COLLEGE OF THE DESERT

Course Code CS-007B

Course Outline of Record

1. Course Code: CS-007B
2. a. Long Course Title: Computer Science II
   b. Short Course Title: COMPUTER SCIENCE II
3. a. Catalog Course Description:
   This second course in computer science introduces more advanced topics in programming. Students will use modularity to develop solutions for larger-scale programming problems. Recursion, file processing, and object-oriented programming are implemented. This course will be taught using the C++ programming language.
   b. Class Schedule Course Description:
   Develop solutions for larger-scale programming problems using recursion, file processing, and object-oriented programming. This course uses C++.
   c. Semester Cycle (*if applicable*): *N/A*
   d. Name of Approved Program(s):
   - COMPUTER SCIENCE AS Degree and Transfer Preparation
4. Total Units: 3.00 Total Semester Hrs: 90.00
   Lecture Units: 2 Semester Lecture Hrs: 36.00
   Lab Units: 1 Semester Lab Hrs: 54.00
   Class Size Maximum: 28 Allow Audit: No
   Repeatability No Repeats Allowed
   Justification 0
5. Prerequisite or Corequisite Courses or Advisories:
   Course with requisite(s) and/or advisory is required to complete Content Review Matrix (CCForm1-A)
   Prerequisite: CS 007A and
   Prerequisite: MATH 012
6. Textbooks, Required Reading or Software: *(List in APA or MLA format.)*
   a. Bjarne Stroustrup, (2014). *Programming: Principles and Practice Using C++* (2nd/e). Pearson. ISBN: 978-03219927
   College Level: Yes
   Flesch-Kincaid reading level: 12
7. Entrance Skills: *Before entering the course students must be able:*
   a. Understand how to implement algorithmic elements such as decision parameters and loops in a high level programming language such as C++.
   - CS 007A - 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
   - CS 007A - Use pseudocode or a programming language to implement, test, and debug algorithms for solving simple problems

   b. Write pseudo-code to plan out the broad outlines of algorithmic development in solving a programming problem.
   - CS 007A - Use pseudocode or a programming language to implement, test, and debug algorithms for solving simple problems

   c. Understand how to test and debug a program using pencil and paper and debugging tools of an IDE.
   - CS 007A - 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. Design and debug programs to solve problems in math and science demonstrating mathematical ability at the precalculus level.

- CS 007A - 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
- MATH 012 - Demonstrate an understanding of rational exponents, roots and their corresponding properties by evaluating, and simplifying algebraic expressions involving these symbols.
- CS 007A - Summarize the evolution of programming languages illustrating how this history has led to the paradigms available today
- MATH 012 - Apply the basic properties of the real numbers and exponents to add, subtract, multiply, divide and factor algebraic expressions including expressions involving fractions and roots.
- CS 007A - Demonstrate different forms of binding, visibility, scoping, and lifetime management
- MATH 012 - Apply the properties of the real numbers and exponents to solve one variable algebraic equations including linear, quadratic, rational, and root equations.
- MATH 012 - Apply the properties of the real numbers to solve one variable inequalities including linear, quadratic and rational inequalities.
- MATH 012 - Translate word problems into one variable equations or inequalities and solve the problem.
- MATH 012 - Find and use the basic characteristics of a polynomial function such as end behavior and zeros to construct a graph and to find an equation for a polynomial function.
- MATH 012 - Perform arithmetic with the complex numbers and use the complex numbers to completely solve a quadratic equation.
- MATH 012 - Solve polynomial equations using factoring, polynomial division and basic results such as the Remainder and Factor theorems and the Fundamental Theorem of Algebra.
- MATH 012 - Model and solve word problems using polynomial functions.
- MATH 012 - Find and use the basic characteristics of a rational function such as domain, end behavior, intercepts, and asymptotes to construct a graph and to find an equation for a rational function.
- MATH 012 - Model and solve word problems using rational functions.
- MATH 012 - Find and use the basic characteristics of an exponential function such as domain, concavity, intercepts, asymptotes, and transformations, to construct a graph and to find an equation for an exponential function.
- MATH 012 - Use a constant growth (decay) factor or rate to write an equation for an exponential function.
- MATH 012 - Demonstrate an understanding of logarithms and their basic properties by evaluating and simplifying algebraic expressions involving logarithms
- MATH 012 - Find and use the basic characteristics of a logarithmic function such as domain, concavity, intercepts and asymptotes to construct a graph and to find an equation for a logarithmic function.
- MATH 012 - Solve logarithmic and exponential equations algebraically and estimate solutions graphically.
- MATH 012 - Model and solve word problems (especially involving growth and decay) using exponential and logarithmic functions.
- MATH 012 - Demonstrate an understanding of the sum and difference identities, the double angle identities and the half angle identities by using them to deduce other identities.
- MATH 012 - Demonstrate an understanding of the inverse trigonometric functions and their inverse properties by evaluating and simplifying expressions involving these functions.
- MATH 012 - Graph the basic 6 inverse trig functions using basic characteristics such as domain, intercepts and asymptotes.
- MATH 012 - Estimate the solutions to trigonometric equations using graphing.
- MATH 012 - Solve trigonometric equations using algebra.
- MATH 012 - Model and solve word problems involving periodic behavior using the trigonometric functions.
- MATH 012 - Analyze independently and set up application problems, thus applying problem solving technique to new situations. Demonstrate the ability to anticipate and check their proposed solutions.
- MATH 012 - Communicate effectively with the instructor and mathematical community using proper terminology verbally as well as proper written notation.

