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

CSI2120 Programming Paradigms

FINAL EXAM

Length of Examination: 3 hrs **April 10, 2014, 14:00-17:00**

Professor: Jochen Lang **Page 1 of 16**

Family Name: _____

Other Names: _____

Student Number: _____

Signature _____

You are allowed one double-sided letter-sized sheet of notes.

This exam will be evaluated out of 40. It contains two bonus points.

At the end of the exam, when time is up: Stop working and close your exam booklet. Remain silent.

Question	Marks	Out of
1		4
2		2
3		2
4		4
5		4
6		4
7		3
8		6
9		3
10		6
11		4
Total		42

Question 1: Prolog BST [4 marks]

a) Draw the tree corresponding to the following Prolog tree representation.

```
t(58,
  t(31,
    t(16,
      t(5,
     t(2,nil,nil),
     nil),
      t(21,
     t(18,
       nil,
       t(19,nil,nil)),
     nil)),
    nil),
  t(67,
    t(63,
      nil,
     t(65,nil,nil)),
    nil))
```

b) Which of the predicates below works correctly? The predicate is to find a key in a binary search tree. For example:

```
?- binarySearch(83,t(73, t(31, t(5, nil, nil), nil), t(101, t(83, t(97, nil, nil), nil), nil))).
      true
```

~~a)~~

```
binarySearch(K, t(K, _, _)).
binarySearch(K, t(R, S, _)) :-
    precedes(K, R),
    binarySearch(S, K).
binarySearch(K, t(R, _, S)) :-
    precedes(R, K),
    binarySearch(S, K).
```

b)

```
binarySearch(K, t(K, _, _)).
binarySearch(K, t(R, S, _)) :-
    precedes(K, R),
    binarySearch(K, S).
binarySearch(K, t(R, _, S)) :-
    precedes(R, K),
    binarySearch(K, S).
```

~~c)~~

```
binarySearch(K, t(_, K, _)).
binarySearch(K, t(_, _, K)).
binarySearch(K, t(R, S, _)) :-
    precedes(K, R),
    binarySearch(K, S).
binarySearch(K, t(R, _, S)) :-
    precedes(R, K),
    binarySearch(K, S).
```

~~d)~~

```
binarySearch(K, t(K, _, _)).
binarySearch(K, t(R, _, S)) :-
    precedes(K, R),
    binarySearch(K, S).
binarySearch(K, t(R, S, _)) :-
    precedes(R, K),
    binarySearch(K, S).
```

Question 2 Prolog Maze [2 marks]

Given the following maze program:

```
link(0,1).
link(1,2).
link(1,5).
link(2,3).
link(2,6).
link(3,7).
link(4,5).
link(4,8).
link(5,6).
link(6,7).
link(7,11).
link(8,9).
link(9,10).
link(10,11).

successor(A,B) :- link(A,B).
successor(A,B) :- link(B,A).

finish(11).

pathFinder([Last|Path],[Last|Path]) :-
      finish(Last).
pathFinder([Curr|Path],Solution) :-
      successor(Curr,Next),
      \+member(Next,Path),
      pathFinder([Next,Curr|Path],Solution).postIt([]).
```

What is printed by the following call?

```
?- pathFinder([0],X).
```

Question 3 Prolog Database [2 marks]

Complete the predicate insertUserId below such that a new user Id can be added to the database even if multiple users with the same last name need to be entered.

```
?- createUserId(name(smith,[joe,k])).
true
?- setof((X,Y),userId(X,Y),L).
L = [ (name(smith, [jane, m]), smith2), (name(smith, [joe, k]),
smith3), (name(smith, [tony, a]), smith1)].
```

```
:- dynamic userId/2.

userId(name(smith,[tony,a]), smith1 ).
userId(name(smith,[jane,m]), smith2 ).

% atomic_concat(+Atomic1, +Atomic2, -Atom)
%     Atom represents the text after converting Atomic1 and Atomic2 to
%     text and concatenating the result:
%
%     ?- atomic_concat(name, 42, X).
%     X = name42.

createUserId( name(LastName,FirstNames) ) :-
    insertUserId( name(LastName,FirstNames), 1 ).



insertUserId( name(LastName,FirstNames), N ) :-
    atomic_concat(LastName,N,Id),



    ____________________________________________________



    ____________________________________________________.







insertUserId( name(LastName,FirstNames), N ) :-
    M is N+1,
    insertUserId( name(LastName,FirstNames), M ).
```

