Name:_____ Seat row **and** number: _____

CSc 372, Fall 2001
Prolog Examination
November 28, 2001

READ THIS FIRST

Fill in your name and seat row/number above.

Do not turn this page until you are told to begin.

DO NOT use your own paper. Do not write on the opposite side of the paper. If you need extra sheets, ask for them.

If you have a question that can be safely resolved by making an assumption, simply write down that assumption and proceed. Examples:

> "Assuming `length(List,Length)`"
> "Assuming `3 / 4` produces a floating point result"

If you have a question you wish to ask, raise your hand and the instructor or teaching assistant will come to you. DO NOT leave your seat.

You may use only language elements and predicates that have been studied in class or mentioned in mail. Here are some predicates that may be useful: `member(X, List)`, `sum(List, Sum)`, `length(List, Length)`, `append(L1,L2,L3)`, `getone(Element, List, Remaining)`.

There are no deductions for poor style. Anything that works and meets all restrictions will be worth full credit, but try to keep your solutions brief to save time.

Unless otherwise specified, predicates should produce only one result.

You need not include any explanation in an answer if you are confident it is correct. However, if an answer is incorrect, any accompanying explanation may help you earn partial credit.

This is a **forty-five** minute exam with a total of 100 points and eight possible points of extra credit. There are eleven regular problems and an extra credit section with eight problems.

When you have completed the exam, enter your name on the exam sign-out log and then hand your exam to the instructor or the teaching assistant.

When told to begin, double-check that your name and seat/row are recorded at the top of this page, and then put your initials in the lower right hand corner of each page, being sure to check that you have all the pages.

Problem 1: (5 points)

Show an example of each of the following:

A fact:

A rule:

A query:

A clause:

An atom:

Problem 2: (2 points)

What is the relationship between facts, rules, and clauses?

Problem 3: (5 points)

True or false: The following is a working implementation of the `member` predicate, as studied in class:

```
member(X, [X]).
member(X, [_|T]) :- member(X, T).
```

Problem 4: (5 points)

True or false: The following is a working implementation of the `length` predicate, as studied in class:

```
length([], 0).
length([_|T], Sum) :- length(T, Sum), Sum is Sum + 1.
```

Problem 5: (12 points)

Write a predicate `sumval(+List, +Value, -Sum)` that produces the sum of all occurrences of `Value` in the list `List`. Assume that `Value` and all elements in `List` are integers. `sumval` should produce only one result. You may abbreviate `sumval` as `sv`.

```
| ?- sumval([3, 2, 1, 1, 3, 1], 1, Sum).
Sum = 3 ?

| ?- sumval([3, 2, 1, 1, 3, 1], 3, Sum).
Sum = 6 ?

| ?- sumval([3, 2, 1, 1, 3, 1], 5, Sum).
Sum = 0 ?

| ?- sumval([ ], 10, Sum).
Sum = 0 ?
```

Problem 6: (4 points)

Write a predicate `sumvals(+List, +ListOfValues, -Sum)` that produces the sum of all occurrences of members of the list `ListOfValues` in the list `List`. Assume that `Value` and all elements in `List` are integers. `sumvals` should produce only one result. You may abbreviate `sumvals` as `svs`.

Note that the only difference between `sumval` (the previous problem) and `sumvals` is that `sumvals` has a list of values that are to be included in the sum..

```
| ?- sumvals([3,2,1,1,3,1], [1], Sum).
Sum = 3 ?

| ?- sumvals([3,2,1,1,3,1], [1,3], Sum).
Sum = 9 ?

| ?- sumvals([3,2,1,1,3,1], [1,3,2], Sum).
Sum = 11 ?

| ?- sumvals([3,2,1,1,3,1], [-1,-3,-2], Sum).
Sum = 0 ?

| ?- sumvals([3,2,1,1,3,1], [], Sum).
Sum = 0 ?

| ?- sumvals([], [], Sum).
Sum = 0 ?

| ?- sumval2([3,2,1,1,3,1], [3,1,1,3,1,3], Sum).
Sum = 9 ?
```

The order of values in `ListOfValues` is inconsequential. Note that a given value may appear multiple times in `ListOfValues` but that does not affect the result.

