Computer Science Department
cs.salemstate.edu

CSC 260 Data Structures and Algorithms **4 cr.**

Instructor: TBA **Office:** location **Phone:** (978) 542-extension
email: TBA@salemstate.edu **Office Hours:** days and times

Section	Time	Room	Final Exam
nn	days and times	location	date and time
Lnn	days and times	location	

Catalog description:

Basic data structures such as stacks, queues, linked lists, and trees are studied and applied to problems in data storage and manipulation. Applications include basic searching and sorting algorithms. Fundamental strategies for algorithm design are reviewed and extended. Design, analysis, implementation, and quality assurance techniques are discussed. Three lecture hours and three hours of scheduled laboratory per week, plus extensive programming work outside of class.

Prerequisites: CSC 105 and CSC 115.

Goals:

The purpose of this course is to develop students' knowledge and appreciation of organization and retrieval techniques and to familiarize students with the basic concepts of order-of-magnitude analysis. The goals of this course are:

CG01: to develop an appreciation for the process of data abstraction and its usefulness in software development;
CG02: to develop the skills and knowledge necessary to perform design and basic analysis of algorithms;
CG03: to present a selection of the most common data structures and their standard implementations and uses;
CG04: to present a selection of the most common algorithms for searching and sorting.

Objectives:

Upon successful completion of the course, student will have:

CO01: applied data abstraction techniques;
CO02: implemented several classic data structures "from scratch";
CO03: demonstrated knowledge and use of ADTs available in one or more language libraries;
CO04: recognized the factors required to perform algorithm design, analysis of algorithms and performed order-of-magnitude analysis;
CO05: chosen, with justification, an appropriate structure to match the requirements of a given problem, implemented the structure if necessary, and used it in an appropriate way to solve the problem;
CO06: utilized standard techniques for program validation;
CO07: demonstrated the ability to use the UML modeling language;
CO08: produced documentation for at least one major completed project, including formal class diagrams and rigorous test set specification and results.
CO09: participated in at least one group project involving problem analysis and design specification and selection
CO10: demonstrated recognition of the need for future professional development through research into future trends in the areas of analysis of algorithms and application development and profiling.

Student Outcome (SO) vs. Course Objectives matrix

SO	CO01	CO02	CO03	CO04	CO05	CO06	CO07	CO08	CO09	CO10
SO-1	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
SO-2		✓	✓	✓	✓	✓	✓	✓	✓	✓
SO-3								✓	✓	✓

SO	CO01	CO02	CO03	CO04	CO05	CO06	CO07	CO08	CO09	CO10
SO-4										✓
SO-5									✓	
SO-6	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Notes:

SO-1: Analyze a complex computing problem and to apply principles of computing and other relevant disciplines to identify solutions.

SO-2: Design, implement, and evaluate a computing-based solution to meet a given set of computing requirements in the context of the program's discipline.

SO-3: Communicate effectively in a variety of professional contexts.

SO-4: Recognize professional responsibilities and make informed judgments in computing practice based on legal and ethical principles.

SO-5: Function effectively as a member or leader of a team engaged in activities appropriate to the program's discipline.

SO-6: Apply computer science theory and software development fundamentals to produce computing-based solutions.

Topics:

- review of design concepts
- recursion (review and new examples)
- the hierarchy of data types and the concept of abstract data type (ADT)
- the three levels of data structuring **SDF1(2, 0, 0), SE5(3, 1 ,0)**
  - the application level (recognizing the behaviors and features needed to solve the problem at hand)
  - the abstract level (selecting or defining an appropriate abstraction that models these behaviors)
  - the implementation level (realizing the abstraction using standard programming language features)
- abstract data types **SDF3(4, 0, 0)**
  - stacks
  - queues
  - priority queues
  - ordered lists
  - access tables
  - links
  - trees
  - heaps
  - graphs
- fundamentals of algorithm design **SDF1(2, 0, 0)**
  - strategic approaches
    - Greedy
    - Divide and Conquer
- other algorithms (introduction): **AL2(1, 2, 0)**
  - linear search, binary search
  - insertion sort, selection sort
  - quicksort, heapsort
- data structures and their algorithms
  - linear linked structures: singly linked lists, bidirectional linked lists, multi-list structures **SDF3(10, 0, 0)**
  - non-linear linked structures **AL3(6, 2 ,0)**
    - hierarchical: binary trees, AVL trees, B-trees
    - network: graphs, digraphs, weighted graphs
  - direct access structures: hash tables (direct and indirect)
- ADTs and object-oriented design **SDF1(1, 0, 0)**
- elementary algorithm analysis (efficiency, speed) **AL1(2, 0, 0)**
- implementing ADTs and data structures: **SDF2(3,0,0), SDF3(5, 0, 0), PL4(2, 2, 0)**
  - static memory allocation (arrays) vs. dynamic memory allocation vs. files
  - pointers and dynamic memory allocation
- quality assurance **SDF4(1, 0, 0)**

- strategies for design of test data (test scenarios) based on
  - standard cases
  - edge/corner cases
- boundary conditions and the selection of test data to embody test scenarios
- use of software libraries **SE5(0, 2, 0)**

This course revolves around the notions of data abstraction and the structuring of data, using the concept of abstract data type (ADT). The most common and most useful data structures are defined and classified, and the appropriate manipulation algorithms are presented in general form (in pseudocode). At least one concrete realization for each structure is then discussed.

Programming Assignments: Five to six programming assignments are given, emphasizing the choice and/or implementation of a specified structure, such as a stack, queue, binary search tree, or hash table. The final assignment requires the student to make the choice of an appropriate data structure or combination of structures to best solve a specified problem.

