Université d'Ottawa
Faculté de génie

School of Electrical Engineering and Computer Science

uOttawa
L'Université canadienne
Canada's university

University of Ottawa
Faculty of Engineering

École de science informatique et de génie électrique

CSI2520 Paradigmes de programmation
EXAMEN FINAL

Durée: 3 heures **Avril 2018**

Professeur: Robert Laganière

Nom: _____

Prénom: _____

Numéro d'étudiant: _____

Signature _____

Une feuille de notes premise (recto-verso).

Question	Points	sur
1		6
2		5
3		3
4		5
5		4
6		5
7		5
8		5
Total		38

Question 1 Base de données [6 points]

Voici une base de données en format Prolog:

```
% name, game, score
score( 'Emma', 'FIFA18', 3 ).
score( 'Benjamin', 'Minecraft', 387 ).
score( 'Liam',  'The Legend of Zelda', 2200 ).
score( 'Ethan', 'Super Mario Odyssey', 15100 ).
score( 'Ava', 'Minecraft', 410 ).
score( 'Liam', 'Minecraft', 222 ).
score( 'Ava',  'The Legend of Zelda', 1900 ).
```

a) En utilisant `setof/3` effectuer une requête permettant d'obtenir la liste triée de tous les scores obtenus dans le jeu Minecraft

```
?- _____________________________________________________________________

   _____________________________________________________________________ .

L = [222, 387, 410].
```

b) En utilisant `findall/3` effectuer une requête permettant d'obtenir la liste des jeux joués par Liam

```
?- _____________________________________________________________________

   _____________________________________________________________________ .

L = ['The Legend of Zelda', 'Minecraft'].
```

c) Compléter le prédicat `countGames` permettant d'obtenir le nombre d'occurrences d'un jeu donné dans une liste de jeux.

```
?- countGames( ['FIFA18', 'Minecraft', 'The Legend of Zelda',
                'Super Mario Odyssey', 'Minecraft', 'Minecraft',
                'The Legend of Zelda'],'Minecraft', N).
```

N = 3.

```
countGames( [], _, 0 ) :-_____ .

countGames( [O|T], G, C ) :- _____

_____

_____ .

countGames( [G|T], G, C ):- _____

_____

_____ .
```

d) Soit les prédicats suivants?

```
popular( P ) :- findall( G, highscore(_,G,_),L),
                setof( G, N^S^highscore(N,G,S),LL),
                findMax( L, LL, P, _ ).


max( MG, MC, _, CC, MG, MC ) :- MC>CC, !.

max( _, _, G, C, G, C ) :- !.


findMax( _, [], 'None', 0 ) :- !.

findMax( L, [G|OG], M, C ) :- findMax( L, OG, MG, MC ),
                              countGames( L, G, CC ),
                              max( MG, MC, G, CC, M, C ).
```

Que sera produit par la requête suivante?

```
?- popular(P).
```

_____

_____ .

Question 2 Traitement de listes en Sheme [5 points]

Soit l'exemple suivant:

```
(contain '(a 7 9 10) '(5 a c 7 10))
⇨ ((a 7 10) (9))
```

Compléter la procédure auxiliaire `containAux` pour `contain` acceptant deux listes en entrée et produisant une liste de deux listes en sortie.

La première des listes en entrée est faite d'éléments à tester afin de savoir si ils se trouvent dans la liste de recherche (la deuxième liste donnée en entrée).

En sortie, la première liste contient les éléments contenus dans la liste de recherche et la seconde contient les éléments ne se trouvant pas dans la liste de recherche.

```
(define (contain S L)
      (cond
       ((or (not (list? S)) (not (list? L))) error)
       ((or (null? S) (null? L) ) (list '() '()))
       (#t (containAux S L '() '()))))


(define (containAux S L FS FO)

      (cond

        ((null? S) _____________________________________________ )

        ((member (car S) L) _____________________________________________

        _____________________________________________

        _____________________________________________)

        (#t   _____________________________________________

        _____________________________________________

        _____________________________________________)))
```

Question 3 Calcul en Scheme [3 points]

L'aire d'un rectangle spécifié par la coordonnée de son coin supérieur gauche A et celle de son coin inférieur droit B est donnée par :

