I agree to abide by the student conduct code at the University of Hawaii in all my work at UH.

Signature:____________________________ Name (print): KEY Master

Practice Final Examination
ICS313 -- Programming Language Theory

Open book includes
1. Scott’s Programming Language Pragmatics, 3rd ed
2. a Lisp reference (e.g. CLQR or Siebel, or another Lisp book)
4. Emacs reference card (purple sheet handout only).

NO notes except one note card – maximum size 4” by 6” text/graphics on both sides. Your crib sheet must be submitted with the exam. (It is OK to use only that much of a full-sized piece of paper).

No calculators, cell phones, computers, music players or other electronic devices.
Blank scratch paper as needed.

Instructions
Write or print clearly.

Show your solution method/reasoning (even on multiple choice and T/F)
You have 120 minutes to complete the exam. There are a total of 150 points on the exam plus 25 points of extra credit.
Make sure that your name is on all pages of your exam and on the crib sheet (if any).

Good luck!!

<table>
<thead>
<tr>
<th>Problem</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7(EC)</th>
<th>ECq</th>
<th>ECcs</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Max</td>
<td>32</td>
<td>16</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>30</td>
<td>25</td>
<td></td>
<td></td>
<td>150</td>
</tr>
<tr>
<td>Score</td>
<td>32</td>
<td>16</td>
<td>24</td>
<td>24</td>
<td>24</td>
<td>30</td>
<td></td>
<td></td>
<td></td>
<td>150</td>
</tr>
</tbody>
</table>
1) (32 p) Multiple Choice – Write the letter for the best answer on the line provided.

I. Which of the following does NOT describe a built-in data type in Perl?
   (a) array
   (b) associative array
   (c) list
   (d) scalar
   (e) none of the above

II. Which of the following is NOT a way to manage complexity in large programs?
    (a) encapsulation
    (b) data hiding
    (c) sub-programs
    (d) constants
    (e) none of the above

III. (Emacs) When editing mystuff.lisp, how would you auto indent
     i) current line and ii) the current region?
    (a) C-x i, M-x i
    (b) C-h indent, C-space
    (c) Tab, M-x p
    (d) Tab, C-M-
    (e) None of the above

IV. Which of the following is not an entry or exit point in a Prolog trace?
    (a) unify
    (b) redo
    (c) fail
    (d) exit
    (e) call

V. Scripting languages do NOT include
    (a) Perl
    (b) Python
    (c) sh
    (d) tcsh
    (e) None of the above

VI. Larry Wall worked on
    (a) Cobol
    (b) Lisp
    (c) Perl
    (d) Prolog
    (e) None of the above

VII. Charles Babbage created
     a) Difference Engine
     b) Analytic Engine
     c) concept of data storage
     d) all of the above
     e) none of the above

VIII. What was the first programmable machine?
     a) analytic engine
     b) ENIAC
     c) Cray supercomputer
     d) weaving loom
     e) none of the above

I C E III D IV A

V E VI C VII D VIII D

2) (16 p) True/False (write your choice on the line provided for each question.) State your assumptions, if any.

a) T/F: Scripting languages like Perl are more readable than Java.

b) T/F: Prolog variables can bind with any type of data structure.

c) T/F: Pass-by-result is used to implement an in parameter.

d) T/F: (Unix) The pipe “|” can be used only once in a single command (at the command prompt).

e) T/F: In Prolog, _59 is a constant.

f) T/F: Perl is more strongly typed than C.

g) T/F: Common Lisp has built-in object-oriented features.

h) T/F: The best programming language is Java.

a) F b) T c) F d) F e) F f) F g) T h) F
3) (24 p) Programming Language Selection

Choose between IMPERATIVE, FUNCTIONAL, LOGIC, SCRIPTING and OBJECT-ORIENTED style languages for the following applications. Name a language and briefly justify each choice.

a) A design simulator for an autonomous underwater vehicle.

   a) This is a very complex problem (autonomous vehicle). Solutions require multiple types of reasoning, suggesting artificial intelligence (AI) reasoning. AI languages (functional) like Lisp or scheme. The ability to use MULTIPLE paradigms so parts can designed and integrated. For example, parts such as signal processing could be compiled for speed, in Lisp or other language such as Fortran. Note: this is an unsolved problem.

