Spring, 1992 Name: _____

CS 541 — Programming Languages 1

Logic Programming Test (Features, Version 1)

Before you begin, please take a moment to look over the entire test so that you can budget your time.

If you get an incomplete on this test, you can get at most a 'B' on the makeup test.

Short answer questions will be graded on the basis of completeness (did you list all the relevant parts of the answer?) soundness (did you make any errors of fact or contradict yourself?) and clarity. You can help ensure clarity by giving examples where possible.

For essays, critical justifications, and the like there may be no one right answer; so the criteria for judgement will be whether your essay is:

- clear (you use examples where appropriate, you write clear sentences, diagrams, and you avoid excess verbosity),
- sound (you start from facts or reasonable assumptions, your logic is convincing), and
- complete (you consider relevant aspects of the problem and alternatives).

The highest grade will be earned by an answer that is, in addition, concise and elegant; that is, answers that say the most with the least amount of excess. Extra consideration will be given to answers that are especially creative. For programming problems, there is usually more than one right answer, hence it is important to present with your program evidence that it is correct. Answers to programming questions will be scored on the following points:

- elegance and clarity (conciseness, using relevant techniques to modularize the solution, making good use of language features, the creativity of your solution, and how clear and convincing your argument is that the code is correct)
- correctness (whether your program solves the given problem correctly in the given programming language)

You should try to eliminate all syntax errors from your programs, as they will affect both the correctness and clarity of your solution; you may receive no points for programs with major syntax errors. Run-time space and time efficiency of programs is not important, unless the problem says so. Also, checking of external inputs, prompting, labeling output, and other conventions for dealing with humans are not important (for this class), unless the problem says so.

The breakdown of points between these categories is roughly as follows.

- 60% for elegance and clarity (using relevant techniques to modularize the solution, making good use of language features, the creativity of your solution),
- 40% for correctness (whether your program solves the given problem correctly in the given subset of the given language)

You are encouraged to use auxiliary subroutines whenever they are helpful.

Don't feel obligated to fill up all the space on the page. Conversely, don't feel limited by the space on the page, since you can use the back if you explicitly tell us on the front when you do so.

1. (5 points) What is a substitution? Give an example.

2. (4 points) What is a ground term? Give an example.

3. (8 points) What is a unifier? Why is it important in Prolog?

4. (10 points) How do the concepts of logic programming apply to database management systems? (What similarities are there, what differences are there?)

Trace the execution of the following Prolog program; that is, give the output, in order, generated by calls to `writeln`, and then the result that Prolog prints for the query. (You can make up your own internal variable names if necessary.) If the query gets into an infinite loop, say so (and give some of the output). Assume that writeln is defined as follows.

```
writeln(X) :- print(X), nl.     % write X and then a newline on the output
```

5. (10 points)

```
rt(X, app(i,X)) :-
        writeln(app(i,X)).
rt(X, app(app(k,X),Y)) :-
        writeln(app(app(k,X),Y)).
rt(W, app(app(app(s,X),Y),Z)) :-
        writeln(app(app(app(s,X),Y),Z)),
        rt(W, app(app(X,Y), app(X,Z))).
?- rt(Z, app(app(app(s,k),k),Z)).
```