Final Examination, Cmput 325
April 17, 2PM at Ice Arena (rows 1, 3)

LAST NAME _____

FIRST NAME _____

STUDENT ID _____

INSTRUCTIONS: This is a closed book exam. No books or notes may be used. The time for this exam is 2 hours.

Note that you can use the backs of these pages for sketches.

QUESTION 1 (20 marks): _____

QUESTION 2 (20 marks): _____

QUESTION 3 (30 marks): _____

QUESTION 4 (20 marks): _____

QUESTION 5 (10 marks): _____

TOTAL:_____ OUT OF 100

1. [20 marks] We use the following logic connectives: $\neg$ for *not*, $\wedge$ for *and*, $\vee$ for *or*, and $\leftarrow$ for *logic implication*. We use lower case for constants, functions, and predicates (including propositions), and upper case for variables.

[3 marks] Consider $\Gamma = \{\neg a \vee b \vee c, \quad d \leftarrow a, \quad d \leftarrow b\}$. Is it the case that $\Gamma \models d$? If your answer is yes, show two interpretations that satisfy Γ. If your answer is no, show one interpretation that satisfies Γ but does not satisfy d.

[2 marks] What is an inference rule?

[2 marks] Given a set of function symbols and a set of constants, define the set of *ground terms*.

[4 marks] Suppose we want to define two predicates over students and courses: $register(X, Y)$ means student X is currently registered for course Y; $passed(X, Y)$ means student X has passed course Y. In this case, what is the Herbrand universe and what is the Herbrand base?

[3 marks] For the pair of atoms below, determine whether they are unifiable. If yes, you should also show a unifier.

$$p(Y, f(a), f(Y)) \qquad p(f(X), f(X), f(f(a)))$$

[6 marks] Let Γ consist of the following clauses describing paths on a directed graph.

$$path(X, Y) \leftarrow arc(X, Y)$$
$$path(X, Z) \leftarrow arc(X, Y) \wedge path(Y, Z)$$
$$arc(a, b)$$
$$arc(b, a)$$
$$arc(b, c)$$
$$arc(d, b)$$

where $arc(X, Y)$ means an arc from node X to node Y. Show the least Herbrand model of these clauses; i.e., show the set of ground atoms that are logical consequences of these clauses.

2. (20 marks)

[6 marks] Consider the following Prolog program:

```
p([], L).
p([A|X],[A|Y]) :- p([A|X],Y).
p([A|X],[B|Y]) :- A \== B, p(X,[A|Y]).
```

For each goal below, show Prolog's response to the goal. If "yes" is returned, you should also show the value bound to the variable in the goal. You only need to show the first answer generated by Prolog.

```
?- p([a,b,c,X], [a,e,1,p]).
```

```
?- p([[a,1],[b,2]|L], [[c,3]]).
```

[7 marks] Recall the negation, not/1, in Prolog is defined as:

```
not(P) :- P, !, fail.
not(P).
```

Now consider the following program

```
home(X) :- not(out(X)).    /* X is at home if not out */
out(lily).                 /* lily is out */
husband(john,lily).        /* john is lily's husband */
```

Show Prolog's response to each of the following queries:

```
?- home(lily).

?- home(john).

?- home(X).
```

[7 marks] Consider the following program and the goal:

```
a(X) :- b(X), !, c(X).

b(X) :- d(X).
d(f(Y)).
d(g(1)).
c(f(1)).
c(f(2)).
c(g(1)).

?- a(W).
```

1) Show all the answers generated by Prolog for the goal.

2) Now let's replace the first clause above by

```
a(X) :- b(X), c(X).
```

What answers will be generated by Prolog for the same goal?

3. (30 marks)

Your Prolog solutions should be clean and understandable. You may not get full credits just because your program seems to work. Comments are not necessary unless you feel otherwise. You can only use Prolog builtin predicates for arithmetics, and various equalities, inequalities and comparisons (you don't really need anything else). Each problem here can be solved by 4 clauses or less. So do not think of a complicated way to solve any of these problems.

[6 marks] Define

```
rest(L1,L2,N)
```

where `L1` is an input list, and `L2` is the rest of `L1` after position `N`, assuming that the length of `L1` is at least `N`. E.g.,

```
?- rest([a,b,c,d,e], L, 3).
L = [d,e]
```

[8 marks] Define

```
evenL(L1,L2)
```

where `L1` is an input list, and `L2` is the list of all elements of `L1` that are in even positions. E.g.,

```
?- evenL([], L).
L = []

?- evenL([a], L).
L = []

?- evenL([a,b,c,d,e], L).
L = [b,d]
```

[8 marks] Define the predicate `merge(S1,S2,S3)` where `S1` and `S2` are input lists which are merged to `S3`. A merge preserves the order of the elements from each list. Your program must allow all different merges to be generated by backtracking. E.g., the goal

```
?- merge([a,b],[1,2],W).
```

should return all 6 answers for `W`: [a,b,1,2], [a,1,b,2], [a,1,2,b], [1,a,b,2], [1,a,2,b], [1,2,a,b].

[8 marks] A *queue* is a first-in, first-out data structure. An element can only be put into a queue at the *end* of the queue and gotten out from the *front*. In this problem you are to decide on a representation of queue, and then define some simple predicates on queues.

I. Representation

How do you represent an empty queue?

How do you represent a nonempty queue with n elements $a_1, a_2, ..., a_n$ arriving in that order (i.e., a_1 arrives first, then a_2 and so on)?

II. Operations on queues

Define the following predicates:

- `isEmpty(Q)`: true if `Q` is an empty queue.
- `put(E,Q1,Q2)`: `E` is a given element and `Q1` a given queue, and `Q2` is the resulting queue by putting `E` into `Q1` at the end.
- `get(Q1,Q2,E)`: `Q1` is a given queue where `E` is at the front, and `Q2` is the queue after getting `E` out.

4. (20 marks)

[6 marks] Write a Lisp function

```
(defun gen(N) ...)
```

which, given a positive number N, generates the list of numbers from 0 to N:

```
(gen 5) ==> (0 1 2 3 4 5)
```

[6 marks] Consider the following Lisp program:

```
(defun F(L)
       (cond ((null L) nil)
             ((atom L) (cons L nil))
             ((atom (car L)) (cons (car L) (F (cdr L))))
             (T (append (F (car L)) (F (cdr L))))
       )
)
```

What will be returned after executing the following expressions?

(a) `(F '(a b c))` Your answer:

(b) `(F '((a (b)) c))` Your answer:

(c) `(F '((a . b) c))` Your answer:

In one sentence, what does the program do?

[8 marks] Define a Lisp function

```
(defun remove (L)  ... )
```

which takes a nonempty list L of atoms and removes the nth atoms in L where n is an odd number. For example

```
(remove '(a))  ==> NIL
(remove '(a b c d))  ==> (b d)
```

5. [10 marks]

[2 marks] In the SECD machine, what is the use of the D stack? Name an instruction whose execution involves the use of the D stack.

[8 marks] Compile the expression below to SECD code, and then show all the steps of executing the code by the SECD machine.

```
(* (+ 2 (+ 3 4)) 5)
```

Your compiled code:

Execution: