UNIVERSITY OF TORONTO
Faculty of Arts and Science
APRIL 2014 EXAMINATIONS
CSC 324H1S
Instructor(s): G. Baumgartner
Duration — 3 hours
No Aids Allowed

Student Number: ____________________________
Last (Family) Name(s): ________________________________
First (Given) Name(s): ________________________________

__________

Do not turn this page until you have received the signal to start.
In the meantime, please read the instructions below carefully.

Marking Guide

# 1: ______/10
# 2: ______/20
# 3: ______/10
# 4: ______/15
# 5: ______/15
# 6: ______/10
# 7: ______/10
TOTAL: ______/90
Question 1. [10 marks]

Part (a) [5 marks]
Implement a no-argument function ratchet that returns a no-argument function. When the returned function is called the first time it returns #t, and every time after it returns #f.

(define r0 (ratchet))
(r0) ; #t
(r0) ; #f
(define r1 (ratchet))
(r0) ; #f
(r1) ; #t

Part (b) [5 marks]
Draw the semantic diagram for the state at the end of the above example usage.
Question 2.  [20 marks]
Use map apply and filter wherever appropriate, except if an earlier part is useful (then use that part).
Do not use conditional expressions containing literals #t or #f.
Do not make helper functions nor helper syntactic forms.

For syntactic forms: use component names that clearly indicate which ones are identifiers versus expressions. For component expressions that must produce a certain type of value, indicate that in their names as well. Avoid multiple clauses when “…” would work instead.

Part (a)  [5 marks]
Implement curry taking a binary function f and a value, returning a unary function that behaves like f
but using the value for its first argument.

Part (b)  [5 marks]
Implement filter2 taking a boolean binary function p and two lists of equal length.
It calls p on elements of the two lists in parallel, returning a list of 2-element lists containing the pairs of
elements for which p is true.

Part (c)  [5 marks]
Implement nth taking a list and a natural number n, returning the nth element of the list.
Use the unary function range, which takes a natural number n and produces a list of all natural numbers
up to but not including n.

Part (d)  [5 marks]
Implement R taking a sequence of one or more expressions and randomly evaluating one of them for the
result.
Question 3. [10 marks]

Consider the following data language for a subset of first-order logic:

(<p> <v>) where <p> is a predicate name and <v> is a variable name.

(¬ <pf>) where <pf> is a formula.

(<pf1> ∧ <pf2>) where <pf1> and <pf2> are formulae.

(∀ <v> <pf>) where <v> is a variable name and <pf> is a formula.

Implement unary function uniquify taking a formula and returning a logically equivalent formula with every quantified variable given a unique name. For example, (uniquify '(∀ x ((P x) ∧ (∀ x (P x))))) could produce: '(∀ x0 ((P x0) ∧ (∀ x1 (P x1))))

Use match appropriately.

You may make helper functions, but if so use higher-order functions where appropriate.

You may use functions from other parts of the exam.

You may assume none of the variable names already contain digits.

You may use the following helper function to generate a name from a number:

(define (name n) (string->symbol (string-append "x" (number->string n))))
Question 4. [15 MARKS]
For this question, follow the same requirements as stated at the start of Question 2.

Recall boolean binary function \( \text{exists?} \) from Assignment 2. It takes a boolean unary function \( p \) and a list, returning whether \( p \) is true for some element in the list.

Recall the unary function from lecture taking an s-expression and returning a list of all its sub-parts/sub-s-expressions. For example, the sub-parts of \((3 \ (\ (3 \ 2 \ 4)))\) are:

\[
(3 \ (\ (3 \ 2 \ 4))), \ 3, \ (\), \ (\ (3 \ 2 \ 4)), \ (3), \ 3, \ 2, \ 4.
\]

Part (a) [5 MARKS]
Implement a curried boolean function \( \text{sub-part?} \) taking a value and an s-expression, returning whether the value is one of the parts of the s-expression.
Use \( \text{exists?} \) appropriately. Do not make a list of all the sub-parts.

Part (b) [5 MARKS]
Implement an operation \( U \) for producing a list by appending lists. An example usage is:

```lisp
#; Produces: (c b a e d i h g f)
(U (reverse l) for l in (cons '(a b c) (list '(d e) '(f g h i))))
```

Part (c) [5 MARKS]
Implement unary function \( \text{sub-parts} \) taking an s-expression and returning a list of all its sub-parts (the order is your choice).
Question 5. [15 MARKS]
Recall the backtracking function an-element from lecture:

(require "CSC324.2014W.Backtracking.Library.rkt")
(define (an-element l)
  (if (empty? l) (fail) (< (first l) (an-element (rest l))))))

Part (a) [5 MARKS]
Implement backtracking function X (Cartesian Product) taking a list l and a natural number n, returning a list of length n where each of its elements is an element of l.

Part (b) [5 MARKS]
Recall assert from Assignment 3:

(define-syntax-rule (assert e) (unless e (fail)))

Re-implement X, without using any conditionals (such as if, cases, cond, match, unless, when, etc). Use ! to minimize the number of asserts and numeric checks.

Part (c) [5 MARKS]
Implement a Prolog predicate x(L, N, X).
For instantiated list L and instantiated natural number N it instantiates X to a list of length N where each of its elements is an element of L.
Use ! to minimize the number of numeric checks.
Question 6. [10 MARKS]
For this question be sure to use patterns instead of "=".

Part (a) [5 MARKS]
Implement a Prolog predicate insert(N, L, LN).
For instantiated number N and instantiated non-decreasing list of numbers L it instantiates LN to L with N
inserted at an appropriate spot to keep the list non-decreasing.

Part (b) [5 MARKS]
Implement a Prolog predicate sort(L, S).
For instantiated list of numbers L it instantiates S to the list of those numbers in non-decreasing order.
The algorithm must be insertion sort.
Question 7. [10 marks]
Consider the following Prolog database that tries to anticipate low class attendance during a particular week next term.

class(monday, 324).
class(wednesday, 324).
class(friday, 324).

storm(wednesday).
storm(friday).

assignment(monday, 301).
assignment(wednesday, 309).

skip(Day, Course) :- class(Day, Course), storm(Day).
skip(Day, Course) :- class(Day, Course), assignment(Day, _).

Part (a) [5 marks]
Write all the answers to the following query, in the order they're returned:

?- skip(When, 324).

Part (b) [5 marks]
Rewrite skip to avoid duplicate answers, and still provide all useful answers to the general query skip(When, What
Use the space on this “blank” page for scratch work, or for any answer that did not fit elsewhere.

Clearly label each such answer with the appropriate question and part number.
Use the space on this “blank” page for scratch work, or for any answer that did not fit elsewhere.

Clearly label each such answer with the appropriate question and part number.
Please write nothing on this page.