$area = (B_x - A_x)\,(B_y - A_y)$

Les coins A, B sont représentés par des pairs Scheme, e.g., `(-2 . 2) (1 . 5)`

L'appel à la function pourrait donc être comme suit:

```
(area (cons -2 2) (cons 1 5))
⇨ 9
```

Compléter la fonction `area` effectuant le calcul de l'aire:

```
(define (area A B)
```

_____

_____

_____

_____

_____

_____

_____

_____)

Question 4 let, let*, letrec, named let [5 points]

Soit la fonction suivante:

```
(define (fct x y)
   (let ((diff (- x y)))
      (let ((diff-squared (* diff diff)))
         (let ((diff-cubed (* diff-squared diff)))
            diff-cubed))))
```

a) Quel sera le résultat de l'appel suivant?

```
(fct 5 7)
```

_____

b) Ré-écrire cette fonction en remplaçant les appels à `let` par un appel à la function `let*`

```
(define (fct x y)
```

_____

_____

_____

_____

_____

_____)

Soit la fonction suivante:

```
(define (abc x y)
 (let loop ((i x) (s 0))
   (if (< i y)
       (loop (+ i 1) (+ s i)) s)))
```

c) Quel sera le résultat de l'appel suivant?

```
 (abc 3 7)
```

Soit la fonction suivante calculant la longueur d'une liste:

```
(define (len L)
  (if (list? L)
     (lengthAux L)
     'error ))

(define (lengthAux L)
  (if (null? L)
      0
      (+ 1 (lengthAux (cdr L)))))
```

d) Ré-écrire cette fonction sans utiliser une fonction auxiliaire (`lengthAux`) mais en utilisant plutôt le `letrec`:

```
(define (len L)
  (if (list? L)
      (letrec
(_____________________________________________________________

_____________________________________________________________

_____________________________________________________________

_____________________________________________________________

_____________________________________________________________

)
  (lenlet L))
     'error))
```

Question 5 Go Routines [4 points]

Soit la fonction `main` ci-dessous faisant appel à la fonction `fourier` en utilisant des go routines.

```
package main

import "fmt"
import "runtime"
import "math"
import "math/rand"

type Series struct {
     a, b float64
}

func main() {
     runtime.GOMAXPROCS(3)

     data := make(chan float64)
     defer close(data)
     var c [32]Series
     TP := 4

     for t := 0; t < TP; t++ {
          for k := 0; k < 32; k++ {
               c[k].a = rand.Float64()/32.0
               c[k].b = rand.Float64()/32.0
          }
          go fourier(c, t, TP, data)
     }
     // Below the results from all of the go routines need to be
     // received and printed to the console. The program is to exit
     // when all data is received.

     _____________________________________________________________

     _____________________________________________________________

     _____________________________________________________________

     _____________________________________________________________

     _____________________________________________________________
}
```

La fonction `fourier` accepte des tableaux de taille 32.

```
func fourier(c [32]Series, t, TP int, out chan float64) {
     res := c[0].a
     for n := 1; n < 32; n++ {
          res += c[n].a*math.Sin(2.0*math.Pi*float64(t)/float64(TP))
               + c[n].b*math.Cos(2.0*math.Pi*float64(t)/float64(TP))
     }
     out <- res
     return
}
```

Modifier cette fonction afin qu'elle puisse accepter des slices de tailles diverses.

```
func fourier(c ____________________________________________________,

              t, TP int, out chan float64) {

     res := c[0].a



     for ____________________________________________________________{

          res += c[n].a*math.Sin(2.0*math.Pi*float64(t)/float64(TP))
               + c[n].b*math.Cos(2.0*math.Pi*float64(t)/float64(TP))
     }
     out <- res
     return
}
```

Question 6 Arbre Prolog [5 marks]

Voici la définition d'un arbre ayant plusieurs enfants et dont chaque nœud contient la coordonnée d'un point:

```
t(2,-3,[t(5,1,[t(7,2,[])]),
        t(-3,4,[]),
        t(2,4,[t(-1,1,[]),
        t(-2,3,[])])]
  )
```

