CSCI 200 Exam 1 Learning Outcomes

Course Learning Outcomes

1. Design an algorithm to solve a problem by breaking the overarching problem into smaller modular components using abstraction and object-oriented design with inheritance.
2. Translate the algorithm into a program using proper C++ syntax and fundamental programming constructs (e.g. control structures, I/O, classes).
3. Recite & apply frequently used Linux command line commands and compile a program using a command line build system.

Module Learning Outcomes

C++: C++ Programming

- Discuss the differences between programming languages, specifically Python & C++
- Create a Hello World program, construct a simple interactive application, and build the program via the terminal.
- List C++ primitive data types and explain the appropriate use of each data type.
- List & identify C++ arithmetic operators, translate math equations to C++, and solve arithmetic expressions.
- Discuss the effects of a statically typed language.
- Convert one data type to another.
- Recite the order of operations and evaluate an expression.
- Explain how C++ generates random numbers and write a program that generates random numbers.
- Identify C++ control structures and conclude which branch a sample program will execute.
- List C++ logic operators and evaluate Boolean expressions consisting of multiple logic operators.
- Evaluate the resultant output of a code block containing a control structure.
- Convert a program written with a for loop to a program using a while loop and vice versa.
- Identify and correct errors in program structure and logic.

CLI: Command Line Interface

- Create a Hello World program, construct a simple interactive application, and build the program via the terminal.
- List common Linux terminal commands and choose the correct commands to work with a file system via the command line.
- Describe how a computer generates a program from code.
- Write and use a Makefile.
- Discuss the advantages of using Makefiles.

DE: Design Elements

- Explain how C++ generates random numbers and write a program that generates random numbers.
- Identify C++ control structures and conclude which branch a sample program will execute.
- Identify C++ repetition structures and explain the following terms: looping parameter, stopping condition, and looping parameter modification.
- Explain the appropriate use and differences between a while loop, for loop, and a do-while loop.
- Discuss the design process and strategies for developing good code.
- Identify and correct errors in program structure and logic.
- Implement various techniques to trace & debug a program.
- Discuss the pros/cons of the various debugging techniques.

- Identify the parts of a function.
- Explain the difference between a parameter and an argument for a function. Discuss what can be returned from a function and what a void function is.
- Explain the meaning of the DRY principle and appropriate uses for functions.
- **S**OLID: Discuss how functions contribute towards the Principle of Single Responsibility.
- Define and implement a procedural programming style.
- Explain the difference between a function prototype and a function implementation. Discuss the pros/cons of separate implementations.
- Implement various techniques to trace & debug a program.
- Define an overloaded function and recite common usages for overloaded functions.

MM: Memory Management
- List C++ primitive data types and explain the appropriate use of each data type.
- Explain how values are stored in memory, how the values are interpreted differently based on data type, and list common errors that can occur with data types.
- Diagram how integer and decimal values are represented in binary.
- Convert between binary and decimal formats.
- Explain the concept of local & global scope when functions are used within a program.