El Camino College

COURSE OUTLINE OF RECORD - Official

I. GENERAL COURSE INFORMATION

Subject and Number: Computer Science 2
Descriptive Title: Introduction to Data Structures

Course Disciplines: Computer Science

Division: Mathematical Sciences

Catalog Description: In this course, the C++ computer language is used to demonstrate methods of representing and manipulating data structures. The student will learn the object-oriented problem solving skills necessary to read, write, and correct complex computer programs, and to make important design decisions. Topics include lists, stacks, queues, trees, searching, sorting, modeling and algorithm analysis.

Conditions of Enrollment:

Prerequisite

Computer Science 1
with a minimum grade of C

Recommended Preparation

Mathematics 190

Course Length: [X] Full Term [ ] Other (Specify number of weeks):
Hours Lecture: 4.00 hours per week [ ] TBA
Hours Laboratory: 3.00 hours per week [ ] TBA
Course Units: 5.00

Grading Method: Letter
Credit Status Associate Degree Credit

Transfer CSU: [X] Effective Date: Prior to July 1992
Transfer UC: [X] Effective Date: Prior to July 1992

General Education:
El Camino College: _____
CSU GE: _____
IGETC: _____

II. OUTCOMES AND OBJECTIVES

A. COURSE STUDENT LEARNING OUTCOMES (The course student learning outcomes are listed below, along with a representative assessment method for each. Student learning outcomes are not subject to review, revision or approval by the College Curriculum Committee)

1. **SLO #1 Programming Solutions**
   Students will design, code, compile, test, and document a programming solution to a problem involving the basic data structures: lists, stacks, queues, trees, and related abstract data types.

2. **SLO #2 Output of Program Segments**
   Students, when given a C++ code segment, will be able to trace the execution, give the output, and analyze the efficiency of the basic data structures and techniques involved.

3. **SLO #3 Correcting Errors**
   Students, when given a C++ code segment with errors, will be able to identify and correct the problems.

4. **SLO #4 Explaining C++ Concepts**
   Students will be able to explain the C++ concepts related to pointers, classes, recursion, searching, sorting, templates and dynamic memory allocation.

The above SLOs were the most recent available SLOs at the time of course review. For the most current SLO statements, visit the El Camino College SLO webpage at http://www.elcamino.edu/academics/slo/.

B. Course Student Learning Objectives (The major learning objective for students enrolled in this course are listed below, along with a representative assessment method for each)

1. Design, code, compile, test and document a programming solution to a specified problem requiring basic data structures.
   Homework Problems
2. Trace the execution and give the output of a given program or program segment pertaining to data structures.
   Objective Exams
3. Identify and correct the errors in a given program or program segment pertaining to data structures.
   Objective Exams
4. Implement and explain the concepts underlying the basic data structures: lists, stacks, queues, trees, and related abstract data types.
   Objective Exams
5. Explain and implement basic data structure techniques: pointers, classes, recursion, searching, sorting, templates and dynamic memory allocation.
   Objective Exams
6. Analyze the efficiency of the basic data structures and techniques.
   Objective Exams

III. OUTLINE OF SUBJECT MATTER (Topics are detailed enough to enable a qualified instructor to determine the major areas that should be covered as well as ensure consistency from instructor to instructor and semester to semester.)

Lecture or Lab	Approximate Hours	Topic Number	Major Topic
Lecture	6	I	Pointers A. Pointer variables. B. Pointers to objects C. Dynamic memory allocation. D. Pointer Arithmetic.
Lab	6	II	Applications of Pointers A. Pointer variables. B. Pointers to objects C. Dynamic memory allocation. D. Pointer Arithmetic.
Lecture	8	III	Built-in C++ data structures A. Strings. B. Static Arrays. C. Dynamic Arrays. D. Templates. E. Abstract Data Types. F. Information Hiding. G. Addressing. H. Numeric base representations including binary and hexadecimal.
Lab	6	IV	Applications of Built-in C++ data structures A. Strings. B. Static Arrays. C. Dynamic Arrays. D. Templates. E. Abstract Data Types. F. Information Hiding. G. Addressing. H. Numeric base representations including binary and hexadecimal.
Lecture	8	V	Classes A. Accessing class members. B. Class Scope. C. Constructors and destructors. D. Static members E. Implementing member functions. F. Overloading operators and functions. G. Data abstraction H. Inner/nested classes