b) A program to keep track information and current location of each painting in a museum.

   b) This is a relatively simple database problem. Maintain a database with the important information about paintings. Paintings may be added, moved or deleted. Prolog (logic languages) a reasonable choice since it can retrieve information about multiple facets of the painting (painter, year, style, etc) and the user interaction is not too difficult.

c) An email filtering program to separate messages into several folders.

   c) This is a relatively simple classification problem if based on the existence of specific words in the messages. The solution should be reasonably fast and easy to integrate into the email environment. That suggests scripting (or functional), languages with built-in pattern matching operations such as Perl or Scheme. A more complex solution could be created in an AI language such as Lisp.

d) A program for translation of files from unicode to extended ASCII encodings.

   d) This is a simple substitution problem. Replace each character in one character set, with the equivalent in the other character set. It should be reasonably fast. Scripting languages like Perl are excellent for this type of application. If large files and speed are important, it could be done in a faster compiled language.

4) (24 p) Logic and Prolog

Translate each of the following sentences into Logic, and then into Prolog facts and rules. Demonstrate how your rules answer the query: chases(snoopy, silly).

- Kris has a cat named Silly.
- Tom has a dog named Snoopy.
- Cats and dogs are animals.
- Dogs chase all other animals.
4A. LOGIC PROLOG

there exists cat(silly) cat(silly). [fact 1]

there exists kris, silly
such that owner(kris, silly) owner(kris, silly). [f2]

there exists dog(snoopy) dog(snoopy). [f3]

there exists tom, snoopy
such that owner(tom, snoopy) owner(tom, snoopy). [f4]

all cats are animals
for all C, cat(C) --> animal(C) animal(C) := cat(C). [Rule1]

all dogs are animals
for all D, dog(D) --> animal(D) animal(D) := dog(D). [R2]

dogs chase all other animals
for all Z, Y, dog (Z) and
animal(Y) and Z not equal to Y
chases(Z, Y) :- dog(Z),
(others interpretations possible) animal(Y), Z \== Y. [R3]

Processing Query:
chases(snoopy, silly).

unify with [R3] binds Z = snoopy, Y = silly
true if dog(snoopy), animal(silly) and snoopy \== silly.

check truth:
dog(snoopy) -> yes fact 3
animal(silly) ?
matches animal in rule 1, binds C = silly,
cat(silly) ? -> true fact 1
thus animal(silly) -> yes
Z \== Y
logical check: snoopy \== silly -> true
therefore:
chases(snoopy, silly). -> true
(24 p) Prolog Comprehension

mystery1([], 0).
mystery1([H|T], A) :- mystery1(T, A1), A is A1 + 1.

mystery2([]).
mystery2([ ]).
mystery2([B|C]) :- append(D, [B], C), mystery2(D).

Show all the results returned for each query (as when typing ‘a’ after the first answer).

A) mystery1([], X). ___ X = 0 yes

B) mystery1([bill, sam], Y). ___ Y = 2 yes

C) mystery1([Sally], 3). ___ no

D) mystery2([D, 5, 5, D]). ___ E = 10 F = 9 ? ; no

E) mystery2([E, F, 9, 10]). ___ E = 10 F = 9 ? ; no

F) mystery2([G|H]). ___ H = [ ] ? ;

An infinite number of solutions

5) (30 p) Lisp Programming

Write recursive Common Lisp functions that will calculate the count, sum and average of all numbers, at any level, in a list. For all functions, do error checking and ignore any non-numeric items (except sublists) in the list.

Examples:
(average-recursive ’(( 10 ) 20 “twenty-five” 30 40)) \rightarrow 25
(sum-recursive ’( 10 ( 20 ) 30 40 )) \rightarrow 100
(count-recursive ’( 10 20 ( 30 (40 “Lisp” )))) \rightarrow 4

See ave-rec.lisp

6) (25 p) Extra Credit - Prolog Programming

Write a Prolog database to find all of someone’s second cousins (grandchildren of your grandparents’ siblings). Write facts for the database using only parent(Parent, Child) and married(Person1, Person2). Write rules for secondcousin(Cousin1, Cousin2). You may use rules with additional predicates. Your database must return the exact answer shown to the following query
secondcousin(jeff, X) \rightarrow X = mary; X = sam; no...

EC solutions not provided