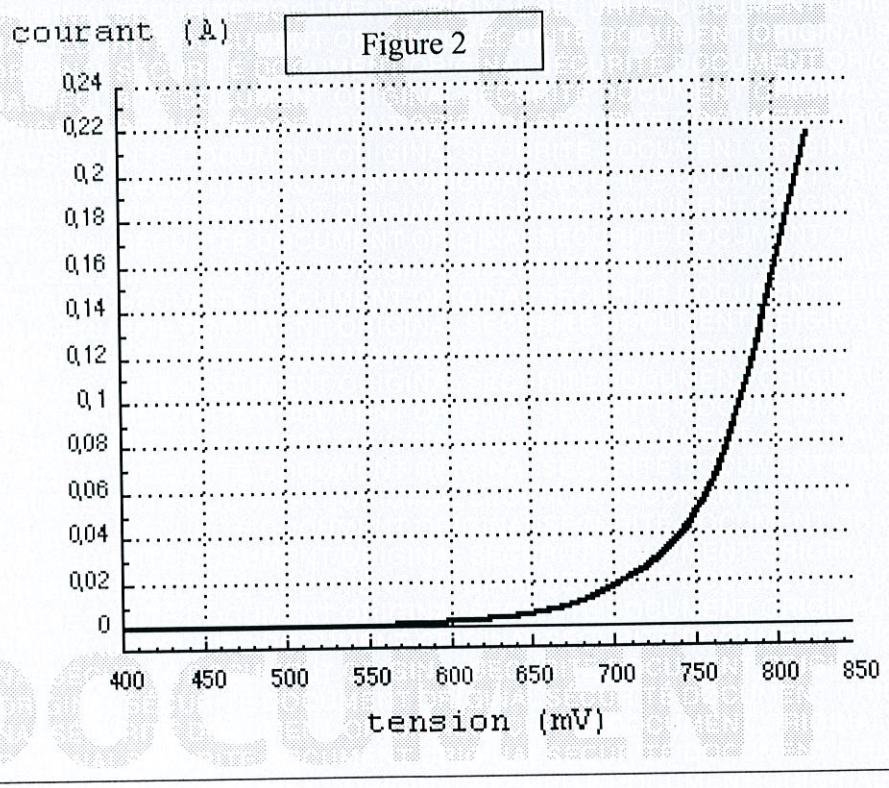


Figure 2

LIF4 : Initiation aux bases de données
Contrôle terminal

21 janvier 2010
durée : 1h30
documents interdits

Exercice 1:

On considère le schéma suivant, utilisé pour la base de donnée d'un magasin :
- Article(code, denomination, prix)
- Rangement(rayon, code, quantite)
- Achat(numero , code, quantite)

La relation Article contient la description (dénomination et prix) de chaque article indentifié par son code barre. La relation Rangement permet de déterminer quel article est disponible en quelle quantité dans quel rayon. Le rayon est identifié par son nom. On supposera qu'un article peut apparaître dans au plus un rayon. La relation Achat associe à chaque passage en caisse, identifié par son numéro, l'ensemble des articles achetés lors de ce passage, avec la quantité correspondante.

Traduire en SQL chacune des requêtes suivantes :

1. Donner la liste (code, dénomination et prix) des articles dont la dénomination contient 'ecran', triés par ordre de prix.
2. Donner la dénomination et le prix des articles du rayon 'Produits frais'.
3. Donner les codes des articles qui ne sont pas en rayon.
4. Donner le nombre total d'articles du passe en caisse numéro 5356.
5. Donner la somme payée en tout pour chaque passage en caisse. On souhaite un résultat trié par numéro de passage en caisse.
6. Mettre à jour le contenu des rayons en fonction des achats du passage en caisse numéro 5356.

T.S.V.P →

Exercice 2:

On considère le schéma suivant utilisé pour une base de donnée servant à la gestion d'une collection de morceaux de musique.

- Morceau(fichier, type, duree)
- Auteur(num, fichier)
- Personne(num, nom, prenom, nationalite)

La relation Morceau donne, pour chaque fichier de musique, son type et sa durée en seconde. La relation Personne donne, pour chaque personne identifiée par un numéro, son nom, son prénom et sa nationalité. La relation Auteur associe à chaque fichier les auteurs du morceau correspondant.

Traduire en calcul relationnel les requêtes suivantes, puis en donner un plan d'exécution optimisé :

- Donner les morceaux de jazz ayant au moins un auteur irlandais.
- Donner sont les morceaux (fichiers) ayant au moins deux auteurs différents et une durée de plus de 180 s.
- Donner le(s) plus long(s) morceau(x) de chaque auteur.

Exercice 3:

On souhaite réaliser une base de données pour gérer une médiathèque municipale. Chaque article (livre, CD, DVD, etc) est identifié par un code barre. On en connait le titre, une année de parution, le ou les auteurs (identifiés par un numéro), le type et une cote qui permet de le repérer physiquement. Cette cote est composée d'un numéro de rayon et de trois lettres. Certains articles sont disponibles en plusieurs exemplaires, qui partagent alors les mêmes informations sauf le code barre. On souhaite pouvoir effectuer une recherche d'article par titre, par année, par nom/prénom d'auteur ou bien par série (ex : DVDs de série télévisée, série de bande dessinées). Pour chaque article, on souhaite pouvoir suivre l'historique des réparations qui ont été faites dessus. On souhaite bien sûr pouvoir gérer les différents emprunts effectués. Un abonné à la bibliothèque est identifié par un numéro et possède un mot de passe pour consulter ses prêts en ligne. On connait le nom, le prénom, la date de naissance, l'adresse de chaque abonné. On connait également la date d'expiration de son abonnement. Pour chaque emprunt, on connait l'abonné, les articles empruntés, la date d'emprunt, la date de retour prévue et la date de retour effectif. Il est possible de demander une prologation d'une semaine, une seule fois par emprunt. Enfin, un abonné peut faire une réservation sur un article en cours d'emprunt. Il est possible d'avoir plusieurs réservations par article, classées par ordre d'ancienneté.

Proposer un diagramme entité-association permettant de modéliser cette base de données.

INF 2601 C

Université Claude Bernard Lyon 1 Juin 2010

LIF4 EXAMEN

Semestre de printemps

Durée 1h30

Documents non autorisés

Exercice 1 : Requêtes SQL (10 points)

Soit la base de données cinémas suivante :

ENSEIGNE (NUMENS, NOMENS)
CINEMA (NUMCINE, NOMCINE, NUMENS, VILLE, REGION)
SALLE (NUMCINE, NUMS, CAPACITE)
CLIENT (NUMCLI, NOMCLI, MAILCLI, DATEDEBCF, DATEFINCF)
BILLET (NUMCLI, NUMCINE, NUMS, DATESEANCE, HEURESEANCE NBADULTE, NBENFANT)

La table ENSEIGNE stocke les informations relatives aux enseignes de cinéma (par exemple 'Pathé', 'Gaumont', 'UGC' …). Une enseigne est identifiée par un numéro 'NUMENS' et on stocke les informations sur le nom de l'enseigne 'NOMENS'.
La table CINEMA stocke les informations relatives aux cinémas de France. Un cinéma (par exemple : Ciné Cité de Lyon, Astoria, La fourmi …) est identifié par un numéro 'NUMCINE' et on stocke les informations sur le nom du cinéma 'NOMCINE', la ville où se trouve le cinéma 'VILLE' ainsi que le nom de la région 'REGION'. Un cinéma appartient forcément à une enseigne qu'il référence *via* l'attribut 'NUMENS'.
La table SALLE stocke les informations relatives aux salles de cinéma. Une salle est identifiée par un numéro 'NUMS' et l'identifiant du cinéma dans lequel elle se trouve 'NUMCINE'. On dispose également des informations sur la capacité maximale de la salle 'CAPACITE'.
La table CLIENT stocke les informations sur les clients. Un client est identifié par un numéro 'NUMCLI'. Pour chaque client, on dispose des informations concernant son nom 'NOMCLI', son adresse email 'MAILCLI' et la date de début 'DATEDEBCF' et la date de fin 'DATEFINCF' de validité de la carte de fidélité).
La table BILLET stocke les informations relatives aux ventes de billets. Un billet est valable pour un client donné 'NUMCLI', pour un cinéma donné 'NUMCINE' pour une salle donnée 'NUMS', pour une date donnée 'DATESEANCE' et pour une heure donnée 'HEURESEANCE'. On stocke également le nombre de personnes associées au billet : le nombre d'adultes 'NBADULTE', et le nombre d'enfants 'NBENFANT'. Il est important de noter que le nombre total de personnes associées à un billet correspond à la somme des valeurs de 'NBADULTE' et de 'NBENFANT'

Remarque : dans ce contexte, nous partons du principe que les informations relatives à un client sont enregistrées lors de l'achat de son premier billet de cinéma, et qu'ensuite pour acheter un autre billet, le client s'identifie via son numéro de client 'NUMCLI'.

Question 1 : (1pt)
Donner un exemple de clé étrangère qu'il serait nécessaire de déclarer dans le SGBD implantant les 5 tables de la base cinémas.

Les requêtes des questions 2 à 8 sont à écrire en SQL.

Question 2 : (0.5pt)

Donner le nom des cinémas de LYON.

Question 3 : (1pt)

Pour chaque cinéma se trouvant à LYON, donner son nom et sa capacité totale (la capacité totale résulte du cumul des capacités des salles du cinéma).

Question 4 : (1pt)

Pour chaque cinéma de la région Rhône-Alpes, donner son nom et le nombre de salle de capacité supérieure à 250 sièges.

Question 5 : (1.5pt)

Donner les villes de la région Rhône-Alpes qui ont au moins 3 salles de capacité supérieure à 250 sièges.

Question 6 : (2pt)

Donner en SQL, le nom des clients ayant vu un film dans un cinéma de LYON appartenant à l'enseigne UGC durant la séance de 20h00 le mercredi 17 juin 2010 dans une salle de capacité de plus de 200 sièges.

Question 7 : (1pt)

Donner le nom des clients non joignables par e-mail (champ MAILCLI non renseigné).