8. Course Content and Scope:

Lecture:

A. Review topics presented in CS-007A such as
   1. Control structures (sequence, selection, iteration)
   2. Data structures such as arrays
   3. Declaring and defining functions
   4. Sorting and searching algorithms
B. Arrays
   1. Manipulating arrays with pointers as well as traditional array notation
   2. Arrays of objects
   3. Arrays of arrays, i.e. multi-dimensional arrays
   4. Character arrays
C. Structs
   1. Using structs to implement records
   2. Passing structs to functions by value
   3. Passing structs to functions by reference
D. Introduction to file handling
   1. Introduction to sequential access
   2. Introduction to random access
E. Pointers
   1. Using pointers for parameter passing
   2. Using pointers and pointer arithmetic to manipulate arrays, including character arrays
   3. Using pointers for dynamic memory allocation
   4. Using pointers to manipulate data structures such as vectors
F. Dynamic Memory Allocation (DMA)
   1. New
   2. Delete
G. Building abstract data types using classes
   1. Public verses private members
   2. Member functions and friend functions
   3. Constructors including the copy constructor and destructor
   4. Overloading the assignment operator
   5. Overloading operators using member functions and friend functions
   6. Static data members and functions
   7. Inheritance
      a. Overriding member functions
      b. Virtual functions
      c. Early versus late binding
      d. Introduction to pure virtual functions and abstract base classes
   8. Polymorphism
H. Using classes from the Standard Template Library (STL)
   1. Using the Vector class
   2. Using the Iterator class
I. Using standard string objects
   1. Creating and manipulating string objects
   2. Comparing string objects with character arrays
J. Introduction to recursion
   1. Introduction to recursion
   2. Recursive versus iterative approach

Lab: *(if the "Lab Hours" is greater than zero this is required)*

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.

9. Course Student Learning Outcomes:

1.
Create programs which use standard C++ language features, including functions, arrays, arrays of arrays, pointers, pointer arithmetic, dynamic memory allocation, and structured data (structs).

2.
Design and implement an abstract data type using a class with member variables, member functions, constructors, and a destructor.

3.
Design and implement modular programs, created in appropriate .h and .ccp files, that use multiple classes with inheritance relationships, friend functions, friend classes, and operator overloading, using modern C++ language features and STL vectors and iterators where appropriate.

10. Course Objectives: *Upon completion of this course, students will be able to:*

a. Write programs that use each of the following data structures: arrays, records, strings, linked lists, stacks, queues, and hash tables
b. Implement, test, and debug simple recursive functions and procedures
c. Evaluate tradeoffs in lifetime management (reference counting vs. garbage collection)
d. Explain how abstraction mechanisms support the creation of reusable software components
e. Design, implement, test, and debug simple programs in an object-oriented programming language
f. Compare and contrast object-oriented analysis and design with structured analysis and design

11. Methods of Instruction: *(Integration: Elements should validate parallel course outline elements)*

a. Activity
b. Demonstration, Repetition/Practice
c. Discussion
d. Experiential
e. Lecture
f. Observation

12. Assignments: *(List samples of specific activities/assignments students are expected to complete both in and outside of class.)*

