Course Outcomes Guide 2018

Course/Program Title: CSC 132/IST 132 Beginner C and C++

Date: 7/16/2019

Course/Program Team: Fall 2018

Expected Learning Outcomes:

- Understand data types (including list types) and operators and learn to use them effectively.
- Learn core programming concepts including exception handling, input/output, control structures and memory management
- Understand the basics of object-oriented programming in C/C++. Design functions, structs, and classes.

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:
  - input/output statements
  - functions with return values and parameters
  - if, while, and for-loop logic structures
  - arrays
  - file I/O
  - See the attached "Blackjack" program assignment with code
- 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 Project with sample code

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

- IST Advisory Committee Recommendations
- ANSI coding practices
- Google C++ Style Guide

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

I'm a big fan of hands on assessment through programming challenges. Programming is not for everyone and I find that hands on assessments are more engaging and more importantly,

better practice for students actually interested in the field. My first class struggled with some of the labs they were provided but the quality of work that was submitted each week improved dramatically. The quality and professionalism of the work was astounding. Students would find unique ways of addressing problems that I hadn't even considered before. Some students would even go above and beyond the scope of the project rubrics and incorporate elements we'd only touched on briefly in class. Throughout the semester I incorporated two smaller programming projects in addition to a midterm project and a final project. The students were also given weekly reading and practice problems.

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

After sampling all of last year's projects I've identified key areas to cover more closely this year and made adjustments to my lectures and samples in order to provide a more comprehensive learning environment for absolute beginners. Good programming involves strategic logic-based assessment of a problem. As such I have decided to more thoroughly break down sample problems and have resolved to do more coding in class with the students, via the projector screen. It is my goal to have students follow along and write their own "cheat sheet" programs in order to help them gain a better understanding of the tools available to them in the C++ standard library.

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

One of the few changes I decided to implement in my instruction of this course was the use of a newer, more widely used, Integrated Development Environment (IDE). Typically, this course was taught using Bloodshed's Dev-C++ IDE. I opted instead to use JetBrains' CLion IDE. The JetBrains IDE is used more often in a business environment and is a great cross platform tool for those students that prefer to develop in a Linux or OSX environment. It is available for free as a 1-year subscription to anyone with a .edu email address. I would only ask that it be installed in my classroom before the start of the semester, with the mingw-w64 toolchain (which is also available for free, from sourceforge). The students can license it on their first day.