You are expected to be able to perform the following tasks on the exam:

- List various passes of a modern compiler and describe the advantages, disadvantages, and complications arising from front-end/back-end separation.
- List and describe formal components of deterministic and non-deterministic automata with application to particular instances.
- Compare and contrast regular vs. context-free languages, descriptions, and automata.
- Compare and contrast deterministic vs. non-deterministic finite automata.
- Compare and contrast finite vs. pushdown automata.
- Analyze and synthesize regular expressions.
- Analyze, synthesize, and manipulate finite automata:
  - Thompson’s construction
  - Subset construction
  - Minimization (Hopcroft’s)
- Analyze, synthesize, and manipulate grammars:
  - Derivations
  - Ambiguity
  - Associativity
  - Precedence
  - FIRST and FOLLOW sets
  - Left recursion elimination
  - Left factoring
  - LL(1) conversion
- Discuss space and time costs for finite automata conversion and execution.
- Compare and contrast top-down vs. bottom-up parsing.
• Build a shift-reduce parser for an SLR(1) grammar.
  – Construct LR(0) item sets and automata.
  – Construct SLR(1) parsing tables (ACTION and GOTO).
  – Trace an SLR(1) parser for a concrete example.

• Discuss various static analysis considerations relevant to compilers.
  – Describe the visitor design pattern and how it can help during the construction of a compiler.
  – Write a simple AST visitor using the framework from our semester-long project.
  – Explain the benefits and costs of static and dynamic type checking.

• Analyze Decaf programs.
  – Construct symbol tables.
  – Perform type inference and type checking.
  – Derive type safety proofs.

• Define and discuss specific terms or vocabulary related to any of the above concepts, including a detailed description of why and how they are relevant to the construction of a compiler.