10 mars 2004

UNIVERSITÉ DU QUÉBEC EN OUTAOUAIS
DÉPARTEMENT D'INFORMATIQUE

STRUCTURES DES INFORMATIONS I
INF4063

EXAMEN INTRA

Directives:

1. C'est un examen à livre ouvert, mais les calculatrices sont interdites.
2. Répondre directement sur ce formulaire.
3. Les questions éventuelles doivent être posées durant les premières 30 minutes.
4. L'examen dure trois heures.
5. 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	Σ
Note									
sur	16	12	10	14	10	14	16	12	100

Ex. #1 (16 pts)

Soit un programme contenant un appel d'une fonction récursive tri_rapide:

```
#include <stdio.h>
#define MAX 9

echanger (int a[ ], int i, int j)
{   int temp;

    temp = a[i]; a[i] = a[j]; a[j] = temp;
}

afficher_tableau (int a[ ], int nbre)
{   int i;

    for (i = 1; i <= nbre; i++)
        printf ("%d  ", a[i]);
    printf ("\n");
}

tri_rapide (int a[ ], int gauche, int droite)
{   int val, i, j;

    if (droite>gauche) {
        val = a[droite]; i = gauche-1; j = droite;
        for ( ; ; ) {

            while (a[++i] < val) ;
            while (a[--j] > val) ;
            if (i>=j) break;
            echanger (a, i, j);
        }
        echanger ( a, i, droite);
        afficher_tableau (a, MAX-1);
        tri_rapide (a, gauche, i-1);
        tri_rapide (a, i+1, droite);
    }
}

main ()
{   int i, tab [MAX] = {0, 7, 6, 1, 4, 2, 3, 8, 5};

    tri_rapide (tab, 1, MAX-1);
}
```

a) (9 pts) Donner l'arbre de récursivité de l'appel principal **tri_rapide (tab, 1, MAX-1)**.

b) (7 pts) Qu'est-ce qui est affiché par ce programme?

Ex. #2 (12 pts)

Répondre par **Oui** ou **Non**:

a) O(n) = 3n-1 ?

b) O(n/2+1) = n ?

c) O(n log n) = n ?

d) O(n) = n log n ?

e) Θ(n) = 3n-1 ?

f) Θ(n/2+1) = n ?

g) $\Theta(2^{\log n}) = 2n$?

h) $\Theta(2n) = 2^{\log n}$?

Ex. #3 (10 pts)

Soit une suite suivante de déclarations en langage C:

```
typedef struct{
    int age;
    char nom [16];
    double salaire;
} employe;

typedef struct{
    char firme [20];
    int nbre_employes;
    employe [30];
} compagnie;

compagnie PetiteBoite;
```

Supposons que
sizeof int = 4,
sizeof double = 8, et
sizeof char = 1
et que la variable PetiteBoite ait été allouée à partir de l'adresse de base = 1000.

a) Quelle est la taille de la zone mémoire occupée par la variable PetiteBoite?

b) Quelle est la taille de la zone mémoire occupée par la donnée PetiteBoite.employe[7]?

c) Quelle est l'adresse de la donnée PetiteBoite.employe[7].nom[10]?

d) Quelle est l'adresse de la donnée inf4063.inscrit[3].matricule[4]?]

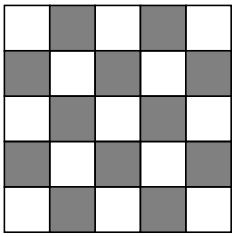
e) Quelle composante de donnée occupe la case mémoire à l'adresse 1746?]

Ex. #4 (14 pts)

