COURSE OUTLINE : CS/IS 211
D Credit – Degree Applicable
COURSE ID 005225
Cyclical Review: August 2020

COURSE DISCIPLINE : CS/IS

COURSE NUMBER : 211

COURSE TITLE (FULL) : Data Structures

COURSE TITLE (SHORT) : Data Structures

CALIFORNIA STATE UNIVERSITY SYSTEM C-ID : COMP 132 - Programming Concepts and Methodology II

CATALOG DESCRIPTION

CS/IS 211 is designed to provide a thorough coverage of data structures with data abstraction applied to a broad spectrum of practical applications. Students who take this course master the principles of programming as a tool for problem solving. The students solve practical problems in a computer equipped laboratory using an Object Oriented Programming (OOP) Language, typically JAVA or C++.

Total Lecture Units: 3.50

Total Laboratory Units: 0.50

Total Course Units: 4.00

Total Lecture Hours: 63.00

Total Laboratory Hours: 27.00

Total Laboratory Hours To Be Arranged: 0.00

Total Contact Hours: 90.00

Total Out-of-Class Hours: 126.00

Prerequisite: CS/IS 135 or equivalent.

ENTRY STANDARDS

	Subject	Number	Title	Description	Include
1	CS/IS	135	Programming In C/C++	Analyze a programming task to develop and communicate efficient algorithms to implement that task;	Yes
2	CS/IS	135	Programming In C/C++	recognize programming problems on a function-by-function basis and develop structured/procedural code based on this approach;	No
3	CS/IS	135	Programming In C/C++	demonstrate an understanding of object-oriented programming concepts and object-oriented design;	Yes
4	CS/IS	135	Programming In C/C++	program in the C++ language including use of objects, pointers, and structures.	Yes

EXIT STANDARDS

1 Create computer programs using data structures such as arrays, records, strings, linked lists, stacks, queues, and hash tables;
2 create simple recursive functions and procedures;
3 create simple programs in an object-oriented programming language;
4 compare and contrast object-oriented analysis and design with structured analysis and design.
5 explain basic abstract data types;
6 explain more complex abstract data types.

STUDENT LEARNING OUTCOMES

1 prepare more complex computer program problems using OOP principles;
2 explain how abstraction mechanisms aid in creating reusable software components;

COURSE CONTENT WITH INSTRUCTIONAL HOURS

	Description	Lecture	Lab	Total Hours
1	Principles of Programming and Software Engineering • Problem solving and software engineering • Key issues in programming • Modular design, parameters (references and value), and templates	5	1	6

2	Recursion • Recursive solutions • Counting and searching • Organizing data • Recursive solutions and mathematical functions • Backtracking, divide-and-conquer, and simple recursive procedures • Activation records and storage management	6	4	10
3	Data Abstraction • Basic data types • Primitive types, arrays, records, strings, pointers and references • Linked structures • Data representation in memory • Abstract data types • Defining types as a set of values with a set of operations • Specifying and implementing ADT	6	2	8
4	Linked lists • Linked structures • Dynamic memory allocation, deallocation, and garbage collection • Programming with linked lists • Variations of the linked list	6	2	8
5	Stacks • The abstract data type stack • Simple application of the ADT stack • Applications utilizing Postfix and Infix expressions • The relationship between stacks and recursion	6	2	8
6	Queues • The abstract data type queue • Simple applications of the ADT queue • Implementation of the ADT queue • Summary of position oriented ADTs	6	2	8
7	Class Relationships • Object-oriented design • Inheritance, encapsulation, and polymorphism • Dynamic binding and abstract classes • Classes, subclasses, and class hierarchies • Advantages of an objects-oriented approach	7	3	10
8	Algorithm Efficiency and Sorting • Determining the efficiency of algorithms • Sorting algorithms and their efficiency	4	2	6

9	Trees • Terminology • The ADT binary tree • The ADT binary search tree • General trees • Heaps • Implementation of trees	5	2	7
10	Tables and Priority Queues • The ADT table • The ADT priority queue	4	3	7
11	Advanced Implementation of Tables • Balanced search trees • Hashing • Implementation of hash tables • Data with multiple organizations	4	2	6
12	Graphs • Terminology • Graphs as ADT • Graph traversals • Applications of graphs	4	2	6
				90

OUT OF CLASS ASSIGNMENTS

1 homework exercises (e.g. create computer programs using data structures such as arrays, records, strings, linked lists, stacks, queues, and hash tables);
2 programming assignments (e.g. use recursion to solve simple problems).

METHODS OF EVALUATION

1 quizzes;
2 programming projects;
3 midterms;
4 final exam.

METHODS OF INSTRUCTION

- [x] Lecture
- [x] Laboratory
- [ ] Studio
- [ ] Discussion
- [ ] Multimedia
- [ ] Tutorial

- [ ] Independent Study
- [ ] Collaboratory Learning
- [x] Demonstration
- [ ] Field Activities (Trips)
- [ ] Guest Speakers
- [ ] Presentations

TEXTBOOKS

Title	Type	Publisher	Edition	Medium	Author	IBSN	Date
Data Abstraction and Problem Solving with C++: Walls and Mirrors.	Required	New York: Addison Welsey	7	Print	Carrano, Frank	978-013446397 1	2016