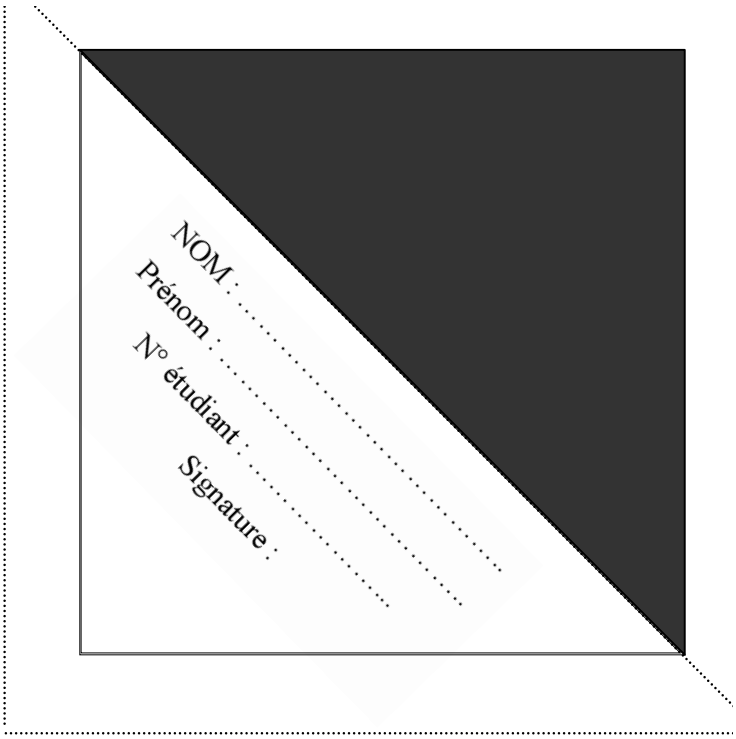
Université Claude Bernard Lyon 1
Licence Sciences, Technologies, Santé – L2
Année 2010-2011, 1er semestre

LIF5 – Algorithmique & Programmation procédurale

Contrôle final
18 janvier 2010

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. Dessiner l'arbre binaire de recherche obtenu lorsqu'on insère successivement aux feuilles de l'arbre les éléments du jeu de données suivant : 12, 25, 14, 5, 18, 9, 41, 8.

2. Dans quel ordre faut-il fournir le jeu de données de la question 1 pour obtenir un arbre binaire de recherche totalement dégénéré (déséquilibré) ?

3. Parmi les structures de données suivantes, laquelle ou lesquelles accèdent en temps linéaire (O(N)) à un élément quelconque?
   - ☐ tableau dynamique
   - ☐ liste doublement chaînée
   - ☐ liste simplement chaînée circulaire avec sentinelle
   - ☐ arbre binaire de recherche bien équilibré.

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 dans un arbre binaire de recherche contenant déjà k éléments est au pire :
   - ☐ $O(k^2)$
   - ☐ O(k.ln(k))
   - ☐ O(ln(k))
   - ☐ O(k)
   - ☐ O(1)

Exercice 2 : Suppression d'éléments dans une liste doublement chaînée (5 points)

Dans cet exercice, on considère un module Liste permettant de gérer des listes doublement chaînées, non circulaires.

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

```
struct sListe
{
  Cellule *adPremiere;
  Cellule *adDerniere;
};
typedef struct sListe Liste;
```

Ecrivez une procédure C qui prend une liste **l** et un élément **e** en paramètres et qui libère toutes les cellules de la liste l contenant **e**. Remarque : le champ « pred » de la première cellule et le champ « suivant » de la dernière cellule pointent sur NULL.

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

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

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


void supprime(.....................................................)
/* complétez la liste des paramètres */
{

/* Complétez les variables locales et le code de cette procédure. Vous ferez
attention aux cas particuliers où e est en tête ou en fin de liste de liste */









}
```

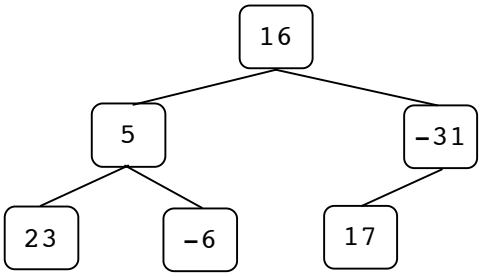
Exercice 3 : Procédures sur les arbres binaires (10 points)

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

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

```
typedef struct sNoeud Noeud;


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

1. Quel affichage obtient-on pour l'arbre suivant lorsqu'on effectue un parcours **postfixé** ?

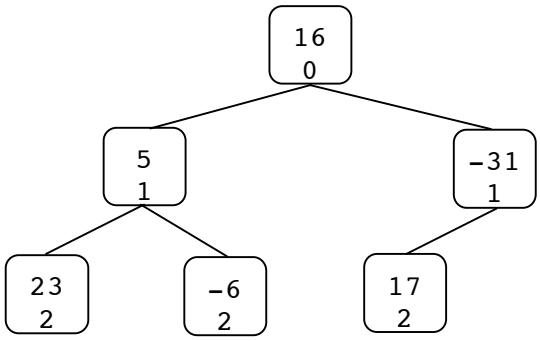


2. Donnez l'implantation en langage C du parcours **en largeur** pour afficher un arbre binaire. La procédure devra être **itérative** et non récursive. Vous pourrez appeler les sous-programmes suivants sans en donner le code :

