Study Guide for CS 313E Test 2 (Fall 2019)

The test is **comprehensive:**

- Knowing concepts from the previous exam will only help but we will not ask you to make another permute function!
- Remember to look over string and list manipulation
- Think about hard dictionary/list/string problems as these can help you solve parts of the questions on test 2!
- YOU DO NOT NEED TO KNOW TREES… yet

Define a simple class:

- A concept you should know from Test 1
- Remember how to use self and how a class works
- Look at the point, link class etc.

Stacks:

- https://www.geeksforgeeks.org/stack-data-structure/
- This link can explain what a stack is
- There are also sample problems near the bottom of the page, try a few. Don't do all, you'll be there for days
- Know that stacks are LIFO (last in first out) data structures
- Know about push, pop, and empty. You are not allowed to use more than those methods unless stated in the test question
- Balancing parentheses is a really good example to study ;)
    - CORRECT: ()() (())
    - INCORRECT CASES:
        - (()
        - (}{}, ({)}
        -)(}{
- Postfix and Prefix operations using a stack is a good example ;)
    - Did Mitra say yesterday (11/08) that we had to be able to convert from prefix (postfix?) to infix - does anyone know if we have to do operations and conversions with infix and not just postfix / prefix ? Thank you ~
- https://www.youtube.com/watch?v=SW14tOda_kI&list=PLqM7alHXFySF7Lap-wi5qlaD8OEBx9RMV&index=9&t=0s

Queues:

- https://www.geeksforgeeks.org/queue-data-structure/
- This link can explain what a queue is
- There are sample problems near the end of the page, just try a few
- Know that queues are FIFO (first in first out) data structures
- Know about enqueue, dequeue and empty
- Count or sum elements in a queue without destroying the list

        - Use a marker, some value that would not exist in the queue otherwise, to denote the end of the queue
            - For a list of positive integers, you may want to use 0 or a negative number
            - Enqueue the marker and dequeue, adding that element. Enqueue what you dequeued until you reach the marker and that denotes the end of the queue.
            - This does not destroy the queue as just a simple series of dequeues would
- https://www.geeksforgeeks.org/data-structure-gq/queue-gq/
- ^^ Do you know what a queue is?
- How do you reverse a queue

LinkedLists:

- Singly/doubly/circular
    - https://www.geeksforgeeks.org/data-structures/linked-list/
- Learn how each works and how to insert/delete and traverse
- Spend time on doubly linkedlist ;)
- https://www.geeksforgeeks.org/reverse-a-linked-list/
- Reorder Singly List:
    - Go from 12345 to 15243 (both input and output are linkedlists)
    - Google this problem after you have given it a try
    - https://www.programcreek.com/2013/12/in-place-reorder-a-singly-linked-list-in-java/
    - Problem covers main concepts such as:
        - Trailer/follow pointer
        - Insert and delete
        - Traversal
- How to check for cycles:
    - Fast pointer and slow pointer

Hashing:

- https://www.geeksforgeeks.org/hashing-data-structure/
- Open vs closed or single chaining for addressing
    - What are the benefits and how to implement them
- Linear/quadratic probing and double hashing
- Very useful in making programs fast
- Look at some sample problems!

Recursion:

- Back Tracking: how does it work
- Look at coding bat for recursion problems
- Look back at N Queens problem and navigating a maze problem

How to tackle the questions:

1. Chill, you got this (Don't chill too much, you only have 2.5 hours to do this test)
2. Look at the question and circle important restrictions and constraints
    a. Think of preconditions and postconditions
3. Ask yourself: "Do I understand what I need to do? How should I do it"
4. Write/think **in english** what the steps I need to do to solve the problem
5. Pseudo code
6. Write the actual python code

Recursion:

- Define the base case (when to stop)
- Do an operation or do something that you need to do (take the first step)
- Update values and call the recursive function (take the subsequent steps)

Big-O:

- The Big O notation defines an upper bound of an algorithm. For example, consider the case of Insertion Sort. It takes linear time in the best case and quadratic time in the worst case. We can safely say that the time complexity of Insertion sort is O(n^2). Note that O(n^2) also covers linear time.
- n stands for the number of elements in the array
- Basically, it's talking about how many operations your program is doing.

Think about Greedy, Brute force and dynamic programming

You may bring a single cheat sheet (8.5" x 11") with you with hand written notes on both sides of the paper.