Use your own paper to answer the following questions. The exam has 100 points total, 20 points for each of the five questions. You may use your notes (including my handouts) but not your textbook, and not your neighbors.

Name:

Student ID Number:

1. In my original grammar for our programming assignments I got in trouble with the production term ::= not expression, since the resulting grammar was ambiguous. Suppose we attempted to remove this problem by moving the handing of the not operator from the bottom to the top of the expression part of the grammar, as in the following:

   expression ::= not expression  
                | relExpression  
                | expression logicalOperator relExpression

Is the resulting grammar still ambiguous? How would you demonstrate this?

2. Let us simplify the grammar rule for reference slightly:

   reference ::= identifier  
                | reference ^  
                | reference dot identifier  
                | reference [ identifier ]

show the deterministic finite automata that would be generated by an LR parser for this grammar. Number your states in preparation for question 3.

3. Using the automata you constructed in question 2, show what actions would be performed in recognizing the input \( a^b \). That is, trace the sequence of states that the parser would move through during the recognition of this input.

4. Suppose we took our grammar and added C++ style cast (or type conversion) expressions, as in

   term ::= ( type ) expression

is the resulting grammar still LL(1)? explain.
5. Imagine we added by reference parameters to our language. Recall that a parameter passed by reference is internally just a pointer, and the pointer value must be dereferenced to obtain the actual value. Consider the following nested functions:

```pascal
function foo ()
var x : integer;

    function bar (byref z : integer)
        var y : integer;

            begin
                print (x + y + z);
            end;
        begin
            bar(x);
        end;

begin
    bar(x);
end;
```

show the internal AST that would be generated for the expression \( x + y + z \). Assume local variables begin at location 4 in the AR, and the first parameter is found at location -52, and the static chain is stored at location 0 in the AR.