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

void initialiserFile(File *pF);
/* Précondition : *pF non préalablement initialisée
       Postcondition : *pF initialisée en File vide */

void testamentFile(File *pF);
/* Précondition : *pF préalablement initialisée
       Postcondition : *pF prête à disparaître (ne doit plus être
       utilisée) */
void enFiler(File *pF, Noeud * adrNoeud);
/* Précondition : *pF initialisée
       Postcondition : adrNoeud est placée en fin de (*pF) */

void deFiler(File *pF);
/* Précondition : *pF initialisée, *pF non vide
       Postcondition : le premier élément de *pF est retiré */

Noeud * consulterPremier(File F);
/* Précondition : F non vide
       Résultat : adresse située au début de F */

int testFileVide(File F);
/* Précondition : F initialisée
       Résultat : 1 si F est vide, 0 sinon */
```

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

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

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

void affichageIteratif(.....................................................)
/* paramètre(s) à compléter */
{
  File FileA;
  Noeud * pN;

  /* complétez le code de la procédure */









}
```

3. On veut à présent indiquer dans chaque nœud le niveau dans lequel il se situe, sachant que la racine est au niveau 0 (voir schéma ci-dessous). Ecrivez une procédure **C** qui effectue cette opération. Cette procédure pourra être itérative ou récursive, et pourra utiliser ou non les sous-programmes du module File.




```
/* Précondition : l'arbre passé en paramètre est initialisé, il peut être vide
 Postcondition : tous les nœuds de l'arbre sont numérotés avec le niveau dans
lequel ils sont  */

void numNiveau(.....................................................)
/* paramètre(s) à compléter */
{









}
```

Informatique Graphique (1h)

Aucun document autorisé.

Exercice 1 : Vecteur normal et lumière (cours et TP) (20 minutes)

a) A quoi servent les vecteurs « normales » N que l'on donne à chaque sommet d'un maillage lors de son affichage ? **(1 ligne)**

b) Faite un schéma décrivant la quantité de lumière arrivant sur une portion de surface en fonction de la direction de la lumière L.

c) En déduire la formule de calcul (donnée en cours) de l'éclairage diffus (surface Lambertienne) en fonction de N et de L.

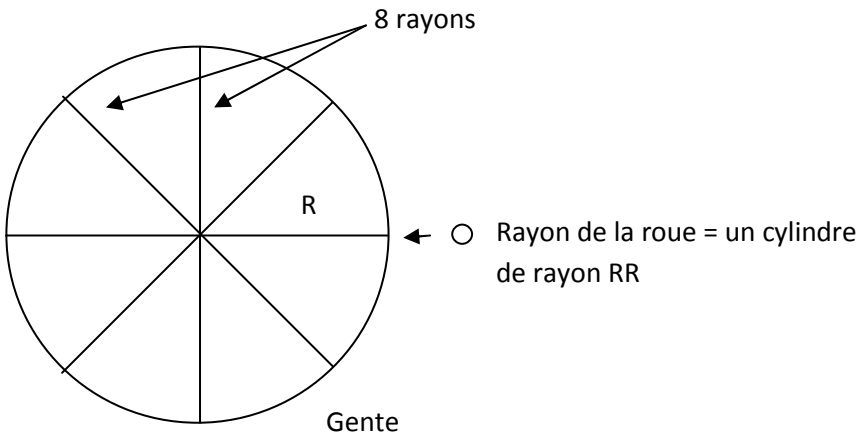
d)) Avec un maillage régulier comme le terrain affiché en TP, comment calcule-t-on les « normales » ? Faites un schéma et donnez une explication sous forme de formule mathématique du calcul de N (pas de code).

Exercice 2 : Affichage d'un vecteur normal (5minutes)

Soit (nx, ny, nz) la normale du point P de coordonnées (x, y, z). Ecrivez le code OpenGL qui permet d'afficher en rouge cette normale au point P, sous la forme d'un segment. Cet affichage permet par exemple de vérifier le calcul de la normale.

Exercice 3 : Roue de vélo (15 minutes)

Vous disposez de la fonction DessineCylindre qui affiche un cylindre de longueur 1 et de rayon 1 sur l'axe des X. Ecrivez le code C/OpenGL qui affiche une roue de vélo. Pour la gente un cylindre de rayon R et de hauteur H. Pour les rayons N cylindres de rayon RR qui partent du centre de la roue vers la gente.



Exercice 4 : réflexion sur la simplification de maillage (15 minutes)

Vous êtes développeur sur le projet de jeu vidéo "*StarWar 18 la revanche*". Les graphistes ont modélisé un vaisseau très détaillé avec de nombreux triangles (image de gauche ci-dessous). Ce vaisseau est utilisé pour afficher un bataillon de milliers de vaisseaux. Après test il apparaît que le jeu ne pourra pas être temps réel car le nombre de triangles est trop important. Vous faites alors remarquer que la plupart des vaisseaux sont trop petits à l'écran pour que l'on distingue précisément tous les détails. Seulement, le joueur étant libre de ses mouvements, il est impossible de prévoir à l'avance les vaisseaux demandant des détails des autres.

L'idée générale est de précalculer plusieurs géométries de vaisseaux allant de très détaillé à peu détaillé et de choisir à la volée celui qui convient le mieux.

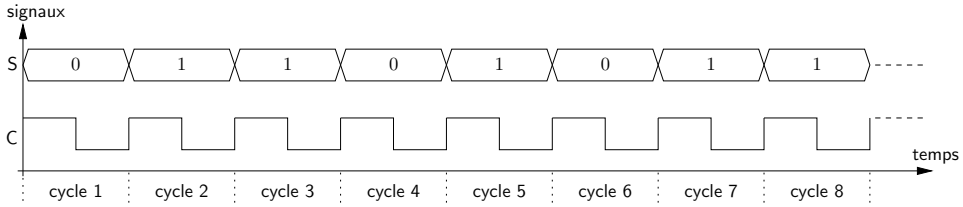
a) Proposez des critères pour décider du niveau de précision d'un vaisseau. **(sous forme de tirets en quelques lignes)**
b) Cette idée permet de gagner en interactivité. Est-ce que vous voyez des inconvénients, des problèmes ? **(quelques lignes)**

CC-TD-F - LIF6 Architecture matérielle et logicielle (INF2003L)

Aucun document autorisé, durée 1h30, barême donné à titre indicatif : il pourra être modifié.
Vendredi 21 janvier 2010, 13h00-14h30, amphi 1 Déambulatoire.

I Un circuit séquentiel (6 pts)

On veut mettre au point un circuit séquentiel pour répéter la séquence 01101 périodiquement. On note S la sortie du circuit. Un chronogramme représentant la valeur de S au cours du temps sera par exemple :



Pour concevoir ce circuit séquentiel, on va le modéliser à l'aide d'un automate fini séquentiel.

Q.I.1 - (0,5 pt) Cinq états sont suffisants pour réaliser l'automate fini séquentiel demandé : pourquoi ?

Comme cinq états sont suffisants, on utilise un registre (à base de bascules flip-flops) à 3 bits pour le stockage de l'état courant $Q = (q_2, q_1, q_0)$ du circuit séquentiel. On choisit la correspondance suivante entre chacun des états de l'automate et sa sortie.

état Q	sortie S
000	0
001	1
010	1
011	0
100	1

Q.I.2 - (0,5 pt) Donnez une représentation graphique de l'automate fini séquentiel demandé (graphique comportant les états, les transitions de l'automate,...).

Q.I.3 - (1 pt) Soit $F = (f_2, f_1, f_0)$ la fonction de transition de l'automate. Donnez la table de vérité de F en fonction de Q. Complétez pour cela le tableau de vérité suivant :

Q			$F(Q)$		
q_2	q_1	q_0	f_2	f_1	f_0
⋮	⋮	⋮	⋮	⋮	⋮

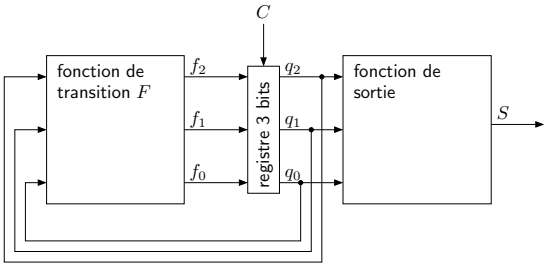
Q.I.4 - (1 pt) Exprimez f_2, f_1 et f_0 en fonction de q_2, q_1 et de q_0 à l'aide de formules booléennes simples.

Q.I.5 - (1 pt) Complétez la table de vérité de S en fonction de $Q = (q_2, q_1, q_0)$ (la valeur pour les états qui ne seront jamais utilisés est fixée à 1, pour simplifier la suite de l'exercice).

Q			$S(Q)$
q_2	q_1	q_0	S
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	1
1	1	0	1
1	1	1	1

Q.I.6 - (1 pt) Exprimez S en fonction de q_2, q_1 et q_0 à l'aide d'une formule booléenne simple.

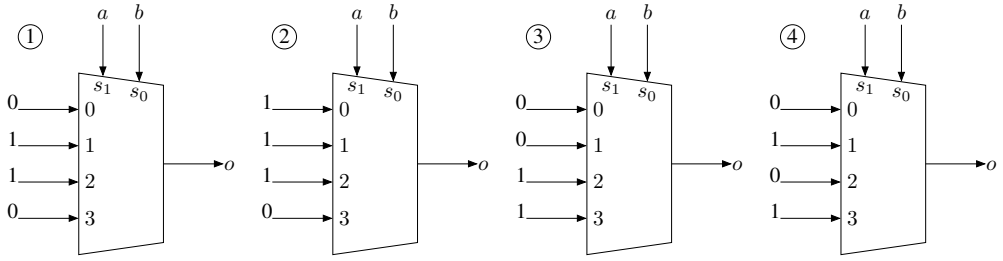
Q.I.7 - (1 pt) Complétez le logigramme ci-dessous, afin d'obtenir le circuit séquentiel demandé. Donnez simplement sur votre copie le logigramme de la fonction de transition et celui de la fonction de sortie, en indiquant bien les noms des signaux d'entrée et de sortie pour chacune.



