Marking Scheme for Test 1 (version A)

Test 1- version B and C have similar solutions.
Here are the main points I was looking for in your answers:

Question 1.

Different in two aspects:

1- Semantics: logic (focus on what rather than how) vs. state-based involving variables and assignments
2- Computation: reasoning vs. state transition

Question 2.

Converting to propositional clauses
Negating the query
Using refutation and obtaining the empty clause
Concluding the answer based on the empty clause (inconsistency)

Question 3.

For this query: ?- delete(1, [1,2,1], List).

a)
Starting with query (linear refutation) & applying prolog rules (top to down, left to right)
Properly labelling and mentioning the unifiers
5 branches in search tree
Renaming variables
3 answers by Prolog: [2,1], [1,2], false
Correct back substitution for second answer: List=[1|T2]=[1,2|T3]=[1,2]
Not matching with C0 when not unifiable
b)
Unique search tree since linear refutation, left to right subgoals, and resolving from top to bottom in list of facts and rules

Question 4.

All steps in conversion to CNF, conversion to logic programming notation:
m(X,Y):- n(X).
n(X):- m(X,g(X)).
Noting they are Horn clauses and rules, therefore definite program.

Question 5.

A code similar to the following:

```
bin2dec(L,N):- bin(L,0,0,N).
bin([],_,A,A).
bin([H|T],B,A,N):- A1 is A + H * 2^B,  B1 is B+1, bin(T,B1,A1,N).
```

Question 6.

A code similar to the following:

```
read_write_code :- read(X), check(X).
check(end_of_file):- !.
check(X):- functor(X, _,N), \+(N=2), !, write('Error: two arguments are needed!'), nl,
read_write_code.
check(X):- X=..[H|L], L2=['Processed'|L], Y=..[H|L2], get_more(Y).
get_more(Y):- write(Y), nl, read_write_code.
```

Question 7.

a)

Note that both are using accumulators, and work for the example given by Prof. X.

```
?- countL(a,[a, b, c],0).
false.

 ?- countL(a,[a, b, c],N).
N = 1 ;
```

false. (although one student's code does not return an extra false)

b)

Note the difference of cut and not, the potential problems with cut

Note that code B does not answer if L is not a list, or X is a list

Note that code A can see nested lists

I will use **countX** and **countY** to refer to student A and B responses in the following example queries.

Nested lists:

```
16 ?- countY(a,[a,[a, b], c],N).
N = 2 ;
```

false.

17 ?- countX(a,[a,[a, b], c],N).
N = 1.

If second argument is not a list:
?- countX(a,a,N).
false.

?- countY(a,a,N).
N = 1 ;
false.

If first argument is a list:
?- countX([a], [a], N).
N = 0 ;
false.

?- countY([a], [a], N).
N = 0 ;
N = 1 ;
false.

4 ?- countX([a], [a], 1).
false.

5 ?- countY([a], [a], 1).
true ;
false.

Potential problems with cut:
?- countX(a, [a,b,a,c], 1).
true.

?- countY(a, [a,b,a,c], 1).
false.