Question 8 : (2pt)

Donner le nom des clients qui n'ont jamais été dans un cinéma de l'enseigne UGC.

Exercice 2 : Schéma Entité/Association (6 points)

La fédération internationale de cyclisme désire mettre au point une base de données. Celle-ci comporte des informations sur les différents coureurs, les équipes, les résultats obtenus aux différentes courses organisées.

Les coureurs sont identifiés par leur nom et leur prénom, on connaît leur taille, leur date de naissance et l'équipe à laquelle ils appartiennent actuellement. Une équipe est identifiée par son nom, elle possède un budget annuel, un directeur sportif dont on connaît le nom, le prénom et la date de naissance. Elle est financée par des sponsors qui peuvent varier selon les années et dont on connaît le nom, l'adresse et le domaine d'activité.

Une course correspond à un nom de course et une année d'édition (ex. « Tour de France », 1968), on en connaît la distance totale à parcourir. Elle peut comporter une ou plusieurs étapes, dont on connaît le numéro d'ordre (ex. « 3è étape »), la date, le type (ex. « Contre la montre individuel »), la ville de départ et celle d'arrivée. Pour chaque coureur ayant participé à une étape d'une course, on connaît le classement qu'il a obtenu lors de cette étape. Pour chaque course, on connaît le vainqueur final et l'équipe à laquelle il appartenait à ce moment là.

Question1 : (3.5pt)

Faire le schéma entité-association correspondant aux besoins de la fédération internationale de cyclisme.

Question 2 : (2.5pt)

En déduire le schéma relationnel de la base de données correspondante, sans oublier de préciser les clés de chaque relation.

Exercice 3 : Transactions (4 points)

Soient cinq transactions T1, T2, T3 et T4 et trois granules X, Y, Z. On considère l'exécution suivante :

L1(X) L3(Y) L4(Y) E2(X) E3(Y) L4(Z) L3(Y) E4(Z) L1(Z) E1(Z) E3(X) L3(X) L3(Z)

où Li(G) correspond à une lecture du granule G par la transaction Ti et Ei(G) correspond à une écriture du granule G par la transaction Ti avec G $\in$ {X, Y, Z} et i = 1,...4.

Question 1 : (0.5pt)

Donner une définition de la notion d'actions conflictuelles.

Question 2 : (2pt)

On considère l'exécution précédente.

Construire le graphe de précédences de cette exécution.

Cette exécution est-elle sérialisable ? Justifier

Question 3 : (1.5pt)

On considère que l'algorithme de verrouillage à 2 phases est appliqué au niveau du moteur transactionnel et qu'il en résulte l'exécution précédente.

Est-ce que cette exécution risque de générer un interblocage ? Justifier

INF 2002 L

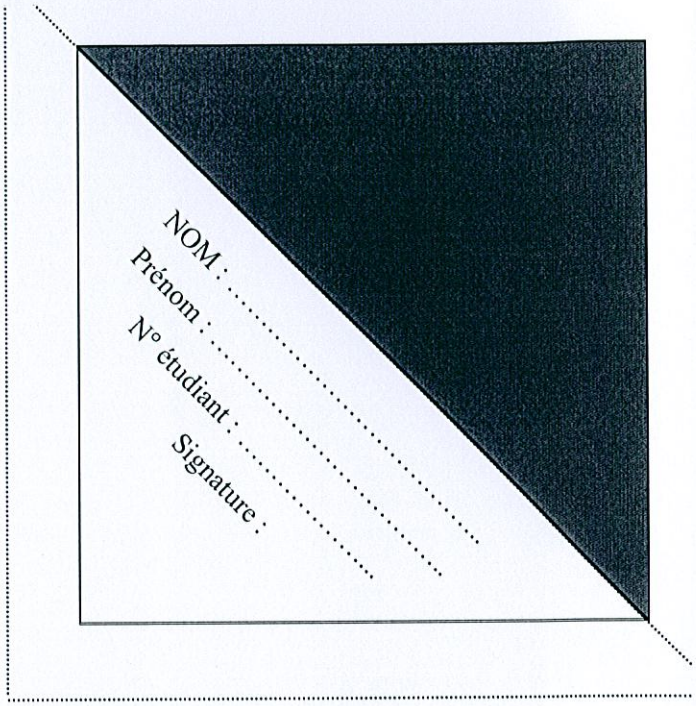
Université Claude Bernard Lyon 1
Licence Sciences, Technologies, Santé – L2
Année 2009-2010, 1er semestre

LIF5 – Algorithmique & Programmation procédurale

Contrôle final du 19 janvier 2010

Durée : 1h30

Note :



Documents, calculatrices, ordinateurs, lecteurs mp3 et téléphones portables interdits.

