B1. Compilers (25 points)
Ms. Ma wants to implement an adding machine that uses postfix notation.

1. (10 points) She decides that numbers will be non-negative integers typed in either decimal notation or in hexadecimal notation. For example, the number twelve can be written either as 12 or as 0xC, and one hundred can be written either as 100 or as 0x64. Write a single regular expression that describes exactly the allowed character strings representing numbers in Ms. Ma's adding machine. The empty string of characters and the string "0x" are not allowed.

2. (15 points) Here is the grammar that Ms. Ma wrote to describe the command language for her adding machine:

E --> E E +
E --> E p
E --> number

The operator "p" prints the value of the expression to its left. If Ms. Ma types this command:

2 0xCF + p 3 + p

she expects her adding machine to print the number 209 and then the number 212.

Attempt to construct an LALR(1) parse table for this grammar. Show all your work so that we can see how you are doing it. If it is not possible to construct an LALR(1) parse table, explain why it can't be done.

B2. Compilers (25 points)

1. (8 points) Describe two markedly different approaches to implicit deallocation of dynamic memory and explain their relative advantages and disadvantages. Assume that all the dynamic storage blocks are the same size so that compaction is not needed.

2. (9 points) What is a syntax tree? What is postfix notation? Show the syntax tree and postfix notation representation for the statement a := (b + c) * a. Why are these representations useful as intermediate representations in compilers?

3. (8 points) What is the difference between three-address codes and quadruples? What advantage does quadruples have over triples? What advantage does implicit triples have over quadruples?

B3. Compilers (25 points)

(A) (14 points) Answer 2 of the following 3 questions.

1. How can error recovery be implemented in an LR parser? How can you incorporate error recovery into a YACC/BISON parser specification?

2. Describe a semantic check that can be performed at compile time, and describe how it could be incorporated into the compiler. Specifically, what phase would be responsible for the check, how would the check be incorporated into that phase, and how would recovery be handled? Describe a semantic check that can only be performed at runtime, explain why it can only be performed at runtime, and how the compiler ensures that the check and recovery is performed gracefully at runtime.

3. Describe how turning on the optimizer could potentially cause a runtime error to occur that would not occur if the code were compiled with no optimization. Describe the general cause and then give a specific example using a particular optimization.

(B) (11 points) Answer 1 of the following 2 questions.

1. You have been charged with writing an optimizer for an existing compiler for C.

a. What optimizations would you choose to implement and why?
b. What program representation would you use as the basis for your optimizer? Be sure to describe the representation as you would to someone not familiar with optimizers, or give a simple example. Justify your decision.
c. Sketch the organization of the phases of your optimizer starting with the program representation as input to the first phase, and ending with the optimized program representation as input to the code generator. Be sure to include any phases needed to identify opportunities for optimization and be specific about what is performed in each phase.
d. Justify the order of your phases.

2. You have been charged with writing a register allocator. You need to consider how to allocate the limited number of physical registers to the symbolic registers appearing in the code, and how to determine which register to spill when a register is needed but all the physical registers are already in use.

a. Describe one technique you might consider using for allocating and assigning physical registers to symbolic registers.
b. What kind of information would you need to gather to perform this allocation?
c. Describe and justify a policy for spilling registers.

B4. Compilers (25 points)

Assume that you are given a compiler for a statically scoped, block structured language with no nested functions (similar to C). Assume that the compiler is constructed with the standard separate phases for lexical analysis, parsing, semantic analysis, intermediate code generation, symbol table management, and code generation. The storage allocation mechanism is a stack of activation records.

You have been asked to modify the compiler in several ways. For each modification listed below,
(a) State the phases that would need to be modified.
(b) Describe briefly how the phases would have to be modified to incorporate the modification.

1.(5 points) Add a new looping structure, REPEAT... UNTIL (expression); when a WHILE (expression) ...DO; statement already exists in the language.
2.(5 points) Change the symbol for MOD from "mod" to "%%", keeping the same semantics.
3.(5 points) Require ";" after each statement, rather than using ";" as a statement separator only. For example,

```
while (x < d)
        x = x + 1;
        y = y * x
do;...
```

changes to:

```
while (x < d)
        x = x + 1;
        y = y * x;
do;...
```

4. (5 points) Add pass by reference parameter passing by adding the keyword REF in front of each parameter to be passed by reference.
5. (5 points) Enable error checking, compile-time error messages, and recovery for variables used in the code before they are declared.