20 décembre 2004

UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS
DÉPARTEMENT D'INFORMATIQUE

STRUCTURES DES DONNÉES ET ALGORITHMES
INF4393

EXAMEN FINAL

Directives:

1. C'est un examen à livre ouvert.
2. Répondre directement sur ce formulaire.
3. L'examen dure trois heures.
4. Le barème est montré au début de chaque question.

Nom, prénom:_____

Code permanent:_____

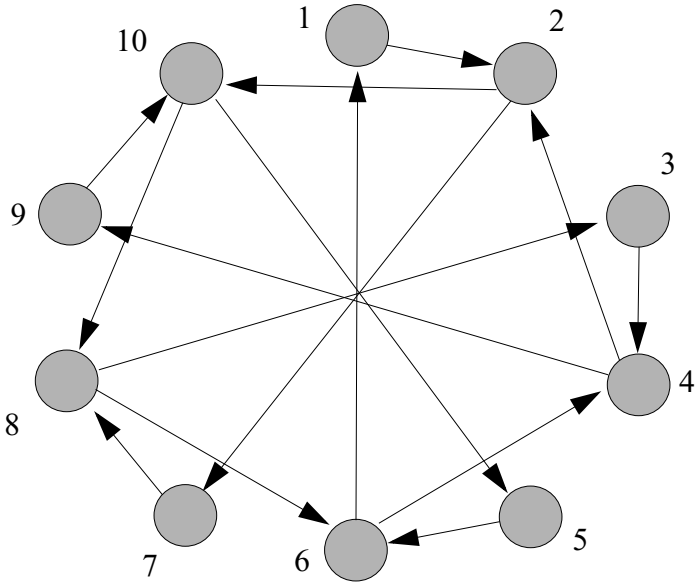
Signature:_____

Numéro de formulaire:_____

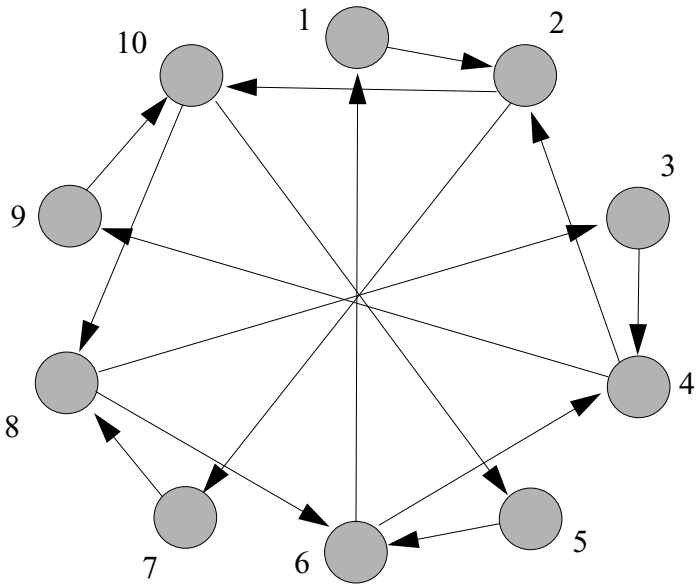
Ex. #	1	2	3	4	5	6	7	8	9	10	11	Σ
Note												
sur	10	10	8	9	9	10	11	7	9	6	11	100

Ex. #1 (10 pts)

a) (4 pts) Pour le graphe donné ci-dessous nous avons performé l'algorithme de parcours en largeur. Tous les sommets et tous les arcs ont été pris en consideration en ordre lexicographique. Marquer les arcs (en les rendant plus epais) qui appartiennent à l'arbre de prédécesseurs de cet algorithme. Pour chaque sommet marquer sa distance par rapport à la source (sommet 1).



a) (6 pts) Pour le graphe donné ci-dessous nous avons performé l'algorithme de parcours en profondeur. Tous les sommets et tous les arcs ont été pris en consideration en ordre lexicographique. Marquer les arcs qui appartiennent à l'arbre de prédécesseurs de cet algorithme. Pour les autres arcs marquer les arcs avants (A), transverses (T) et arcs arrières (R). Pour chaque sommet marquer son temps de debut et le temps de la fin de son traitement.