Lab	6	VI	Applications of Classes A. Accessing class members. B. Class Scope. C. Constructors and destructors. D. Static members E. Implementing member functions. F. Overloading operators and functions. G. Data abstraction H. Inner/nested classes
Lecture	8	VII	Linked Lists A. Implementation using both static and dynamic techniques. B. Unordered vs Ordered linked lists. C. Variations including doubly linked and circular linked lists. D. Efficiency considerations. E. Templated Linked Lists
Lab	6	VIII	Applications of Linked Lists A. Implementation using both static and dynamic techniques. B. Unordered vs Ordered linked lists. C. Variations including doubly linked and circular linked lists. D. Efficiency considerations. E. Templated Linked Lists
Lecture	6	IX	Stacks A. Implementation and manipulation using arrays. B. Implementation and manipulation using linked structures. C. Efficiency considerations.
Lab	6	X	Applications of Stacks A. Implementation and manipulation using arrays. B. Implementation and manipulation using linked structures. C. Efficiency considerations.
Lecture	6	XI	Queues A. Implementation and manipulation using arrays. B. Implementation and manipulation using linked structures. C. Priority Queues and other variations. D. Efficiency considerations.
Lab	6	XII	Applications of Queues A. Implementation and manipulation using arrays.

			B. Implementation and manipulation using linked structures. C. Priority Queues and other variations. D. Efficiency considerations.
Lecture	8	XIII	Recursion A. Direct recursion. B. Indirect recursion. C. Recursion vs iteration. D. Backtracking. E. Applications to other data structures. F. Efficiency considerations.
Lab	6	XIV	Applications of Recursion A. Direct recursion. B. Indirect recursion. C. Recursion vs iteration. D. Backtracking. E. Applications to other data structures. F. Efficiency considerations
Lecture	8	XV	Trees A. General trees. B. Binary trees. C. Binary search trees. D. Implementation and manipulation using arrays. E. Implementation and manipulation using linked structures. F. Heaps. G. Tree variations. H. Efficiency considerations.
Lab	6	XVI	Applications of Trees A. General trees. B. Binary trees. C. Binary search trees. D. Implementation and manipulation using arrays. E. Implementation and manipulation using linked structures. F. Heaps. G. Tree variations. H. Efficiency considerations.
Lecture	8	XVII	Sorting A. Bubble Sort. B. Selection Sort. C. Insertion Sort. D. Mergesort.

			E. Quicksort. F. Heapsort. G. Radix sort. H. Variations in implementation. I. Algorithm analysis including Big-O Notation.
Lecture	6	XVIII	Searching A. Sequential search. B. Binary search. C. Hash functions and tables. D. Variations in implementation. E. Efficiency considerations.
Lab	6	XIX	Applications of Searching and Sorting A. Sequential search. B. Binary search. C. Hash functions and tables. D. Variations in implementation. E. Efficiency considerations. F. Bubble Sort. G. Selection Sort. H. Insertion Sort. I. Mergesort. J. Quicksort. K. Heapsort. L. Radix sort. M. Variations in implementation. N. Algorithm analysis including Big-O Notation.
Total Lecture Hours		72	
Total Laboratory Hours		54	
Total Hours		126	

IV. PRIMARY METHOD OF EVALUATION AND SAMPLE ASSIGNMENTS

A. **PRIMARY METHOD OF EVALUATION:**

Problem solving demonstrations (computational or non-computational)

B. **TYPICAL ASSIGNMENT USING PRIMARY METHOD OF EVALUATION:**

Wing-and-a-Prayer Airlines maintains four scheduled flights per day, which they identify by the numbers 1, 2, 3, and 4. For each of these flights, they keep an alphabetized list of passengers. The database for the entire airline could hence be viewed as four linked lists. Write a program that sets up and maintains this

database by handling commands of the following form:

Command --> Add
Flight Number --> 3
Passenger Name --> BROWN
Command --> Delete
From Flight Number --> 1
Passenger Name --> JONES
Command --> List
Flight Number --> 2
(List alphabetically all passengers for the specified flight.)

Use an appropriate string storage strategy for the passenger names.

C. **COLLEGE-LEVEL CRITICAL THINKING ASSIGNMENTS:**

1. Write a program that analyzes a text file by counting the number of times each word appears in the text, storing the words and their counts in a binary search tree, and then displaying the data in an output file.
2. Using the string, vector, and set classes, write a computer program to locate all files on a specified drive or directory that share common file-names. Present the results as a sorted list of the distinct file-names, and accompanying each file-name, specify the sorted block of path-names for that file.