II Circuits combinatoires (6 pts)

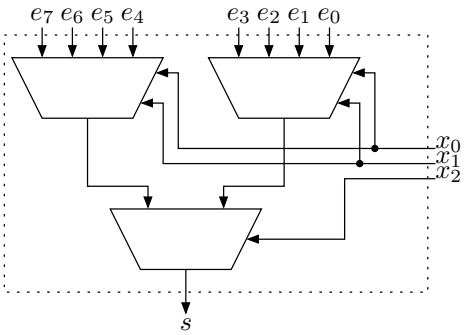
II.1 Multiplexeurs

Q.II.1 - (1 pt) Les circuits ci-dessous sont tous construits autour de multiplexeurs ; pour chacun des circuits, exprimez la sortie o en fonction des entrées a et b à l'aide d'une formule booléenne simple.



II.2 Analyse d'un circuit combinatoire

On considère le circuit combinatoire ci-dessous, construit autour de multiplexeurs :



Q.II.2 - (1 pt) En complétant le tableau ci-dessous, indiquez quelle est la valeur prise par la sortie s en fonction de (x_2, x_1, x_0) :

x_2	x_1	x_0	s
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Q.II.3 - (1 pt) Comment s'appelle le circuit étudié ? Soyez le plus précis possible dans votre réponse.

Q.II.4 - (1 pt) Donnez une formule compacte exprimant la valeur de la sortie s en fonction des entrées, en vous servant des indices indiqués sur la figure.

Q.II.5 - (2 pts) Avec le circuit étudié, on peut en fixant à 0 ou 1 les signaux $e_7, \ldots, e_0$ d'implémenter une fonction donnant la parité des entrées c_2, c_1, c_0 : la sortie prend la valeur 1 si et seulement si le nombre d'entrées égales à 1 parmi c_2, c_1, c_0 est impair. Il vous est demandé de donner sous la forme d'un tableau les valeurs que doivent prendre les entrées $e_7, \ldots, e_0$ pour réaliser cette fonction de parité.

III Programmation en assembleur du LC-3 (8 pts)

Le jeu d'instruction du LC-3 vous est rappelé en fin de sujet.

III.1 Multiplication par une constante

Il vous est recommandé de commencer l'exercice en posant une multiplication en binaire, par exemple celle $(1011)_2$ par $(10)_{10} = (1010)_2$.

Q.III.1 - (4 pts) On considère le programme incomplet ci-dessous :

```
            .ORIG x3000     ; adresse de début de programme
; partie dédiée au code

            A COMPLETER

; partie dédiée aux données et au résultat
n:          .FILL #8        ; entrée n
r:          .BLKW #1        ; espace pour stocker le résultat
            .END
```

Complétez ce programme, de manière à ce qu'il effectue le produit de l'entier naturel à l'adresse désignée par `n` par la constante $(1010)_2$, puis placez le résultat à l'adresse désignée par `r`. Le résultat sera de calculé dans le registre `R0`, avant d'être rangé à l'adresse désignée par `r`. Vous pourrez utiliser `R1` comme registre auxiliaire. Indication : il n'est pas nécessaire d'utiliser de boucles.

III.2 Somme des entiers de 1 à `n`

On considère le programme incomplet ci-dessous :

```
          .ORIG x3000     ; adresse de début de programme
; partie dédiée au code

          ; description du programme à implanter :
          ;   int i;          // indice de boucle, qui sera maintenu dans le registre R0
          ;   int a;          // accumulateur, qui sera maintenu dans le registre R1
          ;   i = n;          // l'entier à l'adresse indiquée par l'étiquette n
          ;   a = 0;          // initialisation de l'accumulateur
          ;   while(i >= 1) {
          ;     a += i;       // on accumule i dans a
          ;     i--;          // on décrémente l'indice de boucle i
          ;   }
          ;   r = a;          // on stocke le résultat à l'adresse indiquée par l'étiquette r

          A COMPLETER

; partie dédiée aux données
n:        .FILL #8        ; entrée n
r:        .BLKW #1        ; espace pour stocker le résultat
          .END
```

Q.III.2 - (4 pts) Complétez ce code, en vous conformant à la partie « `description du programme à implanter` » du listing. Ne reportez sur votre copie que le code correspondant à la partie « `A COMPLETER` ».

