**Stat 36-650 Code Design Checklist**

The checklists below are intended to give you some guidance in designing effective, maintainable, and reusable software. Review these items throughout your development process.

 My code is DRY (Don’t Repeat Yourself) - each piece of embodied in the code has one unambiguous and authoritative representation.

 I have attempted to approximate Orthogonality by minimizing coupling between different components of my system.

 My classes and functions encapsulate the knowledge they need - and only the knowledge they need - to fulfill their purpose.

 I have been as explicit as possible about the contract that my functions and classes satisfy.

 I have avoided hidden side effects in my functions.

 My functions and classes are each designed to serve one purpose well.

 My code appropriately handles errors and other exceptional circumstances.

 My system’s interface presents a clean and consistent abstraction to the outside world.

I have sought to maintain generalizability and reuse.



 Conditionals, loops, and other changes in the ow of control are made as clear and salient as possible.

 I have returned early from a function when it is clearer.

 Variables are defined as closely as possible to where they are used.

 Variables are made visible for as few lines of code as possible.

 I have minimized nesting level of complex constructs.

 I have broken down complex expressions and statements into more digestible pieces.

I have preferred immutable objects.



Each of my classes has a central purpose and is well named to describe that purpose.



 The interface of each class presents a consistent abstraction.

 My classes hide their implementation details as much as possible.

 I have avoided exposing classes’ member data.

 My classes avoid making assumptions about its users, including its derived classes.

 I use inheritance to capture “is a" relationships and containment to capture "has a" relationships.