Syllabus of Record

Program: CET Siena
Course Code / Title: (SN/CS1110) Introduction to Programming
Total Hours: 45
Recommended Credits: 3
Primary Discipline / Suggested Cross Listings: Computer Science / Data Science
Language of Instruction: English
Prerequisites / Requirements: None

Description
This introductory course in programming, software development, and computer science teaches students computing fundamentals and an appreciation for computational thinking. No previous programming experience is required.

Objectives
Through their participation in this course, students:

- Gain a solid understanding of the syntax and semantics of C programming language. Other languages can be used in addition, but students must have enough familiarity with these concepts to be able to start programming in Java in CS2.
- Experience the analysis of a problem and solution creation.
- Work through the necessary steps to produce a small working program given a sufficient set of requirements.
- Understand that a variety of programming languages are available and that different languages/platforms are designed to simplify solving particular programming problems.

Course Requirements
Active participation is essential in this course. Students are expected to attend each class and lab course component, as outlined in the CET Attendance Policy. Students are expected to read all assigned materials before each class. Reading assignments are generally 20-30 pages per class session. Graded assignments include:

- Homework: weekly programming assignments related to reading and lecture material.
- Midterm Exam
- Final Project: create a large program, such as a game.

Grading
The final grade is determined as follows:

- Participation: 20%
- Homework/Programming Assignments: 40%

- Midterm Exam: 20%
- Final Project: 20%

Readings

Deitel, Paul and Harvey Deitel. *C How To Program*. London: Pearson, 2015.

Hanly, Jeri and Elliot Koffman. *Problem Solving and Program Design in C.* Boston: Addison-Wesley, 2009.

Kernighan, Brian. *The Practice of Programming*. London: Pearson, 1999.

Kernighan, Brian and Dennis Ritchie. *The C Programming Language*. London: Pearson, 1988.

Outline of Course Content

Topic 1 - General Programming and Semantics

- Expressions and rules for evaluation, including evaluation of function applications and uses of symbolic names for values
- The concepts of program state, declaration and initialization of variables, and assignment commands with their evaluation and effects
- Literal values
- The general notion data types; idea of numeric, boolean, list/array, string data types
- Operators, including implicit and explicit type conversion
- Control structures
- Procedures/functions/methods/subroutines; how values are passed and returned
- Variable scope, global versus local variables, non-persistence of local variables
- The use of objects (i.e., importing a package/library and using the methods/objects in it)
- Input/output and interactive computations

Topic 2 - Problem Solving and Abstraction

- Ability to turn textual description of algorithms into working code
- Ability to use abstraction to break a problem into sub-problems
- Ability to define an algorithm that can be implemented in a program.
- Have sufficient familiarity with some set of problems to write code that solves problems which cannot realistically be solved by hand due to size, complexity, etc.
- Use models and software to represent real-world phenomena
- Understand that binary sequences are used to represent digital data

Topic 3 - Basic Software Creation Principles

- Familiarity with syntax, runtime, and logical errors; use of print-based debugging
- Ability to test own code to locate, diagnose, and resolve errors
- Ability to use a tool to create software (i.e. IDEs, text editors, etc.)
- Processing a reasonably large data set (e.g. CSV)

Topic 4 - Different Programming Domains

- The role of different programming languages and software in modern engineering and in other fields
- Different programming languages and their different syntax and semantics
- Libraries in languages and how programmers reuse code to solve some categories of problems easily