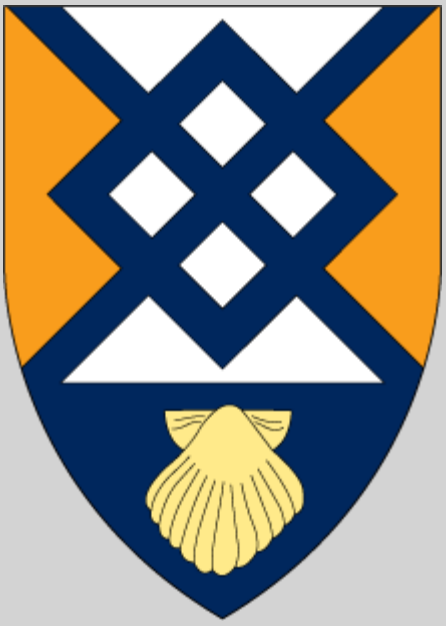

Department	Computer Science
Prefix	CSCI
Number	1170
Name	Engineer Comp Sci I Lab
Description	The course includes hands-on instruction and laboratory exercises in developing programs written in a high-level object oriented programming language applying the principles taught in the CSCI 1370 lecture course.
Credit-hours	1.000 Credit hours
Lecture-hours	0.000 Lecture hours
Lab-hours	3.000 Lab hours
Level	Undergraduate

Schedule-type	Laboratory
Prerequisite	None
Restrictions	None
Course-attributes	Degree Elective, Course fee - Computer Science, CS Instruction Fee - LU
ABET-outcomes	(a) An ability to apply knowledge of computing and mathematics appropriate to the program's student outcomes and to the discipline. (b) An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution. (c) An Ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs. (d) An ability to function effectively on teams to accomplish a common goal. (e) An understanding of professional, ethical, legal, security and social issues and responsibilities. (i) An ability to use current techniques, skills, and tools necesary for computing practice. (j) An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices. (k) An ability to apply design and development principles in the construction of software systems of varifying complexity.
	1. Be proficient with the programming environment and understand the basic aspects of program translation. 2. Analyze a programming problem and develop a solution algorithm. 3 .Use the syntax and semantics of a higher-level language to implement their solutions to programming problems, including the correct use of: Variables and assignment to variables, Primitive types such as integer, character, and floating-point variable types, Commonly-used built-in reference types, e.g., single-dimensional

Learning-outcomes	arrays,strings, Declaration of constants and variables, Assignment, logical, and arithmetic operators, Local and global variables, Selection control structures (e.g., if, nested if, switch), Iterative control structures (e.g., for, while), Functions (user-defined and predefined), Parameter passing to functions (by value and by reference), Arrays and other structures (such as records). 4. Document their solutions. 5. Use functions and parameter passing involving both primitive types and reference types. 6. Apply problem-solving strategies to design a solution to a problem similar to ones seen before. 7. Design simple ADTs to solve a problem similar to one seen before. 8. Apply testing and debugging strategies, including black-box and white-box testing, test drivers, stubs, and test suites, to identify software faults. 9. Formulate complex arithmetic expressions involving operators of differing precedence and associativity; understand the order of evaluation of subexpressions. 10. Formulate complex logical expressions involving multiple and/or/not combinations. 11. Implement nested if statements. 12. Demonstrate proficiency in the use of logic and arithmetic operators and their precedence. 13. Use simple I/O to read/write character and numeric data to/from files and console. 14. Use simple sorting and searching algorithms. 15. Use standard documentation to determine the use of an unfamiliar class or method. 16. Use teamwork roles and methods in the classroom. 17. Discuss and debate ethics in computing.