Course/Program Title: CSC 232/IST 232 Advanced C++ Programming

Date: 6/2019

Course/Program Team: Trudy Gift (resigned in Dec 2018); Cole Snyder, Instructor; Karen Weil-Yates (Lead instructor for CSS/NetAdmin), review and edited

Expected Learning Outcomes:

- Develop C++ programs that use a variety of data structures such as arrays, 2D arrays, vectors, maps, and queues.
- Design, code and test object-oriented applications that incorporate concepts such as structures, classes and inheritance.
- Develop an understanding of EAFP exception handling and use it to prevent errors in code that could otherwise be accounted.
- Develop an understanding of enterprise coding practices including version control systems and common coding practices and patterns

Assessment: (How do or will students demonstrate achievement of each outcome?)

- Programming Assignments – Students will complete software projects which are designed to demonstrate the use of:
  - Data structures such as Arrays, 2D-Arrays, Vectors, Maps, and Queues
  - OOP concepts such as structs and classes
  - Inheritance and Polymorphism
  - Common coding patterns such as builder, factory, visitor, singleton, etc.
  - See the attached “Final Group Project”
- Examinations – Students will be able to demonstrate:
  - use of the C++ programming language syntax and semantics
  - ability to read and write programs
  - See attached final exam.

Validation: (What methods have you used or will you use to validate your assessment?)

- IST Advisory Committee Recommendations
- ANSI coding practices
**Results:** (What do your assessment data show? If you have not yet assessed student achievement of your learning outcomes, when is assessment planned?)

My assessment data would seem to indicate that students do better when exposed to more hands-on projects: there was a class average of 83% for 6 students. Except for one student that simply was not submitting assignments, the rest of the class this semester did a great job mastering the concepts in the advanced course and showed vast improvement when presented with more hands-on projects that built on lectured content. See attached final project student code and rubric: removing that students score changes the average to 94% (for 5 students). Class average for the final exam was 84% (with all students passing the exam).

**Follow-up:** (How have you used or how will you use the data to improve student learning?)

In my opinion, because learning a programming language is roughly analogous to learning a new natural language, learning how to code has a very steep learning curve. It is my goal to impress upon students that they should code what interests them and pave the way for programming to be a creative outlet. I have learned more writing for projects that interest me personally than I have ever been taught in a classroom and I’m a firm believer in incentivizing coding practice by allowing my students some creative license with their code. Based on grades from last semester and an analysis of time spent and lines of code, I believe that the hands-on programming in this course is where students excel the most, and that the weekly lecture and coding time in class simply provides students with a foundation to build in whatever direction they choose. I will use this data to refine my lectures and provide better demos of each week’s content to strengthen this foundation.

**Budget Justification:** (What resources are necessary to improve student learning?)

Same as last year, I don’t really need a budget, however I would like to continue using the JetBrains CLion IDE and would like to make sure that’s kept up to date on the lab computers. The CLion IDE is free for students to use if they make a student account at the JetBrains website and it offers much more relevant Enterprise functionality (Git integration, TODO, testing features, etc) than previous IDEs.