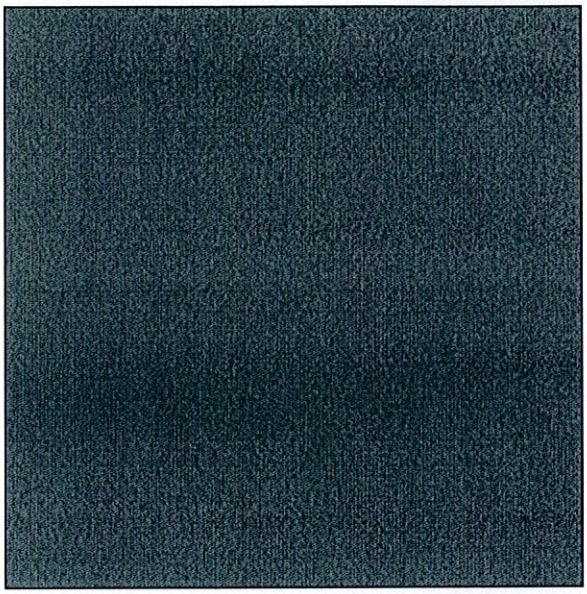
Le barème est donné à titre indicatif.

Travaillez au brouillon d'abord de sorte à rendre une copie propre. Nous ne pouvons pas vous garantir une copie supplémentaire si vous vous trompez.

Exercice 1 : Questions diverses (6 points)

Chaque question est sur un point. Pour les questions à choix multiples : 0 si une proposition fausse est cochée ; sinon, pourcentage de propositions justes cochées.

1. Dessiner l'arbre binaire de recherche obtenu lorsqu'on insère successivement aux feuilles de l'arbre les éléments suivants : 15, 20, 17, 8, 12, 6, 32, 10.



2. Parmi les tableaux suivants, lequel ou lesquels sont structurés en tas binaire décroissant ?
   - ☐ {7, 6, 5, 4, 3, 2 ,1}
   - ☐ {4, 2, 6, 1, 3, 5, 7}
   - ☐ {1, 2, 3, 4, 5, 6, 7}
   - ☐ {7, 5, 3, 1, 2, 4, 6}
   - ☐ {7, 5, 6, 2 ,1, 3, 4}
   - ☐ {4, 6, 7, 3, 5, 1, 2}
   - ☐ {1, 2, 3, 4, 7, 6, 5}

3. Parmi les structures de données suivantes, laquelle ou lesquelles permettent d'accéder en temps constant au *i*-ème élément, quel que soit *i* ?
   - ☐ tableau dynamique
   - ☐ tableau statique
   - ☐ liste simplement chaînée circulaire avec sentinelle
   - ☐ liste simplement chaînée sans sentinelle
   - ☐ liste doublement chaînée sans sentinelle

4. Le coût d'un ajout en queue dans un tableau statique (de taille > k) contenant déjà k éléments est au pire :
   - ☐ $O(k^2)$
   - ☐ O(k.ln(k))
   - ☐ O(ln(k))
   - ☐ O(k)
   - ☐ O(1)

5. Le coût d'un ajout en queue dans un tableau dynamique contenant déjà k éléments est au pire :
   - ☐ $O(k^2)$
   - ☐ O(k.ln(k))
   - ☐ O(ln(k))
   - ☐ O(k)
   - ☐ O(1)

6. Dans la fonction suivante :

```
int mystere(double tab[], int n,
double e)
{
  int i, b;
  b = 0;
  i = 0;
  while ((b == 0) && (i < n))
  {
    if (tab[i] == e){b = 1;}
    i = i + 1;
  }
  return b;
}
```

Combien de comparaisons de doubles effectue-t-on dans le cas le plus défavorable ? .....

Combien de comparaisons de doubles effectue-t-on dans le cas le plus favorable ? .....

Exercice 2 : Inversion d'une liste chaînée (5 points)

Dans cet exercice, on considère un module Liste permettant de gérer des listes de « doubles » simplement chaînées, non circulaires, sans sentinelles.

```
typedef double Elem;
struct sCellule
{
  Elem info;
  struct sCellule *suivant;
};
typedef struct sCellule Cellule;
```

```
struct sListe
{
  Cellule *prem;
  unsigned int nb_elements;
};
typedef struct sListe Liste;
```

Ecrivez une procédure C qui prend une liste en paramètre et qui la modifie de sorte à inverser l'ordre de ses cellules, sans faire aucun appel à malloc ni à free. Vous devez simplement modifier le chaînage des cellules. En clair, une solution qui définirait et appellerait les procédures ajoutEnTete, suppressionEnTete, ajoutEnQueue, suppressionEnQueue telles qu'elles ont été vues en TD ne sera pas acceptée.

```
/* Préconditions : ..................................................

   Postconditions : ................................................... */

void inverse(.....................) /* complétez la liste des paramètres */
{
  /* Complétez les variables locales et le code de cette procédure. */
  /* Indications : vous pouvez décrocher temporairement les cellules de
     la liste pour les chaîner dans le bon ordre dans une liste auxiliaire. */









}
```

Exercice 3 : Procédures itératives sur les arbres binaires (9 points)

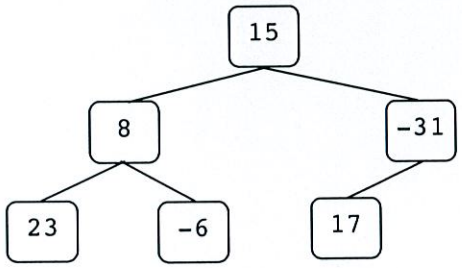
Dans cet exercice, on considère le module Arbre permettant de gérer des arbres binaires d'entiers signés.

```
typedef int Elem;
struct sNoeud {
  Elem info;
  struct sNoeud * fg;
  struct sNoeud * fd;
};
typedef struct sNoeud Noeud;
```

```
struct sArbre {
  Noeud * adRacine;
  int nbElemDansArbre;
};
typedef struct sArbre Arbre;
```

1. On considère un algorithme **itératif** pour afficher les éléments d'un arbre binaire. Cet algorithme utilise deux piles **d'adresses de nœuds**, qu'on appellera A et B :
   - on place la racine dans la pile B,
   - tant qu'on n'a pas épuisé la pile B, on déplace le sommet de la pile B vers la pile A, et on empile son fils gauche (s'il existe) puis son fils droit (s'il existe) dans la pile B, finTantQue
   - une fois la pile B vide, on dépile la pile A jusqu'à la vider, en affichant au fur et à mesure les éléments contenus dans les nœuds.

   a) Quel affichage obtiendrait-on pour l'arbre suivant ?



   b) Cet algorithme est une version itérative d'un type particulier de parcours d'arbre binaire, lequel ?

   c) Donnez l'implantation en langage C de cet algorithme. La procédure devra être itérative et non récursive, et devra pouvoir être appelée sur un arbre vide. Vous pourrez appeler les sous-programmes suivants sans en donner le code :

```
void afficherElem(Elem e);
/* Précondition : aucune
   Postcondition : la valeur de e est affichée sur la sortie standard,
   suivie d'un espace */

void initialiserPile(Pile *pP);
/* Précondition : *pP non préalablement initialisée
   Postcondition : *pP initialisée en pile vide */

void testamentPile(Pile *pP);
/* Précondition : *pP préalablement initialisée
   Postcondition : *pP prête à disparaître (ne doit plus être utilisée) */
```

```
void empiler(Pile *pP, Noeud * adrNoeud);
/* Précondition : *pP initialisée
   Postcondition : adrNoeud est placée au sommet de (*pP) */

void depiler(Pile *pP);
/* Précondition : *pP initialisée, *pP non vide
   Postcondition : le sommet de *pP est dépilé */

Noeud * consulterSommet(Pile P);
/* Précondition : P non vide
   Résultat : adresse située au sommet de P */

int testPileVide(Pile P);
/* Précondition : P initialisée
   Résultat : 1 si P est vide, 0 sinon */
```

```
/* Précondition : ..................................................

   Postcondition : ..................................................... */

void affichageIteratif(.....................) /* paramètre(s) à compléter */
{
   Pile pileA, pileB;
   Noeud * n;

   /* complétez le code de la procédure */









}
```

2. On s'intéresse à présent à la destruction de tous les nœuds d'un arbre binaire. Ecrivez une procédure C **itérative** qui vide complètement l'arbre binaire qu'on lui passe en paramètre. Là encore, vous pouvez si besoin appeler les sous-programmes de la question 1.c) sans en donner le code.

```
/* Précondition : l'arbre passé en paramètre est initialisé, il peut être vide

   Postcondition : tous les nœuds de l'arbre sont supprimés (mémoire libérée),
   l'arbre est donc vide mais laissé dans un état « propre » (permettant
   d'y insérer ultérieurement de nouveaux nœuds, par exemple).  */

void viderArbreIteratif(.....) /* paramètres à compléter */
{
   /* à compléter */









}
```

INF 2002 C

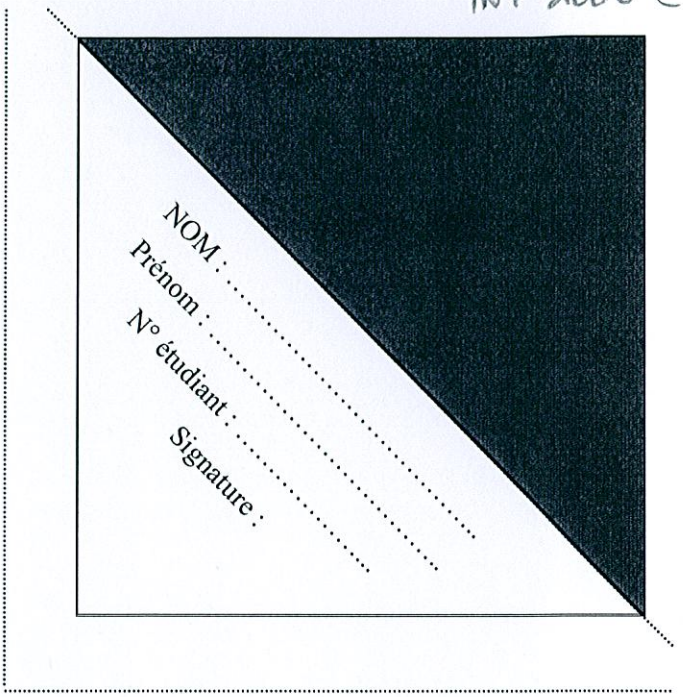
Université Claude Bernard Lyon 1
Licence Sciences, Technologies, Santé – L2
Année 2009-2010, 1er semestre

LIF5 – Algorithmique & Programmation procédurale

Contrôle final du 17 juin 2010

Durée : 1h30

Note :



Documents, calculatrices, ordinateurs, lecteurs mp3 et téléphones portables interdits.