syntaxe	**action**	**nzp**	**codage**																
			opcode				arguments												
			F	E	D	C	B	A	9	8	7	6	5	4	3	2	1	0	
NOT DR,SR	DR <- not SR	*	1	0	0	1	DR			SR			1	1	1	1	1	1	
ADD DR,SR1,SR2	DR <- SR1 + SR2	*	0	0	0	1	DR			SR1			0	0	0		SR2		
ADD DR,SR1,Imm5	DR <- SR1 + SEXT(Imm5)	*	0	0	0	1	DR			SR1			1	Imm5					
AND DR,SR1,SR2	DR <- SR1 and SR2	*	0	1	0	1	DR			SR1			0	0	0		SR2		
AND DR,SR1,Imm5	DR <- SR1 and SEXT(Imm5)	*	0	1	0	1	DR			SR1			1	Imm5					
LEA DR,label	DR <- PC + SEXT(PCoffset9)	*	1	1	1	0	DR			PCoffset9									
LD DR,label	DR <- mem[PC + SEXT(PCoffset9)]	*	0	0	1	0	DR			PCoffset9									
ST SR,label	mem[PC + SEXT(PCoffset9)] <- SR		0	0	1	1	SR			PCoffset9									
LDR DR,BaseR,Offset6	DR <- mem[BaseR + SEXT(Offset6)]	*	0	1	1	0	DR			BaseR			Offset6						
STR SR,BaseR,Offset6	mem[BaseR + SEXT(Offset6)] <- SR		0	1	1	1	SR			BaseR			Offset6						
LDI DR,label	DR <- mem[mem[PC + SEXT(PCoffset9)]]	*	1	0	1	0	DR			PCoffset9									
STI SR,label	mem[mem[PC + SEXT(PCoffset9)]] <- SR		1	0	1	1	SR			PCoffset9									
BR[n][z][p] label	Si (cond) PC <- PC + SEXT(PCoffset9)		0	0	0	0	n	z	p	PCoffset9									
NOP	No Operation		0	0	0	0	0	0	0	0 0 0 0 0 0 0 0 0									
JMP BaseR	PC <- BaseR		1	1	0	0	0	0	0	BaseR			0 0 0 0 0 0						
RET (JMP R7)	PC <- R7		1	1	0	0	0	0	0	1	1	1	0 0 0 0 0 0						
JSR label	R7 <- PC ; PC <- PC + SEXT(PCoffset11)		0	1	0	0	1	PCoffset11											
JSRR BaseR	R7 <- PC ; PC <- BaseR		0	1	0	0	0	0	0	BaseR			0 0 0 0 0 0						
RTI	cf. interruptions		1	0	0	0	0 0 0 0 0 0 0 0 0 0												
TRAP Trapvect8	R7 <- PC ; PC <- mem[Trapvect8]		1	1	1	1	0	0	0	0	Trapvect8								
Réservé			1	1	0	1													

Tab. 1 – Récapitulatif des instructions du LC-3

N° de copie : …… 16 juin 2011 – Département Informatique

LIF4 : Initiations aux BD et Réseaux

EXAMEN - Durée 1h30 - Documents non autorisés

Remarques : Pour l'anonymisation, vous devez reporter le numéro de votre copie ci-dessus. Vous devez répondre sur le sujet et glisser le sujet dans votre copie. Les parties sont indépendantes. Le barème est à titre indicatif.

Partie A : Interrogation d'un SGBD (11pts)

On considère une base de données relationnelles représentant la gestion des vols aériens d'une compagnie d'aviation effectuant des vols et définie par le schéma suivant :

AEROPORT (codeA, villeA, pays, secteurA)

- *codeA : un code unique identifiant de l'aéroport*
- *villeA : la ville où se trouve l'aéroport*
- *pays : le pays d'Europe où se trouve l'aéroport*
- *secteurA : le secteur où se trouve le pays. Le domaine de définition de secteur est {'Nord', 'Est', 'Ouest'}*

VOL (numV, dateV, h_dep, h_arr, dep_CodeA, arr_CodeA)

- *numV : numéro de vol*
- *dateV : date de départ d'un vol*
- *h_dep : horaire de départ d'un vol*
- *h_arr : horaire d'arrivée d'un vol*
- *dep_CodeA : code de l'aéroport de départ*
- *arr_CodeA : code de l'aéroport d'arrivée*

AVION (numAV, typeAV, capacite, dateDerVisite, age)

- *numAV : numéro d'avion*
- *typeAV : type d'avion (Boeing 747, Airbus A320 …)*
- *capacite : le nombre de sièges dans l'avion*
- *dateDerVisite : la date de la dernière visite de contrôle*
- *age : le nombre cumulé d'heures de vol de l'avion*

PILOTE (numP, nomP, anneeBrevet, pays)

- *numP : numéro du pilote*
- *nomP : nom du pilote*
- *anneeBrevet : année d'obtention du brevet de pilote de ligne*
- *pays: nationalité du pilote*

AFFECTATION (numV, numAV, numP)

- *numV : numéro de vol*
- *numAV : numéro d'avion*
- *numP : numéro de pilote*

Exercice 1: Requêtes SQL (8 pts)

Ecrire les requêtes suivantes dans le langage SQL :

1) Retourner le code des aéroports du secteur nord.

2) Retourner le nombre de vols ayant décollé de l'aéroport Lyon Saint-Exupéry (code 'LYS') le 15 juin 2011.

3) Retourner le nom du dernier pilote à avoir obtenu son brevet de pilote (version 1)

4) Retourner le nom du dernier pilote à avoir obtenu son brevet de pilote (version 2 différente)

5) Retourner le nom des pilotes dont on ne connaît pas l'année d'obtention de leur brevet de pilote et qui n'ont jamais été affecté à un vol

6) Donner le type d'avion dont la capacité est supérieure strictement à celle d'un avion de type « Airbus A320 » et qui a effectué au moins un vol depuis sa dernière visite.