Ex. #2 (10 pts)

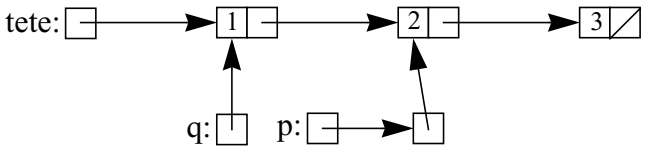
Dans un programme, on gère une liste. Pour cette fin, on a déclaré:

```
typedef struct noeud *ptr_noeud;

struct noeud
{
  int valeur;
  ptr_noeud suiv;
};

ptr_noeud tete, q, *p;
typedef ptr_noeud LISTE;
LISTE L;
```

La liste L contient les éléments avec les valeurs entières. Donner un fragment du programme qui supprime de la liste L tous les éléments conténant les valeurs paires.
\

Ex. # 3 (8 pts)

Dans un programme qui gère des listes chaînées, on suppose les mêmes déclarations que celles dans l'ex. #2. À un moment donné on a construit une liste présentée ci-dessous:



Quelle est la valeur de chacune des expressions suivantes (si vous jugez que l'expression est incorrecte, expliquer pourquoi):

a) `(*(q->valeur)).suiv->valeur`

b) `(**p).valeur`

c) `*q.suiv->valeur`

d) `(*p)->suiv->valeur`

e) `(*q->suiv).suiv->valeur`

f) `&*q = *p`

g) `(**q)->suiv->valeur`

h) `&**p == q->suiv`

Ex. #4 (9 pts)

Combien d'arbres AVL différents peut-on former à partir d'un ensemble de N clés pour

a) (2.5 pts) N=4?

b) (3 pts) N=5?

c) (3.5 pts) N=6?

Ex. #5 (9 pts)

Pour stocker une suite de clés: 73, 77, 15, 17, 50, , (arrivées dans cette ordre) nous avons à notre disposition le tableau à m=7 éléments. La fonction de hachage primaire H est définie, comme modulo m.

a) (4 pts) Donner les valeurs de la fonction de hachage H pour les clés données ci-dessus:

H (73) =
H (77) =
H (15) =
H (17) =
H (50) =

b) (5 pts) Pour traiter des collisions obtenues dans l'ex #5a), on applique la méthode linéaire avec l'incrément fixe égal à 4. Dans quelles positions du tableau sont installées les clés données (dans l'ordre indiqué dans l'ex. #5a) après la résolution des collisions éventuelles?

ADR	CLÉ	LG
0		
1		
2		
3		
4		
5		
6		

c) (4 pts) Dans le tableau donné, remplir la colonne LG, en donnant pour chaque clé de l'ex. #5a), le nombre de positions consultées au cours de la recherche de cette clé dans le tableau.

Ex. #6 (10 pts)

Donner la sortie du programme suivant:

```
#include <stdio.h>

int F (int **x);
int G (int *y);

main()
{
  int a, b, *c;

  a = 2, b = 3, c = &b;
  *c = a + 2;
  b = G(&a) + 3;
  printf ("apres appel de G: a = %d b= %d\n", a, b);
  a = 2, b = 3, c = &b;
  a = F(&c) + 4;
  printf ("apres appel de F: a = %d b= %d\n", a, b);
}

int  F (int **x)
{
  int z = 3;
  printf ("au debut de F:**x = %d\n", **x);
  **x += z + 3;
  z = **x + 4;
  printf ("a la fin de F: **x = %d, z = %d\n", **x, z);
  return (**x + 1);
}

int G (int *y)
{
  int x = 3;
  printf ("au debut de G: *y = %d\n", *y);
  x = *y + 5;
  *y = x + 2;
  printf ("a la fin de G: x = %d *y = %d\n", x, *y);
  return (x + *y);
}
```

Ex. #7 (11 pts)

Pour chaque expression qui sui, donnée en notation postfixée vous devez

1. Donner son arbre d'expression
2. Donner la forme préfixée
3. Donner la forme infixée utilisant seulement les parenthèses absolument nécessaires

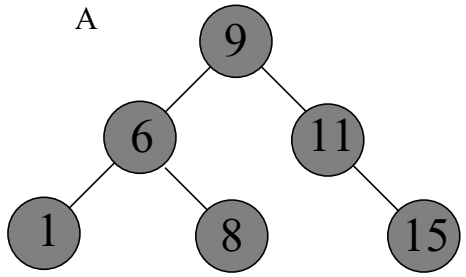
a)(3 pts) `a b c * d - + e f g h / + - *`
(tous les opérateurs sont binaires)

b)(4 pts) `a ++ b c || d -- e * + &&`
(opérateurs du langage C, ++ et -- sont alors unaires)

c)(4 pts) `a b c d ?: e f + ++ ?: g h / -`
(opérateurs du langage C, ?: - opérateur conditionnel)

Ex. #8 (7 pts)

Soit A un arbre binaire de recherches suivant:



Supposons que pour l'arbre A on a performé une séquence de requêtes suivantes:

Ajouter (7);
Ajouter (10);
Supprimer (8);
Ajouter (3);
Supprimer (9);
Ajouter (6);

Donner une suite de 5 arbres qui sont les résultats après chacune de ces requêtes (l'objet de chaque requête est l'arbre obtenu à la suite des requêtes précédentes).

Ex. #9 (9 pts)

Pour le graphe G donné ci-dessous on a appliqué l'algorithme de Bellman-Ford, en traitant le sommet 4 comme le sommet source. Dans chaque itération de la boucle de la ligne 2, l'ordre de relâchement d'arètes est lexicographique, c'est-à-dire si l'arète (x, y) a était relâché avant (a, b) alors x < a ou bien x = a et y < b. À la fin de chaque phase (chaque itération de la boucle de la ligne 2) donner l'état de votre graphe G, en marquant le graphe de prédécesseurs ainsi que la valeur de d(v) pour chaque sommet v.

Ex. #10 (6 pts)

Supposons que maxN est la forme générique de l'opérateur de arité N

`maxN (`e_1`, `e_2`, . . . , `e_N`)`

qui calcule la valeur maximale parmi les expressions e_1, e_2, . . . , e_N. Dans l'expression donnée ci-dessous on aura les apparitions de l'opérateurs max2 (binaire), max3 (ternaire), max4 (quaternaire - arité 4) et max5 (arité 5). Pour l'expression en notation infixée

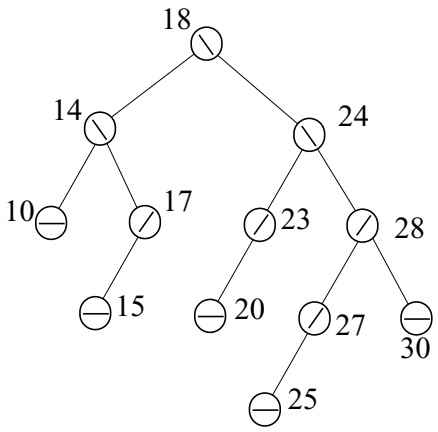
```
max3( max4( 2, max2(3, 4), max3( 5, 6, 7), 8), max2(9, 10), 11)
```

a) (3 pts) Donner sa forme postfixée:

b) (3 pts) Donner sa forme préfixée:

Ex. #11 (11 pts)

Soit A un arbre binaire AVL suivant:



a) (1 pt) Dessiner un arbre qui est le résultat de l'insertion dans l'arbre A de la clé 12:

b) (1.5 pts) Dessiner un arbre qui est le résultat de l'insertion dans l'arbre A de la clé 21:

c) (1.5 pts) Dessiner un arbre qui est le résultat de l'insertion dans l'arbre A de la clé 26:

d) (1 pt) Dessiner un arbre qui est le résultat de la suppression dans l'arbre A de la clé 15:

e) (1.5 pts) Dessiner un arbre qui est le résultat de la suppression dans l'arbre A de la clé 17:

f) (1.5 pts) Dessiner un arbre qui est le résultat de la suppression dans l'arbre A de la clé 10:

g) (1.5 pts) Dessiner un arbre qui est le résultat de la suppression dans l'arbre A de la clé 20:

h) (1.5 pts) Dessiner un arbre qui est le résultat de la suppression dans l'arbre A de la clé 28: