1. Algol Family (Chapter 5)
   (a) for Algol 60, 68, Pascal, and C, be able to describe the historical context of their development (prior programming languages, motivating problems, hardware capabilities, etc.)
   (b) for Algol 60, 68, Pascal, and C, be able to list major contributions and distinctive features

2. ML (Chapter 5-8)
   (a) be able to give examples and read/write code using ML’s:
      i. basic types (unit, bool, string, real, int),
      ii. kinds of data structure types (tuple, record, list, datatype),
      iii. operators (+, −, *, /, div, ^, ::, etc.),
      iv. pattern matching (on tuples),
      v. other keywords (if, let, raise, handle)
   (b) be able to read (answer questions about the behavior of) code using reference cells and while

3. Types (Chapter 6)
   (a) be able to define and explain type errors, type safety, and type inference
      i. be able to identify and give examples of different kinds of type errors
      ii. be able to explain why Java and ML are type safe and why C and Pascal are not
   (b) be able to explain the tradeoff between runtime and compile-time type-checking
   (c) be able to explain what polymorphism is, and be able to give and explain examples of parametric, subtype, and ad-hoc polymorphism
   (d) be able to explain what type inference is and how it is more than simple type-checking
   (e) be able to carry out Hindley-Milner type inference on simple ML expressions

4. Scoping and Activation Records (Chapter 7)
   (a) be able to explain and give examples of block structuring and nesting
   (b) be able to apply static and dynamic scoping rules in block-structured C or ML-like languages
   (c) be able to describe the difference between pass-by-value, pass-by-reference, and pass-by-name
(d) understand how Activation Records implement static scope in block-structured languages (including higher-order languages)
   i. be able to describe what is stored in each of the following activation record fields:
      A. control links,
      B. access links,
      C. return address,
      D. return-result address,
      E. actual parameters,
      F. local variables, and
      G. temporary variable fields
   ii. be able to simulate the behavior of “the reference implementation”’s use of:
      A. control links,
      B. access links,
      C. return address,
      D. return-result address,
      E. actual parameters,
      F. local variables, and/or
      G. temporary variable fields
   iii. be able to explain how the reference implementation implements:
      A. blocks
      B. nested blocks
      C. recursive procedures
      D. static scoping in nested procedures
      E. functions as values
      F. returning function values
   iv. be able to explain why ML requires closures/access links, but C does not

5. Structured Control Flow (Section 8.1)
   (a) be able to explain how and why control structures evolved from conditional/unconditional branches
   (b) be able to explain what spaghetti code is and recognize examples of it
   (c) be able to translate C functions with if (...) {...} while {...} and while statements into functions with goto and if (...) goto ... statements.

6. Exceptions (Section 8.2)
   (a) be able to explain the purpose and general properties of exceptions
   (b) be able to read programs using ML’s raise and handle constructs
   (c) be able to describe the typing rules for ML raise and handle constructs