D. **OTHER TYPICAL ASSESSMENT AND EVALUATION METHODS:**

Other exams
Quizzes
Homework Problems
Other (specify):
Documentation of code - the student will explain each part of their code clearly and concisely.

Creation of computer programs.

V. INSTRUCTIONAL METHODS

Demonstration
Laboratory
Lecture

Note: In compliance with Board Policies 1600 and 3410, Title 5 California Code of Regulations, the Rehabilitation Act of 1973, and Sections 504 and 508 of the Americans with Disabilities Act, instruction delivery shall provide access, full inclusion, and effective communication for students with disabilities.

VI. WORK OUTSIDE OF CLASS

Study

Required reading

Problem solving activities

Estimated Independent Study Hours per Week: 8

VII. TEXTS AND MATERIALS

A. **UP-TO-DATE REPRESENTATIVE TEXTBOOKS**
Nell Dale. C++ Data Structures. 5th ed. Jones and Bartlett Publishing, 2013.

B. **ALTERNATIVE TEXTBOOKS**

C. **REQUIRED SUPPLEMENTARY READINGS**

D. **OTHER REQUIRED MATERIALS**

VIII. CONDITIONS OF ENROLLMENT

A. Requisites (Course and Non-Course Prerequisites and Corequisites)

Requisites	Category and Justification
Course Prerequisite Computer Science-1	Sequential

B. Requisite Skills

Requisite Skills
1. Define simple software engineering terminology, such as software life cycle, problem analysis, program design, testing and top-down design. Utilize software engineering terminology properly when describing program design. CSCI 1 - Define simple software engineering terminology, such as software life cycle, problem analysis, program design, testing and top-down design. Utilize software engineering terminology properly when describing program design.
2. Utilize problem analysis and design techniques in developing solutions to programming problems. In particular, break programming problems down into chunks, leading to efficient use of top-down design in order to create and implement modular solutions. CSCI 1 - Utilize problem analysis and design techniques in developing solutions to programming problems. In particular, break programming problems down into chunks, leading to efficient use of top-down design in order to create and implement modular solutions.
3. Represent data utilizing simple numeric and character data types in a program and use them with input-output processes of the particular implementation of C++ being used. CSCI 1 - Represent data utilizing simple numeric and character data types in a program and use them with input-output processes of the particular implementation of C++ being used.
4. Design solutions requiring translation of mathematical and algebraic steps into a C++ program, using appropriate mathematical operators and math library functions of the C++ implementation in use. CSCI 1 -

Design solutions requiring translation of mathematical and algebraic steps into a C++ program, using appropriate mathematical operators and math library functions of the C++ implementation in use.
5. Design programming solutions requiring decision-making, using appropriate C++ selection statements, such as if-then, if-then-else and switch. CSCI 1 - Design programming solutions requiring the use of repeated processes, using appropriate C++ iteration statements, such as for, while and do-while loops.
6. Design programming solutions requiring the use of repeated processes, using appropriate C++ iteration statements, such as for, while and do-while loops. CSCI 1 - Design programming solutions requiring the use of repeated processes, using appropriate C++ iteration statements, such as for, while and do-while loops.
7. Design programming solutions requiring the storage and manipulation of large amounts of data (with random access ability during execution cycle), using single and multi-dimensional arrays, such as numerical, string, char, and structure types. CSCI 1 - Design programming solutions requiring the storage and manipulation of large amounts of data (with random access ability during execution cycle), using single and multi-dimensional arrays, such as numerical, string, char, and structure types.

C. Recommended Preparations (Course and Non-Course)

Recommended Preparation	Category and Justification
Course Recommended Preparation Mathematics-190	

D. Recommended Skills

Recommended Skills
Computational, logical and communication skills. MATH 190 - Estimate definite integrals using Riemann sums. MATH 190 - Use computing software to solve calculus problems.

E. Enrollment Limitations

Enrollment Limitations and Category	Enrollment Limitations Impact

Course created by David Akins on 11/01/1982.

BOARD APPROVAL DATE:

LAST BOARD APPROVAL DATE: 07/18/2016

Last Reviewed and/or Revised by Satish Singhal on 03/29/2016

19494