Final Exam Review

Date: December 6, 2025

Day: Saturday

Time: 1:00 pm – 3:50 pm

Location: CB2-207 (Classroom Building 2)

Note: All sections of the course will be taking the final exam in CB2.

Topics

A. Multiple Choice/Short Answer/Fill In The Blanks – 6%
B. Summation – 8%
C. Recurrence Relation – 8%
D. Sorting Algorithms – 12%
E. Binary Trees – 12%
F. Binary Heaps – 12%
G. Tries – 10%
H. Bitwise Operators – 8%
I. Hash Tables – 8%
J. AVL Trees – 8%
K. Dynamic Memory Allocation – 5%
L. Linked Lists – 5%
M. Stack/Queue – 5%
N. Algorithm Analysis (minus Sums/Recurrence) – 5%

Exam Aids

You will be given the Foundation Exam Formula Sheet.
NO CALCULATOR OR OTHER ELECTRONIC DEVICES!!!

Sample Exams

Fall 2023 Final Exam:

https://www.cs.ucf.edu/~dmarino/ucf/transparency/cop3502/exam/FE-Fall2023.pdf

Spring 2024 Final Exam:

https://www.cs.ucf.edu/~dmarino/ucf/transparency/cop3502/exam/FE-Spr24.pdf

Outline of Topics for the Exam

I. Basics of C – if, loops, functions, array, strings, files

II. Sums

- **a. Notation**
- **b. Sum of Constant**
- **c. Sum of i, using formula sheet for i^2, i^3**
- **d. Geometric Sum**
- **e. Sum from i = a to i = b**

III. Recurrence Relations

- **a. How to Iterate**
- **b. Making guess after k iterations**
- **c. Using relevant substitution and base case information**

IV. Sorting

- **a. Bubble Sort**
- **b. Insertion Sort**
- **c. Selection Sort**
- **d. Merge Sort**
- **e. Quick Sort**

V. Binary Search Trees

- **a. Creating Nodes**
- **b. Tree Traversals (preorder, inorder, postorder)**
- **c. Insertion**
- **d. Searching**
- **e. Deletion**
- **f. Code Tracing**
- **g. Writing Code (recursive)**

VI. Binary Heaps

- **a. percolateUp**
- **b. percolateDown**
- **c. Insert**
- **d. deleteMin**
- **e. makeHeap**
- **f. Heap Sort**

VII. Tries

- **a. Basic struct**
- **b. Extra items to store in struct**
- **c. Checking for NULL**
- **d. Use of recursion on all 26 children**
- **e. Coding problems**

VIII. Bitwise Operators

- **a. left shift, right shift, and, or, xor**
- **b. How to use a number to indicate a subset.**
- **c. How to iterate through all possible subsets w/bitmask.**
- **d. Use of operators for set tasks (intersection, union), looking for commonality, coverage**
- **e. use of xor(^) in grading a T/F quiz, switching light bulbs**

IX. Hash Tables

- **a. Properties of a good hash function**
- **b. linear probing replacement technique**
- **c. quadratic probing replacement technique**
- **d. linear chaining hashing**

X. AVL Trees

- **a. AVL Tree Property**
- **b. Identifying nodes A, B and C for both insert and delete**
- **c. Restructuring for both insert and delete**
- **d. Delete may have multiple restructures**

XI. Structs, Pointers and Dynamic Arrays

- **a. how to allocate space dynamically**
  (array, 2d array, array of struct, array of ptr to struct,
  linked list node, bin tree node, etc.)
- **b. how to free space**
- **c. how to "resize" an existing array**
- **d. how to declare structs**
- **e. how to use pointers to structs**
- **f. how to use arrays of structs**
- **g. how to use arrays of pointers to structs**
- **h. how to pass structs or pointers to structs into a function**

XII. Linked Lists

- **a. Creating Nodes**
- **b. Checking for NULL**
- **c. Iterating through a list**
- **d. Insertion, Searching**
- **e. Deletion**
- **f. difference between ptr == NULL and ptr->next == NULL**
- **g. idea of storing a string in a linked list and assoc. functions**
- **h. idea of storing a big int in a linked list and assoc. functions**
- **i. Circularly linked**
- **j. Doubly linked**

XIII. Stacks

- a. Stack Array Implementation
- b. Stack Dynamically Sized Array Implementation
- c. Stack Linked List Implementation
- d. Stack Efficiency of push, pop
- e. Determining the Value of Postfix Expressions
- f. Converting Infix to Postfix
- g. Queue Array Implementation
- h. Queue Dynamically Sized Array Implementation
- i. Queue Linked List Implementation
- j. Efficiency of Enqueue and Dequeue
- k. Queue - Use in grid breadth first search

XIV. Algorithm Analysis

- a. Average case vs. Worst case
- b. Determining a Big-Oh bound via code segment
- c. Big-Oh timing problems
- d. Logs and exponents
- e. New problem analysis