CSSE230 Exam 1 SRT Review Session

The exam will cover the following topics:

- On the written part (60-70% of exam):
  - Finding exact and big-theta runtime of code snippets, knowing when and how to use a loop index table and summations.
  - More with big O, Theta, and Omega: true/false, using definitions to do proofs
  - ADT/Collections: Choosing an ADT to solve a given problem and knowing its runtime
  - Growable Arrays
  - MCSS
  - Something from the assignments, like binary search.
- On the programming part (30-40%):
  - Implementing an ADT using an array, nodes, or another ADT, or
  - Writing an efficient algorithm to solve a simple array-based problem

True or False Questions:

*Answer the following questions by circling either (T)rue or (F)alse. A statement is only considered true if it is **always** true, and a statement can be considered false if there is a **single counterexample**.*

a. **T** F if F(N) is O(N) than F(N) is $O(N^2)$

b. **T** F ln(N)/ln(5) is $\Theta(\log_2(N))$

c. T **F** Removing an item from the beginning of an ArrayList<T> is Θ(1)

d. T **F** *All* linked lists have an runtime of Θ(1) when adding at the end of the list.

e. T **F** The reason to use a TreeSet over a HashSet is because its methods run faster.

ADT Identification and Implementations:

For the following examples, supply an ADT, and Implementation, and an operational runtime with explanation.

Solutions have not been provided, there is no "right" answer although I used a list for the first section and a priority queue for the second.

An author, Brett Starson, is writing a huge epic book with hundreds of characters, locations, and important events. Brett's publisher is worried that Brett won't be able to finish his series, and has asked them to figure out just how many sequels there will be. Brett's attempt to document all the major events he needs to organize into books is getting out of hand, and he has realized he needs a digital solution. He needs an application that when given a range of numbers it retrieves a range of events. For example when run with (3, 5) that application would get the third, fourth and fifth events.

ADT:_____
Implementation:_____
Given one number, runtime for returning a series story event: _____
Explanation:

Brett has been able to compile all the important events into a 41 book series. His publisher is excited about this prospect, but is also incredibly stressed that Brett won't be able to finish before he needs to retire. It is only after 3-5 drafts, beta readers, a multi stage editorial process, and a book tour and release event for the publisher to consider the book complete. They want an application that schedules these tasks so the publisher can be confident that the books will arrive in a timely manner. Brett will always be writing, but he needs to be revising and advertising as well. The publisher needs an application that can order, and reorder tasks based off of what is most urgent, or most valuable to be completed.

ADT:_____
Implementation:_____
Given a task, runtime for the publisher to add it to the schedule: _____
Explanation:
(if you can think of other useful operations to optimize, please include them as well)

Exact Runtimes and Big Theta Estimates

*For each of the following give an exact number and theta estimate in terms of n for the number of times that **sum++** is run.*

```
for (int i = 0;i < n * n;i += n) {
      for (int j = n;j > 0;j --) {
            sum++;
      }
}
```

Exact: **N^2** Big-theta: **$\Theta(N^2)$**

```
// Assume n is a power of 3
for (int i = n; i > 0;i /= 3) {
      for (int j = 0;j < i; j ++) {
            sum++;
      }
      sum++;
}
```

Exact: ***Sum of (i = 0 to $\log_3 n$ of 3^i) + $\log_3 n$*** Big-theta: ***$\Theta(N)$***

Maximum Contiguous Subsequence Sum

a. What is the MCSS for {1, 2, -4, 1, 1, 4, -4, -3, 6, -8, 7}

   Answer for part (a) **7**

b. If we use the discussed O(N) algorithm, how many times will the sequence reset to zero (not including the initial value)?

   Answer for part (b) **3**

Big Theta Proof

Show that the following equations match the given big theta values. This will involve selecting two pairs of constants to form Big O and Big Ω proofs.

a. Prove that $f(n) = 4n^2 + \log_2(n) + 4 \in \Theta(n^2)$

$4n^2 + \log_2(n) + 4 <= 4n^2 + n^2 + 4n^2 = 9n^2$
I'll pick for c_O 9, and for n_O 2

$4n^2 + \log_2(n) + 4 >= n^2$
I'll pick for c_Ω 1 and for n_Ω 1

b. Prove that $f(n) = \log_2(2^n) + n^3 + n^2/100 \in \Theta(n^3)$

Very similar, it's still a polynomial time algorithm.