Le parcours in-ordre se fait comme suit:

```
traverse(t(X,Y,[])) :- write(X), tab(1), write(Y), nl.

traverse(t(X,Y,[H|T])) :- traverse(H),
    write(X), tab(1), write(Y), nl,
    traverseTail(T).

traverseTail([]) :- !.

traverseTail([H|T]) :- traverse(H),
    traverseTail(T).
```

Voir page suivante!

a) Que sera affiché par le parcours suivant?

```
?- traverse(t(2,-3,
                    [t(5,1,[t(7,2,[])]),
                     t(-3,4,[]),
                     t(2,4,[t(-1,1,[]),
                            t(-2,3,[])])]
                   ).
```

_____

_____

_____

_____

_____

_____

_____

_____

Le prédicat `findPoint` retourne vrai si un point est dans l'arbre.

```
findPoint(t(U,V,_),U,V) :- !.

findPoint(t(_,_,[H|_]),U,V) :- findPoint(H, U, V).

findPoint(t(_,_,[_|T]),U,V) :- findPointTail(T,U,V).
```

b) Compléter le précidat auxiliaire `findPointTail`:

```
findPointTail([H|_],U,V) :-
```

_____

_____ .

```
findPointTail([_|T],U,V) :-
```

_____

_____ .

Question 7 Arbre Scheme [5 points]

Voici la définition d'un arbre ayant plusieurs enfants et dont chaque nœud contient la coordonnée d'un point:

```
(define pTree (list (cons 2 -3)
                    (list (list (cons 5 1) (list (list (cons 7 2) '())))
                          (list (cons -3 4) '())
                          (list (cons 2 4) (list (list (cons -1 1) '())
                                                 (list (cons -2 3) '()))))))
```

Le parcours de l'arbre se fait comme suit:

```
(define (traverse T)
  (cond
    ((null? T) '())
    ((null? (cdr T)) (car T))
    (#t (cons (car T) (traverseTail (cadr T))))))

(define (traverseTail T)
  (if (null? T)
    '()
    (append (traverse (car T)) (traverseTail (cdr T)))))
```

a) Quel sera le résultat de l'appel suivant ?

```
(traverse pTree)
```

⇨ _____

_____

_____

_____

La fonction `findPoint?` retourne vrai si un point est dans l'arbre.

```
(define (findPoint? T U V)
  (cond
    ((null? T) #f)
    ((and (equal? (caar T) U) (equal? (cdar T) V)) #t)
    ((null? (cdr T)) #f)
    (#t (findPointTail? (cadr T) U V))))
```

a) Compléter le précidat auxiliaire `findPointTail?`:

```
(define (findPointTail? TL U V)
 (if (null? TL)
```

_____

_____

_____

_____

_____))

Question 8 Arbre Go [9 points]

Soit le programme suivant:

```
package main

import "fmt"

type nTree struct {
     x, y     int
     children []nTree
}

func main() {
     tree := nTree{2, -3,
          []nTree{{5, 1, []nTree{{7, 2, nil}}},
               {-3, 4, nil},
               {2, 4, []nTree{{-1, 1, nil},
                    {-2, 3, nil}}}}}

     tree.traverse()
     u, v := -1, 1
     if tree.findPoint(u, v) {
          fmt.Printf("Found: %d %d \n", u, v)
     }
}
```

a) Compléter la méthode `traverse` de facon à effectuer un parcours in-ordre de l'arbre et en imprimant les points à la console lorsqu'ils sont visités.

```
func (t *nTree) traverse() {
     if t.children == nil || len(t.children) == 0 {
          fmt.Printf("%d %d \n", t.x, t.y)
     } else {


          for _____________________________________________ {

           ____________________________________________________________

           ____________________________________________________________

           ____________________________________________________________

           ____________________________________________________________

           ____________________________________________________________

          }
     }
     return
}
```

b) Compléter la méthode findPoint ci-dessous retournant vrai si ce point se trouve dans l'arbre:

```
func (t *nTree) findPoint(u, v int) bool {
     if t.x == u && t.y == v {
          return true
     }
     if t.children == nil {
          return false
     }

     _____________________________________________________________

     _____________________________________________________________

     _____________________________________________________________

     _____________________________________________________________

     _____________________________________________________________

     _____________________________________________________________

     _____________________________________________________
}
```