Problem 7: (12 points)

Write a predicate `listeq(+L1, +L2)` that succeeds if the lists `L1` and `L2` are identical and fails otherwise. `L1` and `L2` may be arbitrarily complicated lists but all values will be either integers or lists.

```
| ?- listeq([1,2,3],[1,2,3]).
yes

| ?- listeq([1,2,3],[1,2,3,4]).
no

| ?- listeq([1,[2,[],[3,4,5],[6],8],  [1,[2,[],[3,4,5],[6],8]).
yes

| ?- listeq([1,[2,[],[3,4,5],[6,[7]]],8],  [1,[2,[],[]]]).
no

| ?- listeq([],[]).
yes

| ?- listeq([],[[]]).
no
```

The instructor believes there is a significant chance of making a careless error in the answer for `listeq` that may result in a major deduction. BE CAREFUL AND DOUBLE-CHECK YOUR ANSWER!!

Problem 8: (15 points)

Write a predicate `consec(+Value, +N, +List)` that succeeds if and only if `List` contains `N` consecutive occurrences of `Value`. Assume that `Value` and all elements of `List` are atoms or integers. Assume `N > 0`.

```
| ?- consec(a, 2, [b,c,a,a,b,a]).
yes

| ?- consec(a, 2, [b,c,a,b,a]).
no

| ?- consec(a, 1, [b,c,a,b,a]).
yes

| ?- consec(b, 3, [b,b,c,b,bb,bbb,a,b,a]).
no

| ?- consec(b, 3, [b,b,c,b,b,b,a,b,a]).
yes

| ?- consec(1, 4, [a,b,c,d,1,2,3,4,1,1,1,1]).
yes
```

For this problem you may assume that you have a `repl(X, N, List)` predicate:

```
| ?- repl(b,3,L).
L = [b,b,b] ?
```

Problem 9: (15 points)

Write a predicate `order3(+L1, -L2)` that assumes that `L1` contains three integers and instantiates `L2` to be a list of those integers in ascending order:

```
| ?- order3([3,1,4],L).
L = [1,3,4] ?

| ?- order3([4,1,3],L).
L = [1,3,4] ?

| ?- order3([4,-1,3],L).
L = [-1,3,4] ?

| ?- order3([1,1,1],L).
L = [1,1,1] ?
```

Note: The instructor's solution uses `getone(X, L, Remaining).`

Problem 10: (20 points)

In this problem you are to write a predicate `inventory/0` that does an inventory calculation for a fruit stand. There are instances of `fruit/1` and `cost/2` for each type of fruit that the stand may stock. There are instances of `qty/2` (quantity) for each type of fruit that is currently in stock.

Example:

`fruit(apple).`
`fruit(banana).`
`fruit(orange).`
`fruit(pear).`
	`cost(apple, 30).`
	`cost(pear, 50).`
	`cost(banana, 15).`
	`cost(orange, 25).`
		`qty(apple, 3).`
		`qty(orange, 5).`

There four types of fruit, and the cost of each is expressed in cents. There are three apples and five oranges currently in stock. For the above collection of facts, `inventory` does this:

```
| ?- inventory.
apple: 3 at 30 = $0.90
banana: 0 at 15 = $0.00
orange: 5 at 25 = $1.25
pear: 0 at 50 = $0.00

yes
```

Note that `inventory` lists the fruits in the order the `fruit` facts appear. The elements in each line of output are separated only by spaces and thus the output text weaves back and forth. Note that `inventory` always succeeds.

There will be at least one fruit and each fruit fact will have an associated cost fact present. At least one fruit will be in stock. (That is, there will be at least one `qty` fact.) Assume that printing a floating point number with `write` or `format` produces two places to the right of the decimal point—exactly what's needed.

Problem 11: (5 points)

For each of the two following queries, write in the values computed for each variable. If a query fails, indicate it.

```
| ?- X = [1,2,3], Y = [4|X], [A,B|C] = Y.

A =

B =

C =
```

```
| ?- A = [], B = 1, C = [A,B], B = 2, [D,E] = C.

D =

E =
```

EXTRA CREDIT SECTION (one point each)

(a) What is the Prolog 1000?

(b) In what country was Prolog developed?

(c) What country made a big investment in Prolog?

(d) What language was used for the first implementation of Prolog?

(e) What is the sound of a combinatorial explosion?

(f) What is inaccurate about this specification: `append(+L1,+L2,-L3)`?

(g) Why is a warning about a singleton variable significant?

(h) Recall that the ML functions `ord` and `chr` convert between ASCII character codes and ASCII characters. In a Prolog library you see these two predicates: `get_chr(+Number, -Char)` and `get_ord(+Char, -Number)`. What's odd about that?