7) Pour chaque pays, donner le nombre d'avions qui ont atterri le 15 juin 2011 dans un des aéroports du pays.

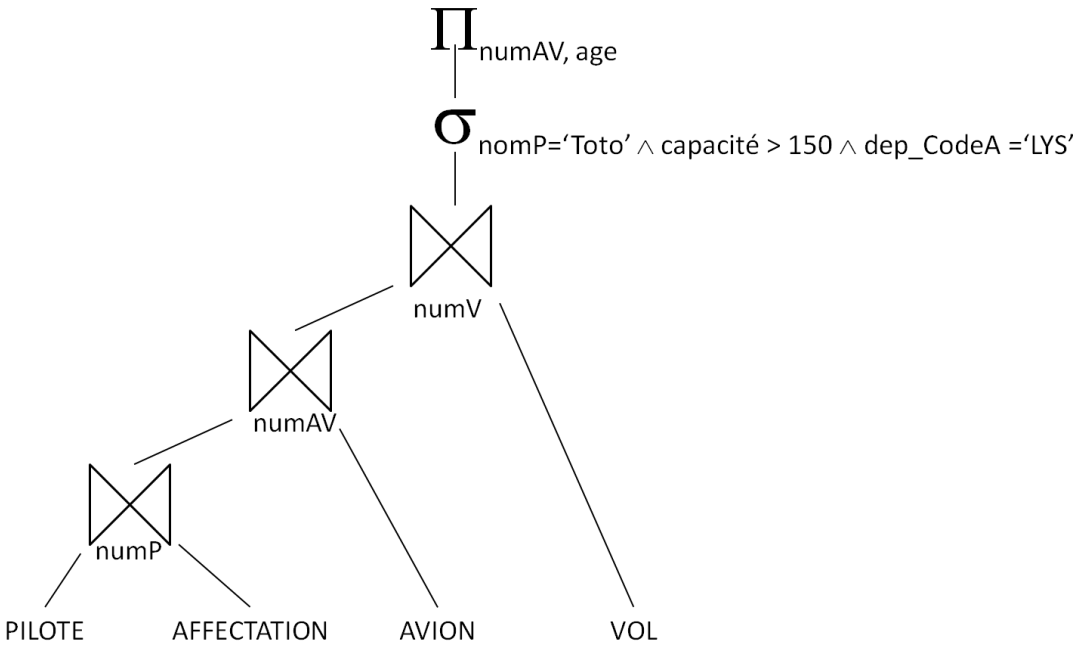
8) Donner le code des aéroports ayant géré plus de 500 fois le décollage d'Airbus A380.

9) Ajouter dans la base le pilote français Toto (numéro 123456) dont on ne connaît pas l'année d'obtention du brevet de pilote.

10) Supprimer la table affectation

Exercice 2: Optimisation de requêtes (3pts)

1) Exprimer en français ce que retourne la requête R1 dont un plan d'exécution possible est :



2) Proposer un plan d'exécution optimisé de R1

Partie B : Questions de cours (4pts)

Pour chacune des assertions suivantes, dire si elle est vraie ou fausse. Dans le cas où l'assertion des fausse, proposez une correction.

1) L'exécution sérielle (en série) de transactions génère potentiellement des conflits.

2) Pour vérifier si une politique de verrouillage génère des interblocages, je construis le graphe de précédence et je détermine s'il existe au moins un cycle dans ce graphe.

3) Pour vérifier si une exécution de transactions est sérialisable, je construis le graphe de précédence et je détermine s'il existe au moins un cycle dans ce graphe.

4) Deux actions sont conflictuelles sur un même granule, si au moins une des deux est une lecture.

5) L'attribution d'un verrou exclusif sur un granule ne permet d'attribuer que des verrous partagés supplémentaires sur ce même granule

6) Dans les propriétés ACID d'un modèle transactionnel le C signifie Concurrence.

7) La clé étrangère garantie l'unicité de la valeur d'un attribut

8) Les dépendances fonctionnelles sont utiles pour normaliser un schéma de bases de données.

Partie C : Conception de BD (5pts)

La société « LocInfo » est spécialisée dans la location de matériel informatique (PC, écran, portable…) aux particuliers et aux entreprises. Le matériel loué dispose d'un numéro unique, une année de fabrication et d'une désignation. Ce matériel est loué par des clients identifiés par un numéro de client et dont on connaît le nom, l'adresse, le code postal et la ville. S'il s'agit d'une entreprise le nom de l'entreprise est spécifié en plus. Un client passe une commande pour pouvoir louer du matériel. La commande est émise par un magasin. Un magasin est identifié par la ville où il se trouve, car il ne peut y avoir plus d'un magasin par ville. Chaque magasin dispose d'au moins un entrepôt, mais pas plus de quatre. Un entrepôt a un numéro qui est relatif au magasin. Le matériel est stocké dans un entrepôt avant d'être fourni au client.

En cas de panne, le matériel fourni est remplacé par un autre. Dans ce cas, on souhaite conserver la trace de ce remplacement. Le matériel défectueux est envoyé dans un centre spécialisé dans les réparations. Ce centre est constitué de plusieurs ateliers. Chaque atelier est identifié par un numéro et a une spécialité de réparation. La demande de réparation est effectuée par un magasin pour un matériel donné.

