Course Outcomes Guide

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

Date: 5/23/2017

Course/Program Team: D. Maruszewski

Expected Learning Outcomes:

- Understand OOP concepts such as classes, inheritance.
- Understand other concepts such as friends, templates.
- Ability to build a program under Microsoft Windows.
- Ability to design, code and test object oriented applications
- More experience in debugging programs.
- Enhance the student's problem solving skills
- Learn techniques of top-down design of algorithms and structured programming
- Develop correct executable programs
- Create appropriate documentation

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

- Programming Labs **–** Students will complete software projects which are designed to demonstrate the use of:
  - Data structures such as Arrays, 2D-Arrays, Vectors, and Queues
  - OOP concepts such as structs and classes
  - Inheritance and Composition

- Follow-Up Examinations – Students will be able to demonstrate:
  - use of the C++ programming language syntax and semantics
  - ability to read and write programs

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

- IST Advisory Committee Recommendations
- Data Grouping and Analysis with the use of rubric form the large project (see assessment folder)

Results: (What do your assessment data show? If you have not yet assessed student achievement of your learning outcomes, when is assessment planned?)

- These students were very open to programming concepts and practice, and they weren't afraid to expand upon the lessons
- Students could take real world rules and translate them into code well
- Understanding of the concept of inheritance was an issue
- They could fall into the trap of "spaghetti code"
- Students were able to express themselves verbally and through writing

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

- Creative projects can foster these students learn well beyond the book. This should be continued/expanded.
- Applying real world examples helps much in class and will be continued
- Inheritance should be tackled earlier in the course. Polymorphism might be better to teach than aggregation (which was taught this semester.)
- Proper coding practices were taught but two assignments to enforce them would help. Overall, these should be part of the grade in all assignments.
- Usually expression is not part of the programmers set of strengths. I'll keep looking into this as a new trend or just an anomaly.

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

- Microsoft Visual Studio or the equivalent Microsoft product should be used. It is the industry standard and will give our students an easier transition into the workplace.
- Up-to-date computer will help the focus of a programmer. One thing that the rubrics didn't show that was part of the class was that all programmers need to get into the "zone." They can't casually program. Glitches and software issues were an issue at the beginning of the course. I had to get a room change from ATC 201 to ATC 202.
- Labs like that of http://myprogramminglab.com might help the students get more hands on practice with good feedback. We will try this in both CSC132 and 232 starting Fall 2017