Le barème est donné à titre indicatif.

Travaillez au brouillon d'abord de sorte à rendre une copie propre. Nous ne pouvons pas vous garantir une copie supplémentaire si vous vous trompez.

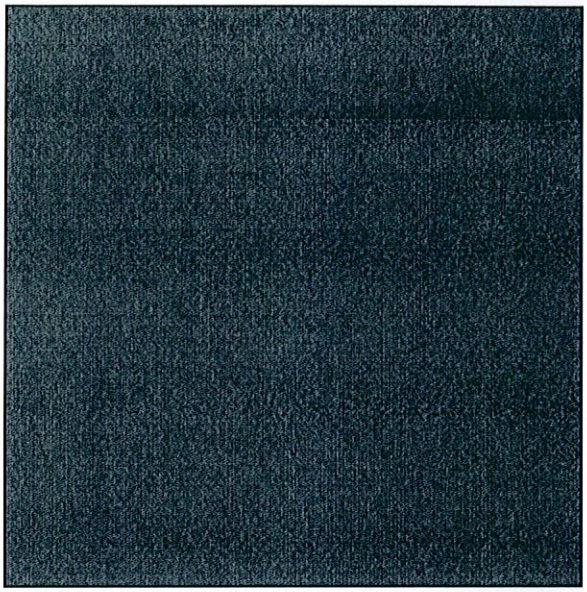
Exercice 1 : Questions diverses (5 points)

Chaque question est sur un point. Pour les questions à choix multiples : 0 si une proposition fausse est cochée ; sinon, pourcentage de propositions justes cochées.

1. Lorsqu'on insère aux feuilles d'un arbre binaire de recherche les éléments 3, 15, 7, 26, 5, 12, 11, 8, 1 dans cet ordre, combien de niveaux obtient-on ?

2. Considérons une file d'attente à priorité gérée à l'aide d'un tas binaire décroissant. Lorsqu'on y insère successivement des éléments de priorités 2, 1, 8, 2, 2, 5, 10, 8, quel tableau obtient-on ?

3. Considérons le module TableauDynamique tel qu'il a été vu en cours, TD et TP. Que se passe-t-il si l'on initialise deux fois de suite un tableau dynamique ?
   - ☐ un "buffer overflow"
   - ☐ un "stack overflow"
   - ☐ une erreur de compilation
   - ☐ une fuite de mémoire
   - ☐ une erreur d'entrée-sortie



4. Dans le cas le plus défavorable, le coût d'une insertion aux feuilles d'un arbre binaire de recherche contenant déjà k éléments est :
   - ☐ $O(k^2)$
   - ☐ $O(k.\ln(k))$
   - ☐ $O(\ln(k))$
   - ☐ $O(k)$
   - ☐ $O(1)$

5. Le coût d'un ajout en tête dans une liste doublement chaînée contenant déjà k éléments est :
   - ☐ $O(k^2)$
   - ☐ $O(k.\ln(k))$
   - ☐ $O(\ln(k))$
   - ☐ $O(k)$
   - ☐ $O(1)$

Exercice 2 : Fusion de deux listes chaînées triées (9 points)

Dans cet exercice, on considère un module Liste permettant de gérer des listes de « doubles » simplement chaînées, non circulaires, sans sentinelles.

```
typedef double Elem;
struct sCellule
{
  Elem info;
  struct sCellule *suivant;
};
typedef struct sCellule Cellule;
```

```
struct sListe
{
  Cellule *premier;
};
typedef struct sListe Liste;
```

a. Ecrivez une fonction C qui prend une liste en paramètre, décroche la cellule de tête en maintenant la liste valide et renvoie l'adresse de cette cellule décrochée.

```
/* Préconditions : ..................................................

Postcondition :...................................................

Résultat : .......................................................

.....................................................................*/

Cellule * dechaineTete(.....................................................)
/* complétez la liste des paramètres */
{
  /* Complétez les variables locales et le code de cette procédure. */









}
```

b. Ecrivez en C une procédure itérative qui prend en paramètres, une liste et un pointeur sur une cellule valide et chaîne en queue la cellule. Il est interdit de faire appel à une fonction ou procédure quelconque.

```
/* Préconditions : ..................................................

   Postconditions : ................................................

.....................................................................

*/


void ChaineEnQueue(.....................................)
 /* complétez la liste des paramètres */
{
  /* Complétez les variables locales et le code de cette procédure. */









}
```

c. A partir d'une liste vide, quel est le coût de l'insertion de n éléments via n appels à ChaineEnQueue, en termes de nombre d'affectations de pointeurs ? (Justifiez).

d. Ecrivez en C une procédure qui fusionne deux listes triées par ordre croissant, L1 et L2, pour obtenir une liste triée L3 par ordre croissant. A la fin de la procédure, les listes L1 et L2 sont vides et il n'y a eu ni destruction ni création de nouvelles cellules.

