name:

Answer the following questions in the space provided. The exam has 150 points total, 25 points for each of the six questions. No notes, no book, and no neighbors.

1. Consider the following grammar:

   \[
   \text{statement ::= if expr then statement else statement} \\
   \quad \quad \mid \text{if expr then statement} \\
   \quad \quad \mid \text{id = expr} \\
   \quad \quad \mid \text{id ( expr )}
   \]

   \[
   \text{expr ::= id} \quad \mid \text{expr + id}
   \]

   What is the follow set for the nonterminal expr?

2. After rewriting the grammar to be LL(1), write a series of recursive descent routines to recognize the nonterminals statement and expr.

3. Give at least 3 reasons why register-style code generation is more difficult than stack-style code generation.

4. Although the Venerable Bede knew already in 730 AD that the Julian year was 11 minutes and 14 seconds too long, it was not until 1582 that Pope Gregory XIII approved the rule we now use for determining if a year is a leap year. This rule says that a year is a leap year if it is divisible by 4, unless it is divisible by 100, in which case it is not, unless it is also divisible by 400. We can write this as follows:

   \[
   \text{if ((year mod 4 == 0) and (year mod 100 != 0)) or (year mod 400 == 0) then printStr("leap year")}
   \]

   show stack-style pseudo-code similar to the kind our compiler would generate for this statement. Assume year is found at location 4 relative to the frame pointer. (By pseudo-code I mean you can assume instructions like push, pop, compare and mod which will operate on the top two elements of the stack)
5. Show the control flow graph that would be generated for the following procedure. Assume c is a 8 by 8 matrix of integer values, indexed from 1 to 8, that begins at offset -100 in the activation record. Then, describe an example optimization of each of the following types that could be performed on this graph:

A. Algebraic transformations
B. Loop invariant code removal
C. Reduction in strength
D. Busy expressions
(You don’t need to rewrite the CFG, just tell me what the optimization would do).

```plaintext
for i := 1 to 8 do
    for j := 1 to 8 do
        if (i == j)
            c[i, j] = 1
        else
            c[i, j] = 0
```

6. Below are four different grammars. One is LL(1), and the other three are not. Identify the LL(1) grammar. For each of the others, give a precise explanation of how the grammar fails to satisfy the LL(1) property.

1. A -> a | AB
   B -> c | dd

2. A -> a | BA
   B -> c | dd

3. A -> a | Bc
   B -> cd |

4. A -> a | BA
   B -> cd | ccd