All programs must conform to departmental guidelines for algorithm design and implementation. Laboratory reports must conform to the written guidelines supplied by the instructor. Regardless of numeric average or individual grades on assignments or examinations, a student will not be eligible for a passing grade in the course unless he or she has submitted a lab report for every programming assignment, within the time frame specified by the instructor.

Laboratory exercises: There will be short programming and design exercises to be completed during weekly scheduled laboratory sessions. Each exercise focuses on a language feature, programming technique or design technique presented in recent lectures. Performance on these exercises will be incorporated into the course grade.

Exams and quizzes: There will be two examinations and a comprehensive written two-hour final examination.

Final grades will be determined on the basis of the following approximate weights: examinations - 45%; programming assignments, lab exercises, homework - 55%.

Course Objective / Assessment Mechanism matrix

	Test / Quiz Questions	Homework Problems	Programming Projects	Lab Exercises	Group Projects
CO01	✓	✓	✓	✓	
CO02	✓	✓	✓	✓	
CO03	✓	✓	✓	✓	
CO04	✓	✓	✓	✓	
CO05	✓	✓	✓		✓
CO06		✓	✓	✓	
CO07		✓	✓	✓	✓
CO08	✓	✓	✓	✓	✓
CO09		✓	✓		✓
CO10		✓	✓		✓

Bibliography

Bloch, Joshua. **Effective Java. Third Edition.** Prentice Hall, 2018.

Carrano, Frank M.; Prichard, Janet J. **Data Abstraction and Problem Solving with Java: Walls and Mirrors. Third Edition.** Addison-Wesley, 2011.

Collins, William J. **Data Structures and the Java Collections Framework. Third Edition.** McGraw-Hill, 2011.

Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L. **Introduction to Algorithms. Third Edition.** The MIT Press, 2009.

Dale, Nell; Joyce, Daniel T.; Weems, Chip. **Object-Oriented Data Structures using Java. Fourth Edition.** Jones & Bartlett, 2016.

Goodrich, Michael T.; Tamassia, Roberto; Goldwasser Michael H.. **Data Structures and Algorithms in Java. Sixth edition**. John Wiley & Sons, 2014.

Horstmann, Cay S.; Cornell, Gary. **Core Java ™, Volume I—Fundamentals. Eighth Edition**. Prentice Hall, 2007.
Horstmann, Cay S.; Cornell, Gary. **Core Java ™, Volume 2—Advanced Features. Ninth Edition**. Prentice Hall, 2013.
Koffman, Elliot B. **Data Structures: Abstraction and Design Using Java, Third Edition.** Wiley, 2015.
Knuth, Donald E. **The Art of Computer Programming, Vol. 1. Third Edition.** Addison-Wesley, 1997.
Knuth, Donald E. **The Art of Computer Programming, Vol. 3. Second Edition.** Addison-Wesley, 1998.
Lewis, J.; Chase, J. **Java Software Structures: Designing and Using Data Structures. Fourth Edition.** Pearson Education, 2013.
Liang, Y. Daniel. **Introduction to Java Programming and Data Structures, Comprehensive Version. Eleventh Edition.** Pearson, 2017.
Naftalin, Maurice. **Java Generics and Collections. First Edition**. O'Reilly Media, 2006.
Weiss, Mark Allen. **Data Structures and Problem Solving Using Java. Fourth Edition.** Addison Wesley, 2010.

Academic Integrity Statement:

"Salem State University assumes that all students come to the University with serious educational intent and expects them to be mature, responsible individuals who will exhibit high standards of honesty and personal conduct in their academic life. All forms of academic dishonesty are considered to be serious offences against the University community. The University will apply sanctions when student conduct interferes with the University primary responsibility of ensuring its educational objectives." Consult the University catalog for further details on Academic Integrity Regulations and, in particular, the University definition of academic dishonesty.

The Academic Integrity Policy and Regulations can be found in the University Catalog and on the University website (http://catalog.salemstate.edu/content.php?catoid=13&navoid=1295#Academic_Integrity). The formal regulations are extensive and detailed - familiarize yourself with them if you have not previously done so. A concise summary of and direct quote from the regulations: "Materials (written or otherwise) submitted to fulfill academic requirements must represent a student's own efforts". *Submission of other's work as one's own without proper attribution is in direct violation of the University's Policy* and will be dealt with according to the University's formal Procedures. *Copying without attribution is considered cheating in an academic environment - simply put,* ***do not do it!***

University-Declared Critical Emergency Statement:

In the event of a university-declared emergency, Salem State University reserves the right to alter this course plan. Students should refer to www.salemstate.edu for further information and updates. The course attendance policy stays in effect until there is a university-declared critical emergency.

In the event of an emergency, please refer to the alternative educational plans for this course, which will be distributed via standing class communication protocols. Students should review the plans and act accordingly. Any required material that may be necessary will have been previously distributed to students electronically or will be made available as needed via email and/or Internet access.

Equal Access Statement:

"Salem State University is committed to providing equal access to the educational experience for all students in compliance with Section 504 of The Rehabilitation Act and The Americans with Disabilities Act and to providing all reasonable academic accommodations, aids and adjustments. **Any student who has a documented disability requiring an accommodation, aid or adjustment should speak with the instructor immediately.** Students with Disabilities who have not previously done so should provide documentation to and schedule an appointment with the Office for Students with Disabilities and obtain appropriate services."

Note: This syllabus represents the intended structure of the course for the semester. If changes are necessary, students will be notified in writing and via email.