Final Examination, Cmput 325

INSTRUCTIONS: The time for this test is two hours. No references should be used. Write all answers in these pages. The mark total for this exam is 90.

Notation convention: In formulas, we use x, y, z, ..., to denote variables and a, b, c, ..., to denote constants.

`Question 1 (25 marks):_______________`

`Question 2 (25 marks):_______________`

`Question 3 (27 marks):_______________`

`Question 4 (13 marks):_______________`

1. (25 marks)

   (a) (2 marks) Why is the notion of a *variable* in an imperative programming language quite different from that in an applicative programming language?

   (b) (2 marks) Give two major differences between functional programming and logic programming.

   (c) (4 marks) Prolog is an implementation of the abstract proof procedure called *resolution*. Give two important differences between these two (i.e., indicate where (pure) Prolog is not an entirely faithful implementation of resolution).

   (d) (4 marks) What do we mean by $\Gamma \models A$, and what do we mean by $\Gamma \vdash A$. Especially, what is the difference between them?

(e) (3 marks) What does the Church-Rosser property say? Why is this property interesting at all?

(f) (4 marks) In lambda calculus, a term may be reducible in more than one way. Which *order* of reduction can guarantee termination whenever a normal form exists? Which *order* of reduction is used in Lisp (just consider the one used in this course), and what is the advantage of it over the other?

(g) (2 marks) Give at least one good reason why we are interested in the foundation of a programming language at all?

(h) (2 marks) Given two arbitrary propositional formulas, how can you determine if they are logically equivalent?

(i) (2 marks) Given two arbitrary formulas without free variables in predicate logic, when we say they are logically equivalent, what do we mean?

2. (25 points)

2.1 (3 marks) Determine if the two given formulas are unifiable. If yes, show the most general unifier.

$$p(f(a, x), x)$$
$$p(f(y, g(z)), g(y))$$

2.2 (3 marks) Skolemize the following formula

$$\forall x \exists y \forall z \exists w \mid p(x, y, z, w)$$

2.3 (4 marks) Consider the following set of formulas: $\Gamma = \{\neg p(a),\ \exists x \mid p(f(x))\}$. Show Γ is satisfiable.

2.4 (15 marks) Consider P that consists of the following clauses:

$$\begin{array}{ll} (1) & q(b) \\ (2) & p(a) \\ (3) & p(f(x, y)) \leftarrow q(x) \wedge p(y) \end{array}$$

(a) (2 marks) Show the Herbrand universe of P.

(b) (3 marks) Show a Herbrand interpretation I which is a model of P.

(c) (3 marks) Show a Herbrand interpretation I which is not a model of P.

(e) (3 marks) Is $p(f(a, f(a, a)))$ a theorem of P (i.e., is it a logical consequence of P)? And why?

(d) (4 marks) Use resolution to show that $P \cup \{\neg p(f(b, f(b, f(b, a))))\}$ is unsatisfiable. In each step, indicate which clause is used to resolve the goal.

3. (27 marks)

3.1 (5 marks) Given the following Prolog program,

```
ancestor(X,Z) :- ancestor(X,Y), parent(Y,Z).
ancestor(X,Z) :- parent(X,Z).
parent(jeff, marry).
parent(marry, ken).
parent(ken, peter).
```

indicate what type(s) of queries will terminate and what type(s) of queries will not? Is there anything that the programmer can do to improve the termination property?

3.2 (10 marks) Consider the following Prolog program:

```
p([], L).
p([X|L1],[X|L2]) :- p(L1,L2).
s(L1,L2) :- p(L1,L2).
s(L1,[Y|L2]) :- s(L1,L2).
```

For each query below, indicate the result of the Prolog execution (only the first answer needs to be shown when yes is returned).

```
?- s([d], [a,b,c,d]).

?- s([a,d], [a,b,c,d]).

?- s([c,d], [a,b,c,d]).

?- s([a,Y], [X|W]).
```

What exactly does this program do?

3.3 (12 marks) Write a Prolog program that defines the intersection of two sets

```
intersection(S1, S2, S3)
```

where S1 and S2 are lists of constants and S3 is the intersection of S1 and S2. In addition, though S1 and S2 may have repeated constants, redundancy in S3 is not allowed. E.g., the query

```
?- intersection([a,b,a], [a,c], S)
```

should bind `S` to `[a]`.

Assume that (i) we are interested only in queries like the one above, where the first two arguments are ground and the third argument is a variable, and (ii) we are interested only in the first answer. You should not use any Prolog built-in predicates. Any predicate that you use to solve this problem should be defined here.

4. (13 marks) Define a Lisp function SetIntersect such that given two lists L1 and L2 of arbitrary s-expressions,

```
(SetIntersect L1 L2)
```

will return a list that contains exactly those that are elements in both L1 and L2. Further, there should be no redundancy in the returned list. E.g.,

```
(SetIntersect '(a c (b)  c) '((b) c d))
```

returns (c (b)). You may use any Lisp primitive or built-in functions that have been allowed in the assignments.