CS 007A: COMPUTER SCIENCE I

Date Submitted: Mon, 08 Jun 2020 16:15:27 GMT

Originator
ghagopian

Justification / Rationale
Essentially, we need to swap SLOs and Course Objectives, as they are reversed in C-ID.

Effective Term
Spring 2020

Credit Status
Credit - Degree Applicable

Subject
CS - Computer Science

Course Number
007A

Full Course Title
Computer Science I

Short Title
COMPUTER SCIENCE I

Discipline

Disciplines List
Computer Science

Modality
Face-to-Face

Catalog Description
This course is an introduction to computer programming and is designed primarily for computer science and related transfer majors. Its main objective is to teach principles and practices of computer science, but students will also engage in problem solving using the C++ programming language. Topics include structured procedural programming with program control structures (sequence, selection, iteration), modular program structures (functions and parameter passing), data types (primitive types, arrays, files and structures) and an intro to object-oriented programming.

Schedule Description
An introduction to computer programming using the C++ language for students with no programming experience. Prerequisite: MATH 005

Lecture Units
3

Lecture Semester Hours
54

Lab Units
1

Lab Semester Hours
54

In-class Hours
108

Out-of-class Hours
108

Total Course Units
4
Total Semester Hours
216

Prerequisite Course(s)
MATH 005

Required Text and Other Instructional Materials

Resource Type
Book

Author
Stroustrup, Bjarne

Title
Programming Principles and Practices

Edition
2nd

Publisher
Addison-Wesley

Year
2014

College Level
Yes

ISBN #
978-0-321-99278-9

For Text greater than five years old, list rationale:
The Stroustrup text is, of this writing, 5 years old, but it's written by the Creator of C++ and is written as a text for an introductory course in C++.

Class Size Maximum
35

Entrance Skills
Demonstrate proficiency with polynomial analysis including the fundamental theorem of algebra and its implications.

Requisite Course Objectives
MATH 005-Use the concept of a function by identifying and describing a function graphically, numerically and algebraically.
MATH 005-Apply the six basic transformations of functions to graph translated functions, including the quadratic functions.
MATH 005-Represent a word problem (especially a geometric problem) with a function, including the use of functions to model real world applications.
MATH 005-Graph the six trigonometric functions and demonstrate the ability to predict the corresponding graphic behavior of changes in parameters that modify amplitude, period, and phase.

Entrance Skills
Demonstrate proficiency with analysis of trigonometry functions, including amplitude, periodicity, phase shifts, and basic identities.

Requisite Course Objectives
MATH 005-Apply the six basic transformations of functions to graph translated functions, including the quadratic functions.
MATH 005-Graph the six trigonometric functions and demonstrate the ability to predict the corresponding graphic behavior of changes in parameters that modify amplitude, period, and phase.

MATH 005-Use trigonometric functions to model periodic behavior.

Entrance Skills

Demonstrate proficiency with a wide variety of mathematical relations, including rational functions, root functions, symmetric functions, and the quadratic relations used in modeling shifted conics.

Requisite Course Objectives

MATH 005-Use the concept of a function by identifying and describing a function graphically, numerically and algebraically.
MATH 005-Apply the six basic transformations of functions to graph translated functions, including the quadratic functions.
MATH 005-Determine when a function has an inverse (one to one functions) and find the inverse function graphically or algebraically.

Entrance Skills

Solve word problems leading to systems of linear and/or non-linear equations in 2 or more variables using methods of elimination and substitution.

Requisite Course Objectives

MATH 005-Represent a word problem (especially a geometric problem) with a function, including the use of functions to model real world applications.

Course Content

I. Programming Fundamentals (PF) PF1. Fundamental programming constructs Minimum coverage time: 9 hours

Topics

1. Basic syntax and semantics of a higher-level language
2. Variables, types, expressions, and assignment
3. Simple I/O
4. Conditional and iterative control structures
5. Functions and parameter passing
6. Structured decomposition

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

Topics

1. Problem-solving strategies
2. The role of algorithms in the problem-solving process
3. Implementation strategies for algorithms
4. Debugging strategies
5. The concept and properties of algorithms

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

Topics

1. History of programming languages
2. Brief survey of programming paradigms
3. Procedural languages
4. Object-oriented languages

PL4. Declarations and types Minimum coverage time: 3 hours

Topics

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

Lab Content

1. Complete programming assignments incorporating design elements and code.
2. Consult with the teacher and classmates in small groups to tackle special programming tasks that arise as part of (1), above.

Course Objectives

	Objectives
Objective 1	Demonstrate effective use of a program development environment by entering/editing and executing programs.
Objective 2	Demonstrate mastery of the conception of types as a set of values together with a set of operations on those values.
Objective 3	Use the basic syntax and semantics of C++ to declare and define variables of specific types, and to form expressions, and assign these expressions to other variables of compatible types using simple Input/Output with conditional and iterative control structures.
Objective 4	Design functions using structured decomposition/modularization and pseudo-code with the "check/expect" model of development that checks that all input expressions evaluate to the value of the expected output expression.
Objective 5	Synthesize problem-solving strategies to design algorithms implementing step-wise refinement for improving algorithmic efficiency and correctness by debugging.
Objective 6	Interpret declaration models and the binding, visibility, scope, and lifetime of variables, especially by distinguishing "pass by value" and "pass by reference" function parameters.
Objective 7	Compare and contrast features of different programming languages by conducting a brief survey of programming paradigms: procedural, object-oriented and functional.
Objective 8	Employ with deliberate effect scalar data including int, double, char, boolean and string types in one and two-dimensional arrays.
Objective 9	Adhere to the Association of Computing Machine's Code of Ethics and Professional Conduct principles of contributing to human well-being, avoiding harm and behaving in an honest and trust-worthy manner and to maintain high standards of professional competence, conduct, and ethical practice.

