COURSE OUTLINE: CSD215 - PROG. PARADIGMS

Prepared: Rodney Martin
Approved: Corey Meunier, Chair, Technology and Skilled Trades

Course Code: Title	CSD215: PROGRAMMING PARADIGMS
Program Number: Name	2095: COMPUTER PROGRAMMING
Department:	COMPUTER STUDIES
Academic Year:	2022-2023
Course Description:	The Object-Oriented Programming (OOP) and Functional Programming (FP) paradigms have been important to software design since the dawn of information technology. Using a variety of programming languages, students explore how these paradigms affect approaches to software design. Topics include composition vs inheritance, higher-order functions, mutability vs immutability, currying, map/reduce/filter, and advanced type systems. Students will employ functional approaches in languages they are already familiar with and will also have an opportunity to explore new programming languages.
Total Credits:	4
Hours/Week:	4
Total Hours:	56
Prerequisites:	CSD121
Corequisites:	There are no co-requisites for this course.
Vocational Learning Outcomes (VLO's) addressed in this course: **Please refer to program web page for a complete listing of program outcomes where applicable.**	**2095 - COMPUTER PROGRAMMING** VLO 10 Contribute to the development, documentation, implementation, maintenance and testing of software systems by using industry standard software development methodologies based on defined specifications and existing technologies/frameworks. VLO 11 Apply one or more programming paradigms such as, object-oriented, structured or functional programming, and design principles, as well as documented requirements, to the software development process.
Essential Employability Skills (EES) addressed in this course:	EES 4 Apply a systematic approach to solve problems. EES 5 Use a variety of thinking skills to anticipate and solve problems. EES 6 Locate, select, organize, and document information using appropriate technology and information systems. EES 7 Analyze, evaluate, and apply relevant information from a variety of sources. EES 10 Manage the use of time and other resources to complete projects.
Course Evaluation:	Passing Grade: 50%, D A minimum program GPA of 2.0 or higher where program specific standards exist is required for graduation.

Other Course Evaluation & Assessment Requirements:

To successfully pass this course, the student must receive passing grades for both the Test and Evaluation portion of the class AND the Laboratory portion.

Grade
Definition Grade Point Equivalent
A+ 90 - 100% 4.00
A 80 - 89%
B 70 - 79% 3.00
C 60 - 69% 2.00
D 50 - 59% 1.00
F (Fail) 49% and below 0.00

CR (Credit) Credit for diploma requirements has been awarded.
S Satisfactory achievement in field /clinical placement or non-graded subject area.
U Unsatisfactory achievement in field/clinical placement or non-graded subject area.
X A temporary grade limited to situations with extenuating circumstances giving a student additional time to complete the requirements for a course.
NR Grade not reported to Registrar`s office.
W Student has withdrawn from the course without academic penalty.

Books and Required Resources:

Grokking Simplicity: Taming complex software with functional thinking by Eric Normand
Publisher: Manning
ISBN: 9781617296208
This book is recommended, but OPTIONAL

Course Outcomes and Learning Objectives:

Course Outcome 1	Learning Objectives for Course Outcome 1
1. Discuss the history and nature of the main programming paradigms	1.1 Describe the history of programming languages and their paradigms 1.2 Distinguish between Procedural, Object-Oriented (OOP), and Functional programming (FP) paradigms 1.3 Discuss FP`s mathematical roots 1.4 Identify the paradigms supported by various popular programming languages 1.5 Examine and compare the syntax of languages from a variety of paradigms 1.6 Describe dis/advantages of the various paradigms
Course Outcome 2	**Learning Objectives for Course Outcome 2**
2. Describe and employ functional programming concepts	2.1 Explain the difference between a variable and a value 2.2 Discuss the advantages of immutable vs mutable data 2.3 Discuss how to use an `immutable` approach even in non-FP languages 2.4 Explain the equivalence of recursion and loops 2.5 Discuss the significance of tail recursion 2.6 Define what a pure function is, and explain the advantages of pure functions vs impure functions 2.7 Define what a higher-order function is 2.8 Explain what a lambda expression is 2.9 Explain how higher-order functions and lambda expressions improve software 2.10 Use higher-order functions and lambda expressions in

		working programs 2.11 Describe and use closures in working programs 2.12 Explain and use currying and partial application of functions 2.13 Describe and use common higher-order functions such as map, reduce, filter, fold 2.14 Identify and use FP concepts in both FP and non-FP languages
	Course Outcome 3	**Learning Objectives for Course Outcome 3**
	3. Describe and use advanced type system concepts	3.1 Discuss the dis/advantages of strong vs weak and static vs dynamic type systems 3.2 Define type inference 3.3 Describe and use algebraic data types, specifically product, sum, intersection, and union types 3.4 Discuss how type systems can help prevent certain kinds of errors 3.5 Discuss pattern matching and how it can be used to achieve polymorphism in functional code 3.6 Write programs using languages of various type systems

Evaluation Process and Grading System:

Evaluation Type	Evaluation Weight
Lab Assignments	40%
Quizzes	10%
Test 1	25%
Test 2	25%

Date: June 16, 2022

Addendum: Please refer to the course outline addendum on the Learning Management System for further information.