Exercice 1 : Schéma Entité-Association (3 pts)

Proposer un schéma Entité/Association permettant de modéliser la base de données de la société « LocInfo ». Vous veillerez à bien préciser les cardinalités.

Cf dernière page pour faire votre schéma

Exercice 2 : Modèle relationnel (2 pts)

1) A partir des Transformer votre schéma Entité/Association selon le modèle relationnel

2) Parmi les entités que vous avez définie, en choisir une, n'importe laquelle et donner la commande SQL permettant de l'implémenter dans un SGBD.

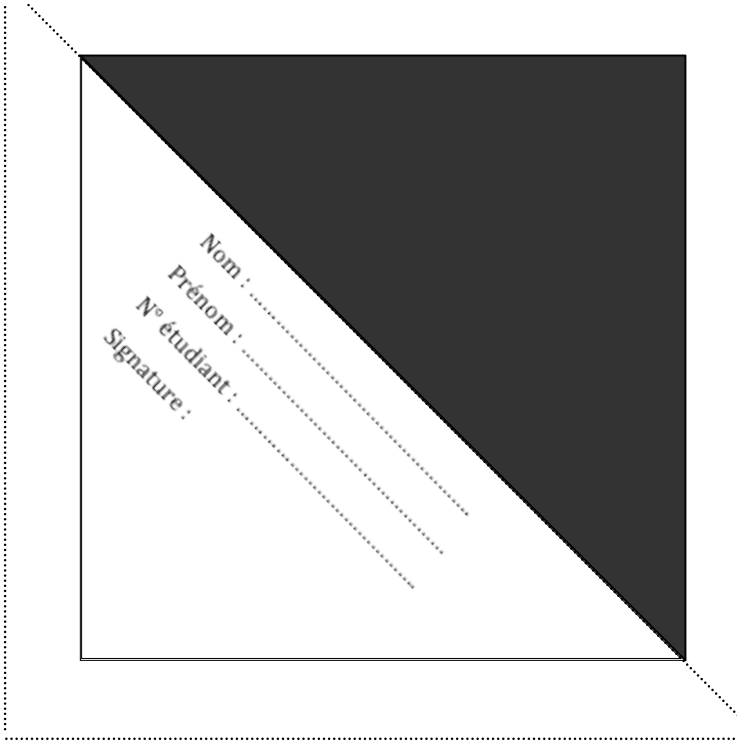
Pour le schéma Entité/Association, tourner la page svp ! ➔

Université Claude Bernard Lyon 1
Licence Sciences, Technologies, Santé – L2
Année 2010-2011, 2ème semestre

LIF5 – Algorithmique & Programmation procédurale

Contrôle final du 20 juin 2011

Durée : **1h30**



Note :

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

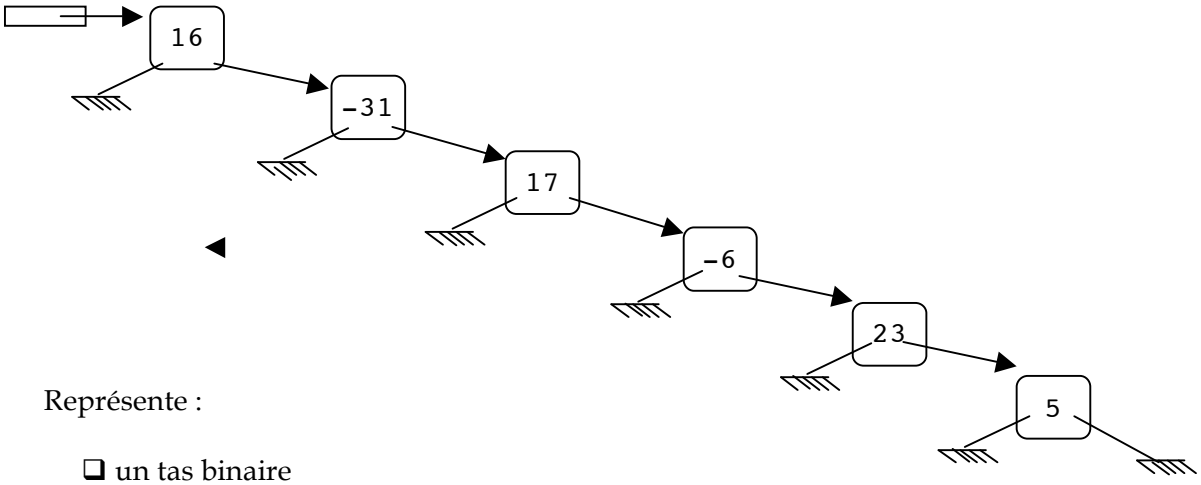
Le barème est donné à titre indicatif.

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. La représentation graphique suivante :



Représente :

- ❑ un tas binaire
- ❑ un arbre binaire
- ❑ un arbre binaire bien équilibré
- ❑ un arbre binaire de recherche
- ❑ une liste doublement chaînée

2. Cochez la ou les propositions correctes au sujet d'un tas binaire décroissant.
   - ❑ {7, 6, 5, 4, 3, 2, 1} est un tas binaire décroissant.
   - ❑ Tout tas binaire décroissant est trié dans l'ordre croissant.
   - ❑ Tout tas binaire décroissant est trié dans l'ordre décroissant.
   - ❑ Un tas binaire décroissant peut être stocké dans un tableau dynamique.
   - ❑ k éléments initialement dans un ordre quelconque peuvent être réorganisés en un tas binaire décroissant en $O(\log_2(k))$.

