

CSC 112 Test 3: Study Guide

At this point in the course, one should be able to demonstrate the following new skills:

- An understanding of the interpretation of classes as user-defined types, specifying what values can be taken on by variables of the type and what operations are legal on variables of the type
- An understanding of the relationship between methods and operators
- An understanding of the role of the following four types of files with respect to developing classes: *main*, *class header file*, *class source file*, and *makefile*
- An understanding of the rationale behind developing classes using the file mechanisms described above.

- The ability to select the variables and methods of classes appropriately based on textual descriptions of problems
- An understanding of the difference between *syntax* and *semantics* for a class
- The ability to choose semantics for an operator appropriately based on textual descriptions of problems

- The ability to write default (no parameter), multi-parameter, and copy constructors as well as destructors for simple to medium complexity classes, including those requiring dynamic memory allocation
- The ability to write an assignment operator for simple to medium complexity classes, including those requiring dynamic memory allocation
- The ability to write an arbitrary simple to medium complexity method or operator for a class, given an appropriate description and function header, using both class variables and parameter variables
- An understanding of the meaning of the terms *public* and *private* and their role in the context of developing classes
- An understanding of the meaning of the terms *friend function* and *friend class* and when such friend functions/classes are required
- An understanding of the meaning of the term *this* as employed in developing classes and the ability to employ the term correctly in an arbitrary method
- An understanding of the notion of *symmetric* operators, including understanding when symmetry is important for operators and syntactical issues required for implementing symmetry

- An understanding of the general properties of the `LinkedList` container, particularly as compared to the array container
- An understanding of the variables that compose the `LinkedList` and `Node` classes employed in defining the `LinkedList` container
- The ability to write simple `LinkedList` methods
- The ability to read and understand simple to medium complexity `LinkedList` methods