COMS 215 - Programming Concepts and Methodology I Course Outline

Approval Date: 04/23/2020
Effective Date: 08/13/2021

SECTION A

Unique ID Number	CCC000100645
Discipline(s)	Computer Science
Division	Career Education and Workforce Development
Subject Area	Computer Studies
Subject Code	COMS
Course Number	215
Course Title	Programming Concepts and Methodology I
TOP Code/SAM Code	0707.10 - Computer Programming/Programmer, General* / D - Possible Occupational
Rationale for adding this course to the curriculum	Non-substantive modifications to Course Description, PreRequisite changed to Recommended Prep and Course Content. Addition of key terms from C-ID Descriptor.
Units	3
Cross List	*N/A*
Typical Course Weeks	18

Total Instructional Hours

	Contact Hours
Lecture	54.00
Lab	18.00
Activity	0.00
Work Experience	0.00
Outside of Class Hours	108.00
Total Contact Hours	72
Total Student Hours	180

Open Entry/Open Exit	No
Maximum Enrollment	30
Grading Option	Letter Grade or P/NP

Distance Education Mode of Instruction On-Campus
Hybrid
Entirely Online

SECTION B

General Education Information:

SECTION C

Course Description

Repeatability May be repeated 0 times

Catalog Description This is an introductory course to the fundamental concepts of computer science. Students will be exposed to a high level programming theories and methodologies, including object-oriented programming.

Schedule Description

SECTION D

Condition on Enrollment

1a. Prerequisite(s): *None*

1b. Corequisite(s): *None*

1c. Recommended

- COMS 120

1d. Limitation on Enrollment: *None*

SECTION E

Course Outline Information

1. Student Learning Outcomes:

A. Design, code, test, and debug a program using an object oriented programming language.

2. Course Objectives: Upon completion of this course, the student will be able to:

A. At the conclusion of this course, the student should be able to: Design, implement, test, and debug a program that uses each of the following fundamental programming constructs: basic computation, simple I/O, standard conditional and iterative structures, and the definition of functions.
B. Use pseudocode or a programming language to implement, test, and debug algorithms for solving simple problems.
C. Summarize the evolution of programming languages illustrating how this history has led to the paradigms available today.
D. Demonstrate different forms of binding, visibility, scoping, and lifetime management.
E.

3. Course Content

I. Programming Fundamentals (PF)

PF1. Fundamental programming constructs

Minimum coverage time: 9 hours

Topics

A. Basic syntax and semantics of a higher-level language
B. Variables, types, expressions, and assignment
C. Simple I/O

D. Conditional and iterative control structures
E. Functions and parameter passing
F. Structured decomposition

Learning Outcomes

A. Analyze and explain the behavior of simple programs involving the fundamental programming constructs covered by this unit;
B. Modify and expand short programs that use standard conditional and iterative control structures and functions;
C. Design, implement, test, and debug a program that uses each of the following fundamental programming constructs: basic computation, simple I/O, standard conditional and iterative structures, and the definition of functions;
D. Choose appropriate conditional and iteration constructs for a given programming task;
E. Apply the techniques of structured (functional) decomposition to break a program into smaller pieces; and
F. Describe the mechanics of parameter passing.

PF2. Algorithms and problem-solving
Minimum coverage time: 6 hours

Topics

A. Problem-solving strategies
B. The role of algorithms in the problem-solving process
C. Implementation strategies for algorithms
D. Debugging strategies
E. The concept and properties of algorithms

Learning outcomes

A. Discuss the importance of algorithms in the problem-solving process;
B. Identify the necessary properties of good algorithms;
C. Create algorithms for solving simple problems;
D. Use pseudocode or a programming language to implement, test, and debug algorithms for solving simple problems; and
E. Describe strategies that are useful in debugging.

II. Programming Languages (PL)
PL1. Overview of programming languages
Minimum coverage time: 2 hours

Topics

A. History of programming languages
B. Brief survey of programming paradigms
C. Procedural languages
D. Object-oriented languages

Learning outcomes

A. Summarize the evolution of programming languages illustrating how this history has led to the paradigms available today; and
B. Identify at least one distinguishing characteristic for each of the programming paradigms covered in this unit.

PL4. Declarations and types
Minimum coverage time: 3 hours

Topics

A. The conception of types as a set of values together with a set of operations Declaration models (binding, visibility, scope, and lifetime)
B. Overview of type-checking

Learning outcomes

A. Explain the value of declaration models, especially with respect to programming-in-the-large;
B. Identify and describe the properties of a variable such as its associated address, value, scope, persistence, and size;
C. Discuss type incompatibility;
D. Demonstrate different forms of binding, visibility, scoping, and lifetime management; and
E. Defend the importance of types and type-checking in providing abstraction and safety.

Topics fulfilling these tasks and outcomes could include OOPS and other programming elements. This course is recommended to contain hands-on programming and problem solving tasks.

4. Methods of Instruction:

Discussion: Discuss the history of Object Oriented programming.
Lab:
Lecture: Example: Lecture on the history of Object Oriented programming.
Projects: Create a programming language to complete a task in your day-to-day life. Test and Debug program, and present it to the class.
Online Adaptation: Activity, Discussion, Group Work, Lecture

4. Methods of Evaluation: Describe the general types of evaluations for this course and provide at least two, specific examples.

Typical classroom assessment techniques

Exams/Tests -- Example test questions: Essay: Provide a brief history of C++. Fill-in: What is a, "Loop" as it relates to C++.
Quizzes -- Multiple choice questions based on chapter reading.
Projects --
Class Participation --
Class Work --
Home Work --
Lab Activities -- Programming assignments

Final Exam --
Mid Term --

Additional assessment information:

Example test questions:
Essay: Provide a brief history of C++.
Fill-in: What is a, "Loop" as it relates to C++.

Letter Grade or P/NP

5. Assignments: State the general types of assignments for this course under the following categories and provide at least two specific examples for each section.

A. Reading Assignments
Read chapter 1 Getting Started.
Read chapter 2 Data Types, Declarations, and Displays.

B. Writing Assignments
Complete case problems 1 and 2 at the end of chapter 1.
Complete case problems 1-3at the end of chapter 2.

C. Other Assignments
Complete Tutorial 1 in the textbook.
Complete Tutorial 2 in the textbook.

6. Required Materials

A. EXAMPLES of typical college-level textbooks (for degree-applicable courses) or other print materials.

Book #1:	
Author:	Deitel, P.
Title:	C++ How to Program (Early Objects Version)
Publisher:	Pearson
Date of Publication:	2014
Edition:	9

Book #2:	
Author:	Gary J. Bronson
Title:	A FIRST BOOK OF C++
Publisher:	Cengage
Date of Publication:	2012
Edition:	

Book #3:	
Author:	D. S. Malik
Title:	Bundle: C++ Programming: From Problem Analysis to Program Design, Loose-leaf Version, 8th + MindTap Computer Science
Publisher:	Cengage Learning
Date of Publication:	2017
Edition:	

B. Other required materials/supplies.