Question 4 Scheme Let Statements [4 marks]

What is returned by the calls below?

```
(let ((x 11))
  (* 2 x))
```

```
 =>
```

```
(let ((x 1))
  (let ((x (* x 2)))
    (* x x)))
```

```
=>
```

```
(let ((x 1) (y 2))
  (let* ((x 3)
         (z (+ x y)))
    (* z x)))
```

```
=>
```

```
(let foo ((x 1))
  (if (< x 5)
      (+ x (foo (+ x 1)))
      x))
```

```
=>
```

Question 5 Scheme Lists [4 marks]

Complete the following function calls with a single function.

Example

```
(define L '(1 2))
(cadr L)
=> 2
```

```
(define L '((a)))
```

(_____ L)

=> a

```
(define L '(a b c))
```

(_____ L)

=> b

```
(define L '(a (b c) d))
```

=>
(_____ L)

=> d

```
(define L '(2 (3 (4 () (6 () (7 () ()))))))
```

(_____ L)

=> 3

Question 6 Scheme Queue [4 marks]

The following functions implement a stack in Scheme.

```
(define a-stack '())

(define (empty?)
  (null? a-stack))

(define (push e)
  (set! a-stack (cons e a-stack)))

(define (pop)
  (if (empty?)
      ()
      (set! a-stack (cdr a-stack))))

(define (top)
  (if (empty?)
      ()
      (car a-stack)))
```

Complete the corresponding definitions of a queue on the **next** page.

Hint: `procedure: (append list1 ... listn)`
`returns: the concatenation of the input lists`

Question 6 Scheme Queue (continued)

```
(define a-queue '())

(define (empty?)
  (null? a-queue))

(define (enque e)

    _____________________________________________________

(define (deque)

    _____________________________________________________

    _____________________________________________________

    _____________________________________________________


(define (top)
  (if (empty?)
      ()
      (car a-queue)))
```

Question 7 Scheme Vector-Product [3 marks]

The following function calculates the vector product by looping from the end of the vector to the beginning.

```
(define vector-product
  (lambda (vec)
    (do
      ((remaining (vector-length vec) (- remaining 1))
       (total 1 (* total (vector-ref vec (- remaining 1)))))
      ((zero? remaining) total)
)))

(vector-product #( 2 5 7))
=> 70
```

Redefine vector-product to loop forwards over the vector.

```
(define vector-product
  (lambda (vec)
    (do

   ____________________________________________________

    ____________________________________________________

    ____________________________________________________

)))
```

Question 8 Scheme BST [6 marks]

The function removemax-BST removes the maximum element from a binary search tree.

```
(define removemax-BST
  (lambda (t)
    (cond
     ((null? (caddr t)) (cons (cadr t) (car t)))
     (else
      (let ((r (removemax-BST (caddr t))))
      (cons (list (car t) (cadr t) (car r)) (cdr r))
        ))
     )))

(removemax-BST '(73 (31 (5 () ()) ()) (101 (83 () (97 () ())) ())))
=> ((73 (31 (5 () ()) ()) (83 () (97 () ()))) . 101)
```

Give a corresponding function for removemin-BST such that:

```
(removemin-BST '(73 (31 (5 () ()) ()) (101 (83 () (97 () ())) ())))
=> ((73 (31 () ()) (101 (83 () (97 () ())) ())) . 5)
```

```
(define removemin-BST
  (lambda (t)
    (cond
     ((null? ( ____________________________________________________))


              (cons (____________________________________________________)


                    (____________________________________________________)))
     (else
      (let ((r (removemin-BST (cadr t))))
      (cons (list

                  (____________________________________________________)


                  (____________________________________________________)


                  (____________________________________________________) )
         (cdr r)))
        ))
     ))
```

