Do Practice Problems!

Algorithm Analysis

- Be able to look at code and determine if it's O(1), O(log(n)), O(n), etc.
- Be able to solve time estimation problems

For each of the Data Structures covered so far (see below), know:

- All relevant Operations
    - O(???) for each
        - Worst case, Best case, Average case
    - Theoretical (non-Python) Algorithms for each (including recursive ones), and including drawing pictures of linked-lists and arrays.
    - Be able to Read/Write/debug Python code for each operation

- Data structures:
    - Stack
    - Queue
    - Circular Queue
    - Link-based list
    - Array-based list
    - Binary Search Tree (Know the algorithms, not necessarily the code since you haven't written it yet.)

Data Structures covered so far:

Stack

- Both Array and Linked List implementations, as implemented in Lab 2
- Operations:
    - push
    - pop
    - size
    - is_empty
    - is_full

Circular Queue, as implemented in Lab 3

- Both Array and Linked List implementations
- Operations:
    - enqueue
    - dequeue

List

- Both Array based and Link based
- Operations:
    - add (beginning, end, middle)
    - remove (beginning, end, middle)
    - find/get/contains (beginning, end, middle)
    - set (beginning, end, middle)

Binary Search Tree, will be implemented in Lab 5

- Operations:
    - insert/add
    - remove/delete
    - find/get
    - find min/max

Code Reading/Writing

- Be able to read/write Python code for operations of the covered data structures, especially for the operations implemented in labs/projects.
    - Labs: 1, 2, 3, 4
    - Projects: 1, 2
- Be able to read/write recursive functions such as those implemented in Lab 1 and Project 1
- Be able to read/write code that evaluates postfix expressions, and be able to evaluate them by hand.