Prepared by the Department of Mathematics
Date of Departmental Approval: March 29, 2017
Date approved by Curriculum and Programs: April 12, 2017 Effective: Fall 2017

1. **Course Number: CSC125**
   Course Title: Procedural Programming

2. **Description:** Students design programs in C using a procedural design paradigm that examines issues associated with low-level programming such as explicit memory management, efficiency, pointers, the compilation process, and debugging. C programs are run in a Unix/Linux environment.

3. **Student Learning Outcomes (instructional objectives, intellectual skills):** Upon successful completion of this course, students are able to do the following:
   - Describe a mental model of program execution that is closer to the actual execution by the machine.
   - Compare and contrast (1) the procedural/functional approach (defining a function for each operation with the function body providing a case for each data variant) and (2) the object-oriented approach (defining a class for each data variant with the class definition providing a method for each operation).
   - Explain the procedural/functional approach and the object oriented approach defined as a matrix of operations and variants.
   - Describe the difference between the implementation of a given program using a dynamic, interpreted, memory-managed language (e.g. Python), and a static, compiled, low-level language (e.g. C), considering language features such as: type declarations, string representations, arrays vs. lists, explicit pointers vs. implicit references, and parameter passing.
   - Analyze and explain fundamental programming concepts including basic syntax and semantics: variables, expressions, and assignment basics types (e.g., numbers, strings, arrays, and lists); conditional and iterative control structures; functions, parameter passing, user-defined functions; algorithms; simple 1/0; modularity and structured decomposition.
   - Design, implement, test, and debug programs for solving problems using structured (functional) decomposition to break a program into smaller pieces.
   - Design and write unit tests.
   - Demonstrate a good understanding of C libraries for input and output, and the interface between C programs and the UNIX operating system.
   - Demonstrate an ability to use UNIX tools for program development and debugging.
   - Demonstrate the central elements of team building and team management.

4. **Credits:** 4 credits

5. **Satisfies General Education Requirement:** No

6. **Prerequisite:** Any college-level programming course

7. **Semester(s) Offered:** Fall

8. **Suggested General Guidelines for Evaluation:** programs, quizzes, tests, group projects

9. **General Topical Outline (Optional):**