Semester:	02				
Course Code:	CO1810				
Course Name:	Programming for Engineers II				
Credit Value:	3 (Notional hours 150)				
Pre-requisites:	None				
Core/Optional:	Core for Electrical and Electronics Engineering, Manufacturing and Industrial Engineering and Mechatronics Engineering specialization streams				
Hourly Breakdown	**Lecture hrs**	**Tutorial hrs**	**Practical class hrs**	**Design hrs**	**Independent Learning & Assessment hrs**
	27	6	24	-	93

Course Aims: To strengthen the computer programming skills and knowledge of engineering undergraduates.

Intended Learning Outcomes:

On successful completion of the course, the students should be able to:

- ➢ Independently **build** computer programs with memory manipulations and I/O operations using the C language, build and run them;
- ➢ **Use** debugging techniques to understand the execution of computer programs and detect errors;
- ➢ **Demonstrate** good practices in programming such as pre-planning, naming conventions, indentation, commenting and modularization in order to write high quality code;
- ➢ **Describe** and use basic static and dynamic data structures in computer programs, along with fundamental algorithms and programming techniques;
- ➢ **Compute** asymptotic runtime complexity of simple algorithms;
- ➢ **Explain** simple programs written in Assembly Language using a given instruction set.

Course Content: *(Only main topics & subtopics)*

- ➢ **Introduction to C programming syntax:** Comments, variables, data types, type casting, operators, expressions, flow control (conditions, loops, interrupting loop flow, program termination), statements & blocks, library functions, basic user inputs & outputs.

- ➢ **Structured programming:** Functions syntax, naming conventions, parameter passing by value

and reference, recursion.

- **Memory allocation and I/O:** Memory layout (stack, heap, global data, code/text), static memory allocation, arrays, string representation, string manipulation, pointers, structures, file input and output.
- **Error detection and prevention:** Syntax/semantic/logical errors, compile-time/run-time errors, assertions, debugging tools, good practices in programming.
- **Time efficiency of algorithms:** Running time and time-complexity of algorithms, use of Big-Oh notation, linear search, sorting algorithms (bubble, insertion, selection), divide and conquer techniques (binary search, quick/merge-sort), hashing.
- **Dynamic data structures:** Dynamic memory allocation and deallocation, implementation of linked data structures (list, queue, stack, tree), efficiency of search and maintenance operations.
- **Introduction to Assembly Language:** Introduction to Instruction Set Architecture, Assembly Language syntax and operations using a specified ISA.
- **Programming Laboratory:** Write structured programs using the C language, memory manipulations, I/O operations, debugging, implementation of data structures and algorithms.

Teaching/Learning Methods:

Flipped classrooms, small group discussion classes.

Assessment Strategy:

Continuous Assessments 50 %	Final Assessment 50 %		
Details: Practicals & quizzes 30% Mid-semester examination 20%	Theory(%) 50%	Practical(%) -	Other(%)(specify) -

Recommended Reading:

- Brian W. Kernighan and Dennis M. Ritchie, The C Programming Language, Second Edition, 1990
- Jeffrey D. Ullman, Alfred V. Aho, Foundations of Computer Science C Edition, 1994