Exemple : l1 = (2, 7, 18, 19, 27)
l2 = (3, 5, 9, 20, 21, 28, 30, 35, 39)
l3 =(2, 3, 5, 7, 9, 18, 19, 20, 21, 27, 28, 30, 35, 39)

```
/* Préconditions : ..................................................

   Postconditions : ..................................................

*/
void fusionListesTriees(..................................................)
/* complétez la liste des paramètres */
{/* Complétez les variables locales et le code de cette procédure. */









}
```

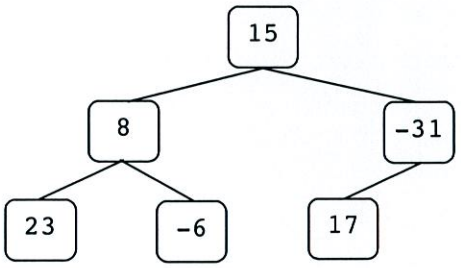
Exercice 3 : Procédures sur les arbres binaires (6 points)

Dans cet exercice, on considère le module Arbre permettant de gérer des arbres binaires d'entiers signés.

```
typedef int Elem;
struct sNoeud {
  Elem info;
  struct sNoeud * fg;
  struct sNoeud * fd;
};
typedef struct sNoeud Noeud;
```

```
struct sArbre {
  Noeud * adRacine;
};
typedef struct sArbre Arbre;
```

a. Quels sont les différents types de parcours qu'on peut effectuer pour afficher l'arbre suivant ? Quel est le résultat obtenu pour chaque parcours appliqué à l'arbre suivant ?



b. Ecrire en C la fonction récursive qui teste si deux arbres binaires A et B sont identiques (cadre-réponse page suivante). Deux arbres A et B sont identiques si :
   - Il y a égalité de l'information aux racines,
   - L'arbre enraciné au fils gauche de A est identique à celui enraciné au fils gauche de B.
   - L'arbre enraciné au fils droit de A est identique à celui enraciné au fils droit de B.

```
/* Précondition : ..................................................

   Résultat : ..................................................

 */
int testEgaliteApartirDeNoeud(.....................................)
/* complétez la liste des paramètres */
{/* Complétez les variables locales et complétez le code de la procédure */









}




/* Précondition : ..................................................

   Résultat : ..................................................

 */
int testEgalite(Arbre a, Arbre b)
{
  return testEgaliteApartirDeNoeud(.....................................);
  /* complétez la liste des paramètres effectifs */
}
```

INF 3003 L

LIF10 - Fondements des bases de données

Contrôle Continu final, jeudi 21 janvier 2010

Durée : 90 minutes. Documents interdits. Toutes les réponses doivent être justifiées.

1 Normalisation (10pts)

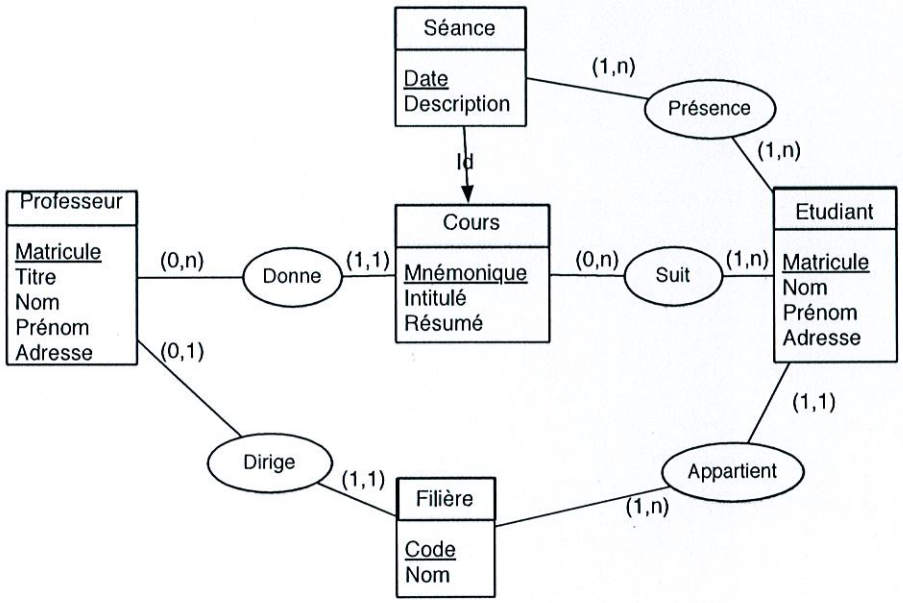
La TCL désire informatiser la gestion quotidienne de ses bus. Chaque soir, les conducteurs ramènent les autobus au dépôt et complètent une feuille de route avec la date du jour, leur numéro de matricule, leur nom, le numéro d'immatriculation du bus et son kilométrage. Le soir, ces feuilles sont analysées pour en déduire les entretiens à réaliser le lendemain (vidange, graissage, parallélisme, ...). Chaque entretien d'autobus fait l'objet d'un ordre d'entretien daté et numéroté précisant les opérations à effectuer (n°opération, type, description, quantité), exemple : (23987, 'changement de pneu', 'changer les pneus arrières', 2). Le numéro d'opération est un numéro unique identifiant l'opération effectuée sur un bus à un moment précis. Le lendemain, l'opération est réalisé par un ensemble de mécaniciens. Parmi les mécaniciens assignés à l'ordre d'entretien, un seul est désigné responsable de l'opération d'entretien et doit préciser, pour chacune des opérations dont il a la responsabilité, la date de l'opération, si elle a pu être menée à bien ou non et quels sont les mécaniciens qui sont intervenus.