3. Parmi les structures de données suivantes, laquelle ou lesquelles permettent d'accéder en temps constant au i-ème élément, quel que soit i ?
   - ❑ tas binaire décroissant
   - ❑ liste simplement chaînée circulaire
   - ❑ liste doublement chaînée circulaire
   - ❑ arbre binaire de recherche
   - ❑ tableau statique

4. Cochez la ou les affirmations correctes au sujet d'une File d'Elements implantée sous forme de liste chaînée.
   - ❑ les Elements sont stockés dans le segment Tas de l'espace d'adressage
   - ❑ les Elements sont stockés de façon contiguë en mémoire vive
   - ❑ les Elements sont stockés dans un fichier binaire sur le disque dur
   - ❑ il faut mémoriser des pointeurs en plus des éléments eux-mêmes

5. Qu'appelle-t-on le « coût amorti » d'une insertion dans une structure de données dynamique ?
   - ❑ le nombre d'opérations élémentaires effectuées dans le pire des cas
   - ❑ le nombre d'opérations élémentaires effectuées dans le meilleur des cas
   - ❑ le nombre d'opérations élémentaires moyenné sur plusieurs insertions successives
   - ❑ le nombre d'opérations élémentaires exprimé relativement à celui pour une suppression

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

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

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

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

A. Ecrivez en C une fonction qui décroche la cellule de tête d'une liste chaînée bien initialisée, éventuellement vide et qui renvoie l'adresse de cette cellule. Il ne faut faire appel à aucune procédure ou fonction vue en cours, en td ou en tp.

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

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

   Résultat : .........................................................
*/

.................  decrocheTete(.........................)
/* complétez le type de retour de la fonction et la liste des paramètres */
{









}
```

B. Ecrire en C une procédure qui chaîne en tête d'une liste, éventuellement vide mais bien initialisée, une cellule déjà créée et bien initialisée dont l'adresse est passée en paramètre de la procédure. La liste est elle aussi passée en paramètre de la procédure.

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

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

*/

......... chaineTete(.............................................)
/* complétez la liste des paramètres */
{









}
```

C. Soit deux listes chaînées L1 et L2, triées dans l'ordre CROISSANT. Ecrivez en C une procédure qui fusionne ces deux listes pour obtenir une liste triée dans l'ordre DECROISSANT. Exemple, L1 = {5, 7, 9, 11} ; L2 = {6, 10, 15}. Le résultat obtenu est {15, 11, 10, 9, 7, 6, 5}. On interdit la création de nouvelles cellules, c'est-à-dire qu'à la fin de la procédure, les deux listes L1 et L2 sont vides.

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

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

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

*/
......... FusionSansCreation(.........................................)
/* complétez la liste des paramètres */
{









}
```

D. On suppose donnée une procédure dont le prototype est

```
void CreeListeTriee (Liste * l, int taille) ;
/* Précondition : la liste l est bien initialisée, taille est valide.
   Postcondition : construit aléatoirement l triée par ordre croissant avec
taille éléments */
```

Ecrivez le programme principal qui utilise cette procédure pour créer 2 listes triées par ordre croissant et qui les fusionne en appelant la procédure **`FusionSansCreation()`**. Toute procédure appelée doit être préalablement définie, à l'exception de CreeListeTriee.

```
.....main(.....................)
{









}
```

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. On considère un arbre binaire de recherche a non vide. Ecrire en C, la fonction qui renvoie l'adresse du Nœud contenant le minimum dans l'arbre a.

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

   Résultat.....................................................................

*/
.............. minimum(.....................................................)
/* paramètre(s) à compléter */
{









}
```

B. On considère un arbre binaire a non vide. Ecrire en C une fonction ITERATIVE qui recherche l'adresse du père d'un nœud de l'arbre a, contenant l'Elem e donné. On supposera que le nœud recherché n'est pas à la racine de a. 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 initialiserFile(File *pF);
/* Précondition : *pF non préalablement initialisée
   Postcondition : *pF initialisée en File vide */

void testamentFile(File *pF);
/* Précondition : *pF préalablement initialisée
   Postcondition : *pF prête à disparaître (ne doit plus être utilisée) */

void enfiler(File *pF, Noeud * adrNoeud);
/* Précondition : *pF initialisée
   Postcondition : adrNoeud est placée en fin de (*pF) */

void defiler(File *pF);
/* Précondition : *pF initialisée, *pF non vide
```

```
         Postcondition : le premier élément de *pF est retiré */

      Noeud * consulterPremier(File F);
      /* Précondition : F non vide
         Résultat : adresse située au début de F */

      int testFilevide(File F);
      /* Précondition : F initialisée
         Résultat : 1 si F est vide, 0 sinon */
```

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

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

*/
.................recherchePere(.................................)
/* paramètre(s) à compléter */
{









}
```