Course number and name:	**CS 04103: Computer Science & Programming**
Credits and contact hours:	4 credits / 4 contact hours
Instructor's or course coordinator's name:	Jacob Levy
Text book, title, author, and year:	C++: Early Objects, 10th Edition. Tony Gaddis, Judy Walters, & Godfrey Muganda

Specific course information

Catalog description: This course emphasizes programming methodology, algorithms and simple data structures. A programming language rich enough to allow easy implementation of data structures is studied. Prior programming experience in any programming language is expected for this course.

Prerequisites: None

Specific goals for the course

- **Problem Solving & Design Techniques for Programming.** Students have successfully studied and utilized problem solving & design techniques (e.g. Top-Down Design, Modular coding, the use of functions and classes) that are critical for well-written, large-scale, modular programs. This should include how classes and functions implement the basic tenets of OOP such as the Single Responsibility Principle, the Open/Closed Principle, avoidance of code duplication (Reusability), and Extensibility (inheritance). Lastly, students should have an understanding of what it means to create an "abstraction" and to hide implementation details as needed.

- **Basic Control Structures and Algorithmic Design in C++.** Students have successfully demonstrated an understanding of and have utilized basic control structures including Branching and Selection (if/else & switch), a variety of looping structures, and basic input/output to files. Students should be able to demonstrate how to use them to assemble simple algorithms for problem solving.

- **Simple Data Structures and User Defined Data Types** By the end of the course the students should have a basic understanding of simple data structures (Arrays and Structs) as well as how to model and define and use their own simple composite Data Types using structs and classes/objects. Students have a clear understanding of the differences between instance and static members of a class as well the difference between a specific class instance and the class itself. They should understand and have utilized the technique of inheritance to extend/customize various classes (either or both Standard Library classes and those they have defined themselves)

Required list of topics to be covered. Topics may be covered in a different order than listed, and may be more in depth than what is shown. Refer to your specific class syllabus and Instructor for more details.

1. Basic C++
    a. Primitive Data, variables, constants
        i. scope
    b. cin/cout
2. Selection Structures
    a. if/else
    b. switch
3. Loops
    a. While
    b. For
    c. do
4. Simple File I/O
    a. Reading and Writing text files
5. Problem Analysis/Breakdown
    a. Functions
        i. Instance vs Static
    b. Modular Programming
        i. Separation of Concerns/Single Responsibility Principle
    c. Modeling Data
        i. Structs
        ii. Structs vs Primitive Data
            1. ADTs: How to create simple composite Data types
6. Introduction to Classes
    a. Basic OOP
        i. Intro to Objects
        ii. Objects/Classes vs Structs
            1. Value Type vs Reference Type
        iii. Basic Data Abstraction/Encapsulation/Data Hiding
    b. Data Abstraction and OOP Design
        i. Introduction to Inheritance and extensibility of classes
        ii. Open/Closed Principle
7. Arrays, Vectors
    a. How to work with collections of Data

Optional list of topics that may be included pending on class pace

1. Simple Searching & Sorting
2. Pointers & Dynamic Data
3. Inheritance in Depth
4. Recursion