Student Learning Outcomes

	Upon satisfactory completion of this course, students will be able to:
Outcome 1	Write and predict the results of code using standard input and output, with files, and test and debug such programs.
Outcome 2	Write and predict the results of code with numeric and Boolean expressions, if-statements and loops, including nested control structures.
Outcome 3	Design, write and predict the results of code using functions that have parameters (both call by reference and call by value) and return values.
Outcome 4	Design and develop code to process arrays.

Methods of Instruction

Method	Please provide a description or examples of how each instructional method will be used in this course.
Laboratory	Students will practice developing the design principles introduced in lecture by writing programs that solve problems of varying difficulty. Typically, students may be assigned to work either individually or in small groups to address the problem of writing code to accept input in a specific format and analyze that input produce a desired output.
Lecture	Programming design practices and principles are introduced in concept and by example.
Collaborative/Team	Take turns role-playing as designer/tester/developer in solving programming challenges to produce software meeting prescribed input/output specification.

Methods of Evaluation

Method	Please provide a description or examples of how each evaluation method will be used in this course.	Type of Assignment
Mid-term and final evaluations	There will be a midterm and a final exam, generally in written format, but this may be combined with some computer work.	In Class Only
Tests/Quizzes/Examinations	There will be a sequence of short quizzes to gauge student understanding of new concepts.	In and Out of Class

Group activity participation/observation	Three or more major projects encompassing at least two weeks for development of complex solutions to complex tasks. Typical problems involve assignments such as solving triangles, investigating the behavior of generalizations of the Collatz conjecture, generalizing the Babylonian algorithm for computing nth roots in the complex plane, exploring knight's tours on toroidal chess board, simulating the Game of Life and writing a calculator that will parse arithmetic and/or algebraic expressions.	In and Out of Class
Computational/problem-solving evaluations	Branching and looping structures are employed in algorithms such as that logistic iteration x(n+1) = cx(n)(1-x(n)), testing for bifurcation values, etc.	In and Out of Class
Laboratory projects	Often these will require students to solve problems from the ICPC (International Collegiate Programming Contest) while using concepts introduced in lecture.	In Class Only

Assignments

Other In-class Assignments

1. Take quizzes.
2. Take tests.
3. Participate in discussion.
4. Develop original programs to solve given problems.

Other Out-of-class Assignments

1. Read the text.
2. Write descriptions of programs in pseudocode.
3. Finish unfinished lab work.
4. Take quizzes.

Grade Methods

Letter Grade Only

Comparable Transfer Course Information

University System

CSU

Campus

CSU San Bernardino

Course Number

CSCI 201

Course Title

Computer Science I

Catalog Year

2009

University System

UC

Campus

UC Santa Barbara

Course Number

CMPSC 16

Course Title

Problem Solving with Computers I

Catalog Year
2010

University System
UC
Campus
UC Riverside

Course Number
CS 10
Course Title
Intro to Computer Science for Science, Mathematics, and Engineering I

Catalog Year
2010

University System
UC
Campus
CSU Los Angeles

Course Number
COM SCI 31
Course Title
Introduction to Computer Science I

Catalog Year
2010

MIS Course Data

CIP Code
11.0701 - Computer Science.

TOP Code
070600 - Computer Science (transfer)

SAM Code
E - Non-Occupational

Basic Skills Status
Not Basic Skills

Prior College Level
Not applicable

Cooperative Work Experience
Not a Coop Course

Course Classification Status
Credit Course

Approved Special Class
Not special class

Noncredit Category
Not Applicable, Credit Course

Funding Agency Category
Not Applicable

Program Status
Program Applicable

Transfer Status
Transferable to both UC and CSU

General Education Status
Not applicable

Support Course Status
Course is not a support course

C-ID
COMP 122

Allow Audit
No

Repeatability
No

Materials Fee
No

Additional Fees?
No

Approvals

Curriculum Committee Approval Date
9/03/2019

Academic Senate Approval Date
9/12/2019

Board of Trustees Approval Date
10/31/2019

Chancellor's Office Approval Date
6/03/2020

Course Control Number
CCC000523776

Programs referencing this course
Engineering AS Degree (http://catalog.collegeofthedesert.eduundefined?key=24/)
Liberal Arts: Business and Technology AA Degree (http://catalog.collegeofthedesert.eduundefined?key=27/)
Mass Communication A.A. Degree (http://catalog.collegeofthedesert.eduundefined?key=273/)
Liberal Arts: Math and Science AA Degree (http://catalog.collegeofthedesert.eduundefined?key=29/)
Computer Science AS-T Degree (http://catalog.collegeofthedesert.eduundefined?key=35/)
Mathematics AS Degree (http://catalog.collegeofthedesert.eduundefined?key=68/)