Question 9 Python Slices [3 marks]

Given the following list (array) what is the result of the slice commands below.

animals = ['giraffe', 'tiger', 'monkey', 'mouse']

>>> animals[0:2]

giraffe, tiger,monkey

>>> animals[0:3]

>>> animals[0:]

>>> animals[:]

>>> animals[1:]

>>> animals[1:-1]

Question 10 Python List Comprehension [6 marks]

a) Given the following list (array) select all values smaller and equal to 2.

nums = [2, 8, 1, 6]

```
small = [ _____
          _____ ]
```

b) Given the following loop turn it into a list comprehension.

```
numbers = [ 1, 2, 3, 4, 5, 6, 7, 8 ]
letters = [ 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H' ]
fields = []

for l in letters:
    for n in numbers:
        fields.append((l,n))
```

```
numbers = [ 1, 2, 3, 4, 5, 6, 7, 8 ]
letters = [ 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H' ]


fields = [ _____
           _____ ]
```

c) Given the following list (array) select all strings containing the letter a and insert them in upper case into the afruit list (array).

fruits = ['apple', 'cherry', 'bannana', 'lemon']

Hint: `str.upper()`
`Return a copy of the string with all the cased characters converted to uppercase.`

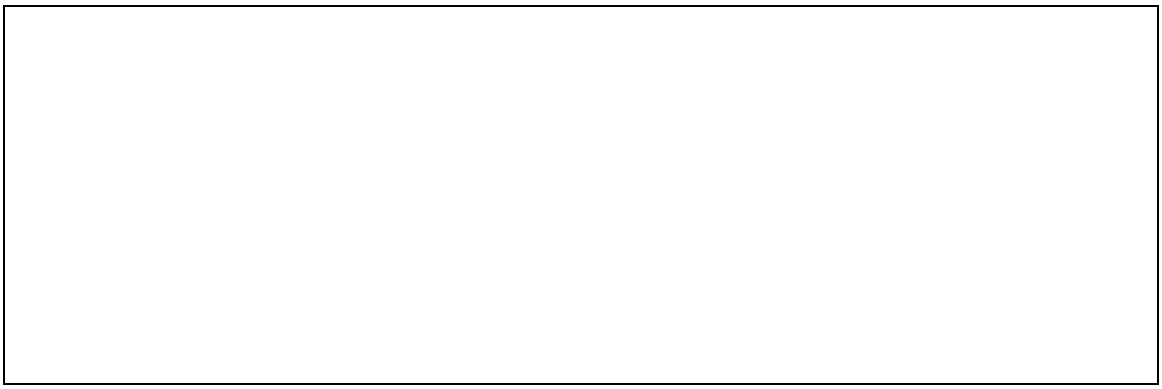
```
afruit = [ _____

           _____ ]
```

Should do the same here as: `afruit = ['APPLE', 'BANANA']`

Question 11) Go [4 points]

a) What is printed by the following program:

```
package main

import (
    "fmt"
    "time"
    "strconv"
)

var i int

func prepare(cs chan string) {
    i = i + 1
    cakeName := "Cake " + strconv.Itoa(i)
    fmt.Println("Preparing ...", cakeName)
    cs <- cakeName // send
}

func receive(cs chan string) {
    s := <-cs
    fmt.Println("Received: ", s)
}

func main() {
    cs := make(chan string)
    for i := 0; i<3; i++ {
        go prepare(cs)
        go receive(cs)

        time.Sleep(1 * 1e9)
    }
}
```



Question 11) (*continued*)

b) Complete the following two methods:

```
package main
import "fmt"

type rect struct {
    width, height int
}



func _____________________________ area() int {

    return r.width * r.height
}


func _____________________________ perim() int {

    return 2*r.width + 2*r.height
}


func main() {

    r := rect{width: 10, height: 5}

    fmt.Println("area: ", r.area())
    fmt.Println("perimeter:", r.perim())
}
```