B1. Compilers (25 points) Answer all questions.

1. **(14 points)** Consider the following grammar:

S -> Bx | yBz | wz | ywx
B -> w

a. **(5 points)** Build the DFA of LR(0) items for this grammar.
b. **(4 points)** Show that this grammar is not SLR(1). Explain your answer.
c. **(5 points)** Show that this grammar is LR(1). Explain your answer.

2. **(4 points)** Describe two language features that require a heap allocation as opposed to a stack allocation. Be sure to explain why the stack storage will not suffice in each case.

3. **(7 points)** Distinguish how dynamic versus static scoping affects the implementation of accesses to nonlocal variables when blocks can be nested. Particularly, explain how accesses to nonlocal variable storage can be implemented under dynamic scoping versus under static scoping.

B2. Compilers (25 points)

1. **(16 points)** You are charged with designing a type checking system for a new language. Answer the following questions about your type checker.

   a) **(5 points)** The programming language allows programmers to declare new types and give them names. This brings up the issue of name versus structural equivalence. How would you implement name versus structural equivalence in your checker? What are the tradeoffs in structural versus name equivalence in design/implementation of a language? You might want to use examples to justify your claims. Who makes the decision of structural or name equivalence?

   b) **(3 points)** How is type coercion specified by the language handled in your type checker?

   c) **(4 points)** How is operator overloading handled correctly by your type checking?

   d) **(4 points)** Typically, we want as much type checking as possible to be done at compile time. Why? Give an example of a situation when we have to wait until runtime to do the type checking.

2. **(9 points)** There are many different data structures created and used during compilation, for which the compiler writer is responsible for constructing and using. Choose 2 of the following data structures and explain (a) their purpose, (b) challenges in constructing them, and (c) give a small example to demonstrate you know what it might look like for a given situation.

Register interference graph
Call graph
Abstract syntax tree
Control flow graph
Activation record
Runtime stack
Runtime heap
Inheritance graph
String table
Symbol table

B3. Compilers (25 points)

1. Translate the following arithmetic expression

   a = b * (c + d) * (e + f) % f

   into

   a) **(3 points)** a syntax tree
   b) **(3 points)** quadruples
   c) **(3 points)** triples
   d) **(3 points)** postfix notation

2. Consider the following basic block used in subquestions a and b below.

```
S1:  A = 100
S2:  B = 200
S3:  C = A + B
S4:  D = A * 2
S5:  E = B * 2
S6:  F = D - C
S7:  G = E + F
```

We are interested in using registers for all the variables in this block. Notice that we can minimize the number of registers needed by reusing them.

a) **(8 points)** Construct the interference graph for the variables A, B, C, D, E, F and G in the example block above. Color the graph using the minimum possible number of colors, and use this coloring to assign each variable to a register. Give the final code after register allocation.

b) **(5 points)** Construct a data dependence dag for the basic block above using the statement numbers (e.g., S1, etc.) as nodes. The block is above so I've changed the sentence to specify that.

B4. Compilers (25 points)

1. You just got hired by ABC, Inc and your first task is to include a garbage collection algorithm for applications that are running on hand-held devices. You can choose between a reference counting or a mark-sweep garbage collection algorithm. Which algorithm would work best for the following scenarios? Justify your answers.

    a) **(3 points)** The applications on this device will create many directed graphs where objects will refer to each other.

    b) **(3 points)** The applications should not incur any long pause times, that is, the applications that will run on the device are interactive or real-time.

    c) **(3 points)** The device will only run applications that create linked-list data structures.

2. **(5 points)** Explain the difference between external and internal fragmentation.
3. **(6 points)** Single inheritance languages can use a simple technique called "prefixing" for laying out fields in class descriptors. Explain prefixing and why this simple technique does not work for multiple inheritance languages.
4. **(5 points)** Describe what needs to be done to class descriptors to allow new classes to be loaded and extended at runtime.