Important COP 3502H Final Exam Information

Date: 4/25/2024 (Thursday)
Time: 10 am – 1 pm

Exam Aids: Three sheets of regular 8.5”x11” paper, front and back, and the Foundation Exam Formula Sheet (provided)

Test Format: Short Answer, Tracing, Coding, Math/Analysis Questions, etc.

Total Points = 125

Exam Archive:

www.cs.ucf.edu/~dmarino/ucf/transparency/cop3502/exam/

Archive shows types of questions I've asked in the past.

Outline of Topics for the Exam

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

II. Problem Solving on Arrays

- **a. Sorted List Matching**
- **b. Binary Search**
- **c. Sweeping Through Data**

III. 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**
- **g. how to pass structs or pointers to structs into a function**

VIII. Algorithm Analysis

- **a. Average case vs. Worst case**
- **b. Determining a Big-Oh bound via code segment**
- **c. Use of sums**
- **d. Big-Oh timing problems**
- **e. Logs and exponents**
- **f. Recurrence Relations**
- **g. New problem analysis**

VII. Recursion

- **a. Fibonacci, Factorial, Power, TipChart, SumDigits, Base Conversion, etc.**
- **b. Towers of Hanoi**

c. Binary Search
d. Fast Modular Exponentiation
e. Linked List Code
f. Floodfill
g. Brute Force (odometer, combinations, permutation, derangements, upwards idea)
h. Backtracking (Digit Divisibility, Sudoku, Hexagram, Prefix Primes)
i. Min-Max Trees

IV. 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

V. Stacks
a. Array Implementation
b. Dynamically Sized Array Implementation
c. Linked List Implementation
d. Efficiency of push, pop
e. Determining the Value of Postfix Expressions
f. Converting Infix to Postfix

VI. Queues
a. Array Implementation
b. Dynamically Sized Array Implementation
c. Linked List Implementation

d. Efficiency of Enqueue and Dequeue
e. Use in grid breadth first search

X. 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)

XI. 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

IX. Sorting

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

XII. Binary Heaps

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

XIV. Hash Tables

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

XIII. 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**

XV. Base Conversion

- **a. definition of number bases**
- **b. conversion from base b to base 10.**
- **c. conversion from base 10 to base b.**
- **d. conversion between two bases both powers of 2.**

XVI. 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**

XVII. Binary Search Applications

- **a. Searching an increasing/decreasing function**
- **b. Real Valued binary search**
  - **i. mid = (low+high)/2.0 (real div)**
  - **ii. low = mid or high = mid**
- **c. Integer binary search**
  - **i. mid = (low+high)/2 or (low+high+1)/2, problem specific so trace to figure out which**
  - **ii. Several options for low, high be careful**
- **d. Examples**
  - **i. Etch**
  - **ii. Careful Approach**
  - **iii. Airport Shuttle**