On désire mémoriser l'historique des feuilles de route ainsi que l'historique des réparations réalisées. Durant une journée, un bus est conduit par un seul conducteur, mais l'inverse n'est pas toujours vérifié. De plus un mécanicien n'est jamais conducteur, et un conducteur n'est jamais mécanicien. Ce sont les deux seuls types d'employés apparaissant dans cette base de données.

1. Qu'est ce qu'une dépendance fonctionnelle, une dépendance multi-valuée, et une dépendance d'inclusion ?
2. A partir du texte précédent, dressez la liste de toutes les dépendances possibles.
3. Qu'est ce qu'une clé ? clé minimale ?
4. Enumérez toutes les clés minimales de la relation universelle.
5. Quelle est le forme normale de la relation universelle ?
6. Proposez une normalisation sans perte d'information si possible.

2 Modèle entité-association (5pts)

Traduisez le schéma entité-relation suivant dans le modèle relationnel.



3 Relation d'Armstrong (5pts)

Soit un schéma de relation $R = ABCDE$ et un ensemble de dépendances fonctionnelles $F = \{AB \to C; AC \to B; C \to D; D \to C; B \to CD; AD \to BC; E \to CD; C \to E; D \to E\}$.

Pour montrer les redondances d'un tel schéma, nous souhaitons fournir un exemple représentatif vérifiant exactement F^+ pour l'aider à normaliser ou détecter les anomalies. Pour cela, nous allons créer une *relation d'Armstrong* pour l'ensemble F.

1. Calculez l'ensemble complet des fermés de F, défini par $Cl(F) = \{X^+, X \subseteq R\}$. Justifiez la méthode utilisée.
2. Appliquez l'algorithme suivant pour construire la relation d'Armstrong.

Algorithm 1: Relation d'Armstrong pour un ensemble de DF

Data: Un schéma de relation R, un ensemble de DF F sur R
Result: Une relation d'Armstrong r de F sur R

```
begin
    forall A ∈ R do
        t[A] = 0;
    r = {t};
    i = 1;
    forall X ∈ Cl(F) − R do
        forall A ∈ X do
            if A ∈ X then
                t[A] = 0;
            else
                t[A] = i;
        r = r ∪ {t};
        i := i + 1;
    return (r);
end
```

Durée : 1H00
Tous documents papier autorisés. Appareils électroniques non autorisés.

1. On considère sur $\Sigma = \{a,b\}$ le langage L des mots dont la première lettre et la dernière lettre sont différentes.
   a) Donnez une expression rationnelle pour décrire L.
   b) Donnez une expression rationnelle pour décrire le complément de L.
   c) Dessinez un automate non déterministe qui reconnait L.
   d) Déterminisez l'automate précédent.

2. L'utilisation du lemme de l'étoile n'est pas le seul moyen de montrer la non-rationalité de certains langages. Nous allons étudier ici le *non-pumping lemma*.
   a) Rappelez le théorème de Myhill – Nerode vu en cours.
   b) Soit Σ un alphabet, $L \subset \Sigma^*$ un langage rationnel et w un mot de Σ^*. On rappelle que [u] dénote la classe d'équivalence du mot *u* suivant le langage *L*. Montrez qu'il existe deux entiers *m* et *n* distincts strictement positifs tels que $[w^m] = [w^n]$. Déduisez-en le théorème :

> Soit $L \subset \Sigma^*$ un langage rationnel. Pour tout mot w de Σ^*, il existe deux entiers $m > n > 0$ tels que $\forall u \in \Sigma^*$, $w^m u \in L$ si et seulement si $w^n u \in L$.

   c) Utilisez le théorème précédent pour montrer que les langages suivants ne sont pas rationnels :
      - $L_1 = \{a^n b^n \mid n \geq 0\}$
      - $L_2 = \{uu^R w \mid u, w \in \{a,b\}^*\}$

3. Soit la grammaire algébrique $G = (V, \Sigma, R, S)$ avec
   $V = \{S, X\}$, $\Sigma = \{a, b\}$, $R = \{S \rightarrow XbX, X \rightarrow S \mid XaXbX \mid XbXaX \mid e\}$
   a) Quel est langage L généré par G ? Exprimez votre réponse de manière formelle ou non. Prouvez votre réponse (rappel : il y a deux démonstrations à faire).
   b) Montrez que le langage $L' = \{a^m b^n \mid m \neq n\}$ est algébrique.
   c) Montrez que la classe des langages rationnels est stable par différence ensembliste.
   d) Déduisez de ce qui précède que L' est non rationnel.