DEREE COLLEGE SYLLABUS FOR:

CSC 4343 Programming Languages and Compilers **3/1/3**
(Fall 2025) **UK LEVEL: 6**
UK CREDITS: 15

PREREQUISITES:	ITC 2088 Introduction to Programming ITC 2197 Object Oriented Programming Techniques ITC 3287 Advanced Object-Oriented & Functional Programming
CATALOG DESCRIPTION:	Programming language concepts, syntax and semantics. Grammars, parsing, lexical analysis, semantic analysis, code generation.
RATIONALE:	The course aims to equip students with deep understanding of the forms, semantics, and design principles of programming languages and, therefore, enable them to effectively communicate with computers and create efficient, robust software. Students will be exposed to compiler design methodologies essential for optimizing code performance. This course fosters critical thinking and problem-solving skills, as students learn to navigate complex programming challenges, a skill that is transferable to various domains, including those involving genAI tools.
LEARNING OUTCOMES:	As a result of taking this course, the student should be able to: 1. Compare and contrast programming paradigms. 2. Assess trade-offs involved in the design of programming languages. 3. Utilize fundamental components of compilers to design a small-scale programming language. 4. Design and develop compiler components.
METHOD OF TEACHING AND LEARNING:	In congruence with the teaching and learning strategy of the college, the following tools are used: • Lectures, class discussions, use of generative AI tools to inform course content, laboratory practical sessions. • Office hours: Students are encouraged to make full use of the office hours of their instructor, where they can ask questions and go over lecture material. • Use of the Blackboard Learning platform, where instructors post lecture notes, assignment instructions, timely announcements, as well as additional resources.
ASSESSMENT:	Summative: 1st assessment: Coursework short problems — **30%** 2nd assessment: Portfolio of student work and oral assessment. — **10%** Final assessment: Individual Project Programming project to address language design and compiler implementation. — **60%** **Formative:** Homework, In class quizzes or lab exercises — **0%** The formative assessments aim to prepare students for the summative assessments and expose them to teamwork. The 1st summative assessment tests LOs 1, 2.

	The 2nd summative assessment tests LOs 1, 2, 3, 4. The final summative assessment tests LOs 1, 2, 3, 4. *The final grade for this module will be determined by averaging all summative assessment grades, based on predetermined weights for each assessment. If students pass the **final summative assessment,** which tests all Learning Outcomes for this module, and the average grade for the module is 40 or above, students are not required to resit any failed assessments.*
INDICATIVE READING:	**REQUIRED MATERIAL:** 1. "Engineering a Compiler" by Keith D. Cooper and Linda Torczon, 2nd ed. 2011. 2. Instructor's notes. **RECOMMENDED READING:** 1. "Introduction to Compilers and Language Design" by Douglas Thain, 2nd ed. 2021. 2. "Compilers: Principles, Techniques, and Tools" by Alfred V. Aho, Monica S. Lam, Ravi Sethi, and Jeffrey D. Ullman. The Dragon book, is considered the "bible" of compilers, though quite old today. 3. "Modern Compiler Implementation in C/Java/ML" by Andrew W. Appel
INDICATIVE MATERIAL: *(e.g. audiovisual, digital material, etc.)*	**REQUIRED MATERIAL:** N/A **RECOMMENDED MATERIAL:** N/A
COMMUNICATION REQUIREMENTS:	Daily access to the course's site on the College's Blackboard CMS. Use of word processing and/or presentation graphics software for documentation of assignments.
SOFTWARE REQUIREMENTS:	Linux O/S and the following tool-chain: flex, bison (or yacc), antlr, LLVM, appropriate IDE (e.g. CLion)
WWW RESOURCES:	https://online.stanford.edu/courses/soe-ycscs1-compilers https://www.cs.cornell.edu/courses/cs4120/2021sp/#:~:text=Overview%20An%20introduction%20to%20the,support%20for%20modern%20programming%20languages
INDICATIVE CONTENT:	1. Introduction 2. Syntax & Semantic a. formal syntax b. lexical analysis & tokenization c. parsing d. abstract syntax trees e. semantic analysis & type checking 3. Language features a. Review of data types & structures, control structures and flow control, sub-routines, functions and procedures, OO features b. FP concepts, concurrency & parallelism 4. Compiler design a. compiler architecture b. lexical analysis c. syntax analysis

	d. intermediate representations e. semantic analysis and symbol tables f. code generation and target architectures g. run-time environments and memory management 5. Advanced compiler techniques a. Just-In-Time compilation b. dynamic and static analysis c. optimization strategies d. garbage collection algorithms e. machine independent optimizations 6. Programming language design and evaluation a. programming language design b. domain-specific languages c. Evaluation criteria