NORTHERN NEW MEXICO COLLEGE

INTERMEDIATE PROGRAMMING

Course Number **Course Name**	EECE 231L INTERMEDIATE PROGRAMMING
Credit Value (Breakdown of theory and lab credits)	3 credits, 3 Theory + 1 Lab
Catalog Course Description	This class teaches how to write medium complex computer programs that make use of structured decomposition, basic data structures, strings, recursion, files and dynamic memory. Knowledge of basic programming concepts is assumed. Prerequisite: EECE 152L (3, 2T+1L)
Student Learning Outcomes/Objectives /Competencies of the Course	1. Utilize primitive data types and built-in data structures. 2. Describe common applications for each data structure in the topic list. 3. Write programs that use arrays, records, strings, linked lists, stacks, and queues and choose the appropriate data structure for modeling a given problem. 4. Describe a simple hash function. Object-oriented programming: 1. Discuss and identify the concepts of encapsulation, abstraction, inheritance, and polymorphism. 2. Design, implement, test, and debug simple programs in an object-oriented programming language. 3. Describe how the class mechanism supports encapsulation and information hiding. 4. Design, implement, and test the implementation of “is-a” relationships among objects using a class hierarchy and inheritance. 5. Compare and contrast the notions of overloading and overriding methods in an object-oriented language. 6. Describe the relationship between the static structure of the class and the dynamic structure of the instances of the class. 7. Utilize iterators to access the elements of a container. 8. Describe how constructors and destructors relate to the life of an object. 9. Describe the relationship between an object and its corresponding class. Event-driven programming: 1. Explain the difference between event-driven programming and command-line programming. Design, code, test, and debug simple event-driven programs that respond to user events. 2. Develop code that responds to exception conditions raised during execution. Algorithms and Problem Solving: 1. Discuss the importance of algorithms in the problem-solving process. 2. Identify the necessary properties of good algorithms.

	3. Create algorithms for solving simple problems. 3. Use a programming language to implement, test, and debug algorithms for solving simple problems. Apply effective debugging strategies.
College-Wide Student Learning Outcomes	Information regarding which of the following college-wide objectives will be addressed in the course along with which assignment will be used to measure this outcome: 1. Critical Thought