In Class Hours: 90.00
Outside Class Hours: 72.00

a. In-class Assignments

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

b. Out-of-class Assignments

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

13. Methods of Evaluating Student Progress: *The student will demonstrate proficiency by:*
    - College level or pre-collegiate essays
    - Written homework
    - Critiques
    - Portfolios
    - Term or research papers
    - Reading reports
    - Laboratory projects
    - Computational/problem solving evaluations
    - Group activity participation/observation
    - True/false/multiple choice examinations
    - Mid-term and final evaluations

14. Methods of Evaluating: Additional Assessment Information:

15. Need/Purpose/Rationale -- *All courses must meet one or more CCC missions.*
    PO-GE C4.b - Language & Rationality (Communication & Analytical Thinking)
    Raise questions and problems, formulating them clearly and precisely.
    Gather, assess, and interpret relevant information.
    Express solutions to complex problems using language and logic.
    Apply logical and critical thinking to solve problems; explain conclusions; and evaluate, support, or critique the thinking of others.
    IO - Critical Thinking and Communication
    Apply principles of logic to problem solve and reason with a fair and open mind.
    Apply standard conventions in grammar, mechanics, usage and punctuation.
    Conduct research, gather and evaluate appropriate information, organize evidence into oral and written presentation, using proper MLA, APA, and other discipline-specific formats to cite sources.
    Utilizing various communication modalities, display creative expression, original thinking, and symbolic discourse.

16. Comparable Transfer Course

University System	Campus	Course Number	Course Title	Catalog Year
CSU	CSU San Bernardino	CSE 202	Computer Science II	2010
CSU	California Polytechnic University, Pomona	CS 256	C++ PROGRAMMING	2010
CSU	California Polytechnic University, San Luis Obispo	CSC 102	Fundamentals of Computer Science II	2010
UC	UC Riverside	CS 12	Intro to Computer Science for Science, Mathematics, and Engineering II	2010
UC	UCLA	COM SCI 32	Introduction to Computer Science II	2010
UC	UC Davis	ENG CS 40	Sftwr/Obj-Orien:C++	2010

17. Special Materials and/or Equipment Required of Students:

18. Materials Fees: ☐ Required Material?

Material or Item	Cost Per Unit	Total Cost

19. Provide Reasons for the Substantial Modifications or New Course:

ADT requirement. Align with C-ID COMP 152

20. a. Cross-Listed Course *(Enter Course Code)*: *N/A*
    b. Replacement Course *(Enter original Course Code)*: *N/A*

21. Grading Method *(choose one)*: Letter Grade Only

22. MIS Course Data Elements
    a. Course Control Number [CB00]: CCC000587414
    b. T.O.P. Code [CB03]: 70600.00 - Computer Science (Transfe
    c. Credit Status [CB04]: D - Credit - Degree Applicable
    d. Course Transfer Status [CB05]: A = Transfer to UC, CSU
    e. Basic Skills Status [CB08]: 2N = Not basic skills course
    f. Vocational Status [CB09]: Clearly Occupational
    g. Course Classification [CB11]: Y - Credit Course
    h. Special Class Status [CB13]: N - Not Special
    i. Course CAN Code [CB14]: *N/A*
    j. Course Prior to College Level [CB21]: Y = Not Applicable
    k. Course Noncredit Category [CB22]: Y - Not Applicable
    l. Funding Agency Category [CB23]: Y = Not Applicable
    m. Program Status [CB24]: 1 = Program Applicable

    Name of Approved Program *(if program-applicable)*: COMPUTER SCIENCE
    Attach listings of Degree and/or Certificate Programs showing this course as a required or a restricted elective.)

23. Enrollment - Estimate Enrollment
    First Year: 28
    Third Year: 28

24. Resources - Faculty - Discipline and Other Qualifications:
    a. Sufficient Faculty Resources: Yes
    b. If No, list number of FTE needed to offer this course: *N/A*

25. Additional Equipment and/or Supplies Needed and Source of Funding.

    N/A

26. Additional Construction or Modification of Existing Classroom Space Needed. *(Explain:)*

    N/A

27. FOR NEW OR SUBSTANTIALLY MODIFIED COURSES
    Library and/or Learning Resources Present in the Collection are Sufficient to Meet the Need of the Students Enrolled in the Course: Yes

28. Originator Geoffrey Hagopian Origination Date 09/05/17