Syllabus

CSC 200 Cs3: Data Structures

General Information

Date

July 12th, 2018

Author

Sandra Brown

Department

Computing Sciences

Course Prefix

CSC

Course Number

200

Course Title

Cs3: Data Structures

Course Information

Credit Hours

4

Lecture Contact Hours

4

Lab Contact Hours

1

Other Contact Hours

Catalog Description

CS3: Data Structures covers the fundamentals of data structures, introduction to analysis of algorithms, and team development of software applications. This course is the third in a series of three required programming courses for a traditional computer science degree. Data structures covered include sets, lists, stacks, queues, linked lists, binary trees, and heaps. Advanced topics include, binary search trees, search and sort algorithms, recursion, and algorithm efficiencies in software development. Students will be introduced to project management and team dynamics through the development of a large software solution.

Key Assessment

This course does not contain a Key Assessment for any programs

Prerequisites

CSC 190

Co-requisites

None

Grading Scheme

Letter

First Year Experience/Capstone Designation

This course is designated as satisfying the outcomes applicable for status as a

Capstone Course

SUNY General Education

This course is designated as satisfying a requirement in the following SUNY Gen Ed category

None

FLCC Values

Institutional Learning Outcomes Addressed by the Course

Vitality
Inquiry
Perseverance
Interconnectedness

Course Learning Outcomes

Course Learning Outcomes

1. Identify and implement advanced data structures for the manipulation of data
2. Integrate multiple algorithms to form a complex computer solution
3. Analyze and evaluate the efficiencies of available data structures in order to select the appropriate solution for a given algorithm
4. Plan, prioritize, and build a complex computer solution within a collaborative work environment

Outline of Topics Covered

1. The Recursive method and the benefits of using recursion
2.

Benefits of generics

3. Explore the relationship between interfaces and classes in the Java Collections Framework hierarchy
4. Store unordered, nonduplicate elements using a set
5. Compare the performance of sets and lists
6. Estimate algorithm efficiency using Big O notation
7. Explain growth rates and why constants and nondominating terms can be ignored in an estimation
8. Determine the complexity of various types of algorithms
9. Describe common growth functions
10. Study and analyze time complexity of various sorting algorithms
11. Design and implement a linked list using a linked structure
12. Design and implement a stack class using an array list and a queue class using a linked list
13. Design and implement a priority queue using a heap
14. Design and implement a binary search tree
15. Analyze the complexity of search, insertion, and deletion operations in AVL trees
16. Understand what hashing is and what hashing is used for
17. Model real-world problems using graphs
18. Describe the graph terminologies:
    1. Vertices
    2. Edges
    3. Simple graphs
    4. Weighted/unweighted graphs
    5. Directed/undirected graphs
19.

Multithreading overview

20. Create threads to run tasks using the Thread class
21. Explain terms:
    1. TCP
    2. IP
    3. domain name
    4. domain name server
    5. stream-based communications
    6. packet-based communications
22. Develop an example of a client/server application
23. Work in a team to develop a software application
24. Understand professional responsibilities and liabilities associated with software development