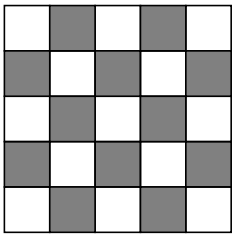
Soit un programme de *Cinq reines* qui affiche toutes les configurations de cinq reines qui ne s'attaquent pas sur un échiquier de 5x5.

```
int Col[5];                         /* colonne contenant la reine*/
bool ColLibre[5]                    /* est-ce la colonne libre?*/
bool DiagMontLibre[11]              /* est-ce la diagonale montante libre?*/
bool DiagDescLibre[11]              /* est-ce la diag. descendante libre?*/

int ligne = -1;                     /* dans cette ligne on place la reine*/

void main (void);
{
    int i;

    for (i=0; i<5; i++)
        ColLibre[i] = VRAI;
    for (i=0; i<11; i++) {
        DiagMontLibre[i] = VRAI;
        DiagDescLibre[i] = VRAI;
    }
    AjouterReine();
}

/* AjouterReine: essaye de placer les reines, en retournant si nécessaire */
void AjouterReine(void);
{
    int c;                                          /* on va essayer de placer*/
                                                    /* une reine dans cette colonne*/
    ligne++;
    for (c=0; c<5; c++)
        if (ColLibre[c] && DiagMontLibre[ligne+c] &&
                DiagDescLibre[ligne-c+4]) {
            Col[ligne] = c;
            ColLibre[c]= FAUX;
            DiagMontLibre[ligne+c] = FAUX;
            DiagDescLibre[ligne-c+4] = FAUX;
            if (ligne==4)                           /* condition de terminaison*/
                AfficherEchiquier();
            else
                AjouterReine();                     /* continuer récursivement*/
            ColLibre[c]= VRAI;
            DiagMontLibre[ligne+c] = VRAI;
            DiagDescLibre[ligne-c+4] = VRAI;
        }
    ligne--;
}
```

a) (6 pts) Donner la première configuration affichée par le programme *Cinq reines* en cochant les cases appropriées:



b) (8 pts) Donner la dernière configuration affichée par le programme *Cinq reines*:



Ex. #5 (10 pts)

Soit la déclaration en langage C:

int Tab [3] [4] [5];

Supposons que sizeof int = 4 (c'est-à-dire que chaque élément entier de ce tableau occupe 4 octets), et que le tableau a été alloué à partir de l'adresse de base = 1000.

a) Quelle est la taille de la zone mémoire occupée par le tableau Tab?

b) Quelle est la taille de la zone mémoire occupée par l'élément Tab [2] de ce tableau?

c) Quelle est l'adresse de l'élément Tab [1, 2, 4] de ce tableau?

d) Quelle est l'adresse de l'élément Tab [2, 3] de ce tableau?]

e) Quel élément du tableau Tab occupe la case mémoire à l'adresse 1135?]

Ex. #6 (14 pts)

Deux joueurs participent au “Jeu de 25”. Le premier joueur (joueur A) choisit un nombre parmi 5 et 8. Ensuite c’est le tour du second joueur (joueur B), qui aussi choisit un nombre parmi 5 et 8. Après ça, à nouveau c’est le joueur A qui doit choisir un nombre parmi 5 et 8, etc. Le jeu se termine quand la somme des nombres choisis par les deux joueurs dépasse la valeur de 25. Le joueur qui a choisi le dernier nombre gagne.

a) Donner l’arbre de jeu pour le “Jeu de 25”. Les arêtes de cet arbre doivent être etiquetées par les nombres choisis par chacun des deux joueurs.
b) Ensuite associez les valeurs binaires aux sommets terminaux de votre arbre de sorte que ça corresponde aux gains de chacun des joueurs (1 - joueur A gagne, 0 - joueur B gagne).
c) Appliquer la méthode de minimax pour déterminer lesquel des deux joueurs a une stratégie gagnante.

Ex. #7 (16 pts)

Soit un programme contenant un appel d'une fonction récursive truc:

```
int truc (int n)
{int i, j;
    if ((n==1) || (n==2))
            return(n);
    else {
            printf ("avant %d\n", n);
            i = truc ((n+1)/2);
            printf ("entre %d\n", n);
            j = truc (n-2);
            printf ("apres %d\n", n);
            return( i + j);
    }
}
 main ()
{
    printf("truc(9) = %d\n", truc (9));
}
```

a) (7 pts) Donner l'arbre de récursivité de l'appel truc(9).

b) (3 pts) Donner la valeur retournée par l'appel truc(9) du programme principal.

c) (6 pts) Donner la sortie affichée par ce programme.

Ex. #8 (12 pts)

Supposons que le programme main contient l'appel de truc(16) à la place de truc(9)

a) (4 pts) Donner la valeur retournée par l'appel truc(16) du programme principal.

b) (4 pts) Donner le nombre maximal Max d'appels de la fonction truc qui peuvent se trouver sur la pile d'appels de fonctions (c-à-d la plus grande profondeur de la pile). d'appels de la fonction truc.

c) (4 pts) Combien de fois la valeur de Max de l'ex. #8 b) peut être atteinte durant l'exécution de truc (16)?