Part 1 - Misc Review [15 Points]

1. For each of the following functions, give (i) a tight lower asymptotic bound, (ii) a tight upper asymptotic bound, **and** (iii) a tight asymptotic bound or state that no tight bound exists. Your answers should be a concise description of the applicable complexity class: ***O***, $\mathbf{\Omega}$, $\mathbf{\Theta}$, each with a single term, no constants. [6 points]

RUBRIC: 0.5 for correct O, Omega. 1 point for correct Theta

a) $f(n) = 6n^2 + 4n\log(n) + n^3$

All bounds n^3

$$\text{b) } g(n) = \begin{cases} n & \text{if } n < 100 \\ n\log(n) & \text{if } n \bmod 2 = 0 \\ \log(n) & \text{if } n \bmod 2 = 1 \end{cases}$$

omega log n, O n log n, no theta

$$\text{c) } h(n) = \sum_{i=1}^{\sqrt{n}} \sum_{j=1}^{n} i$$

All bounds n^2

For the following two questions, consider the following code snippet:

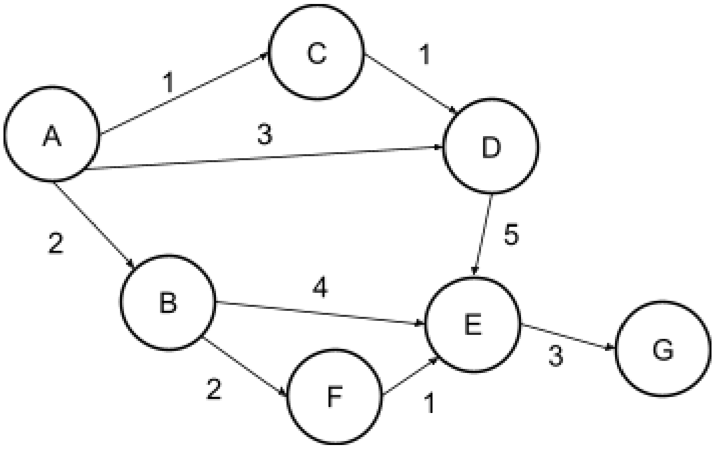
```
val seq = new MysterySequence()
seq.addSomething("X")
seq.addSomething("Y")
seq.addSomething("Z")
seq.addSomething("1")
print(seq.removeSomething())
print(seq.removeSomething())
print(seq.removeSomething())
seq.addSomething("2")
print(seq.removeSomething())
seq.addSomething("3")
print(seq.removeSomething())
```

2. What is printed if MysterySequence is a queue? [2 points]
   RUBRIC: 2 points if exactly correct
   XYZ12

3. What is printed if MysterySequence is a stack? [2 points]
   RUBRIC: 2 points if exactly correct
   1ZY23

4. Recall the sorted linked list from PA2 with a hinted version of insert. Describe how you could choose hints for the hinted insert that would result in an O(1) insertion time for situations where elements happen to be inserted in ascending order. [5 points]
   RUBRIC: 5 points if they mention any hint that would result in O(1) insert
   Pass in the most recently inserted node as the hint (or the tail as the hint)

Part 2 - Graphs (PA3) [20 Points]



Imagine we want to find the shortest paths in the above weighted graph starting from A.

5. In PA3, what **ADT** was used to organize the nodes in the work list to ensure we traversed them in the correct order to compute the shortest path? [5 points]
RUBRIC: 5 points if correct (priority queue, or heap). 2 points if queue.

6. At each step of the algorithm we dequeue a `(vertex, distance)` pair from our work list. For example, the first dequeue for the above graph would be `(A, 0)`. Write out the sequence of dequeue operations starting with `(A,0)`, and ending when the node **F** is dequeued. **Break ties in alphabetical order.** [10 points]
RUBRIC: 2 points per correct dequeue after (A,0) [only 1 point per dequeue that is a correct element but in the wrong order]
(A,0), (C,1), (B,2), (D,2), (D,3), (F,4)

7. Write out the `(vertex, distance)` pairs that are still in the work list **after F is processed**, or write none if no pairs will remain in the work list. [5 points]
RUBRIC: 2 points each for mentioning (E,6), (E,7). 1 for (E,5)
(E,5), (E,6), (E,7)

Part 3 - Hash Tables [30 Points]

8. Determine the bucket that each of the following movies would be assigned to in a HashTable (with chaining) that has **10 buckets**, and a hash function that assigns a hash code to each element based on the first letter of the movie title. (ie "A" => 0, "B" => 1, "C" => 2, etc) [5 points]
   RUBRIC: 1 point per correct answer

   Frankenstein: **5**

   The Shining: **9**

   Jeepers Creepers: **9**

   The Thing: **9**

   We Are Still Here: **2**

9. There are two important properties of a good hash function. Which of these properties does the hash function from question 8 have? [5 points]
   RUBRIC: 5 points for mentioning constant runtime
   It has an O(1) runtime

10. Why is the hash function from question 8 not a good hash function? [5 points]
    RUBRIC: 5 points for stating anything related to non-uniform distribution
    It does not uniformly distribute keys to buckets

11. Illustrate the process of inserting elements A, B, C, D, E, F into a hash table with an initial size of N = 5 and, a maximum load factor α_{max} = 0.7. To do so, draw out the contents of the hash table before each resize, as well as the final hash table after all elements have been inserted. [15 points]

    Assume the hash table uses chaining, and a hash function defined as follows:

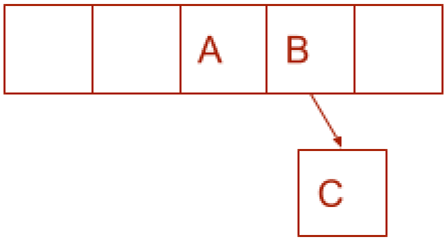
hash(A) = 42, hash(B) = 123, hash(C) = 128, hash(D) = 1, hash(E) = 57, hash(F) = 99

RUBRIC:

5 points for correctly hashing everything inserted when N = 5. Deduct 1 for each minor error.

5 points for correctly rehashing to size 10 after 3 items inserted

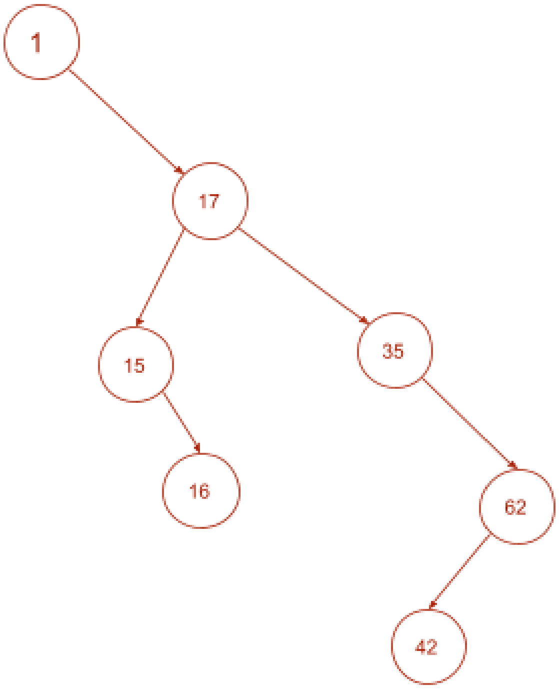
5 points for correctly hashing everything inserted when N = 10. Deduct 1 for each minor error.



	D	A	B				E	C	F

Part 4 - Trees [20 Points]

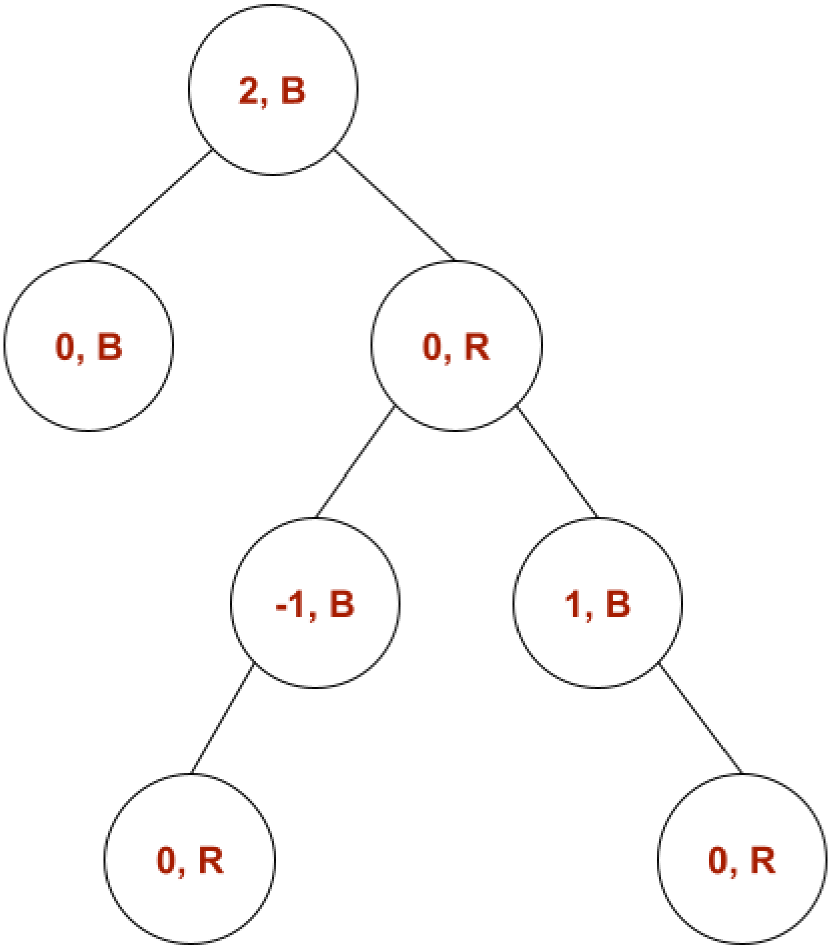
12. Draw the BST (regular BST, not AVL or Red-Black) that results from inserting the following elements in the order listed: [7 points]
RUBRIC: 1 point per correctly inserted node
1, 17, 35, 15, 16, 62, 42



13. Give one example of a rotation that would **decrease** the height of the tree in the previous question. [3 points]
RUBRIC: 3 points for correct rotation (rotate left around 1 or 35), or if they give a correct rotation for the tree they drew.

14. The following tree is a Red-Black tree. For each node, label it with its balance factor, as well as a color (red or black) that would lead to a valid coloring according to the Red-Black tree constraints. [7 points]

RUBRIC: 0.5 points per correct label (if only thing wrong is sign, deduct 1 total)



15. Is the above tree a valid AVL tree? Why or why not? [3 points]

RUBRIC: 1 point for no. 2 for pointing out valid reason why.

No. The balance factor of the root is 2.

Part 5 - Short Answer [15 Points]

Imagine you are designing a database for the university library. The data they have is a sequence of records that contain Book Title, Transaction Type, and Date of Transaction, where transaction type is either CHECKOUT or RETURN. A snippet from this dataset is shown below:

…
Frankenstein, CHECKOUT, 3/16/23
Dracula, CHECKOUT, 3/17/23
Dracula, RETURN, 3/19/23
The Invisible Man, CHECKOUT, 3/24/23
Frankenstein, RETURN, 3/25/23
Pet Sematary, CHECKOUT, 3/31/23
Frankenstein, CHECKOUT, 3/31/23
…

You can assume there are ***n*** records in the dataset.

16. The library wants you to iterate through the dataset and quickly determine which books have been checked out but not returned. In 3-5 sentences, describe an algorithm to efficiently complete this task. In your description make sure to name any specific data structures you use, as well as the runtime of your solution. [5 points]
RUBRIC: 2 points for mentioning hash table in some way
2 points for describing an O(n) algo. 1 point for describing O(n log n) algo.
1 point for correctly stating runtime of their algorithm

Use a Set (or Map) implemented using a HashTable.
Iterate through the dataset, and for each record, if it is a CHECKOUT record insert that book (keyed by title) into the HashTable. If the record is a RETURN, remove it from the HashTable.
This only requires one pass through the data, and each operation is expected O(1), therefore total runtime is expected O(n).

17. The library wants to be able to determine which book has been checked out most in a given range of time. In 3-5 sentences, describe an algorithm to efficiently complete this task below. In your description make sure to name any specific data structures you use, as well as the runtime of your solution. [5 points]
RUBRIC: 2 points for mentioning hash table in some way
2 points for describing an O(n) algo. 1 point for describing O(n log n) algo.
1 point for correctly stating runtime of their algorithm

Use a HashMap of title to count.
Iterate through the dataset, and for each record, if it is a CHECKOUT that falls within the given time range, add it to the map with a count of 1, or if already in the map, increment the count by 1. As you do this, keep track of which book has the highest count.
This only requires one pass through the data, and each operation is expected O(1), therefore total runtime is expected O(n).

18. The library wants you to reorganize the dataset in such a way that they can query the data to get a list of all books that have been checked out in a given range of time. What data structure would you use to reorganize the data to ensure that these queries can be answered quickly, but that also allows new records to be added quickly? Briefly describe why your data structure is the best choice. [5 points]

RUBRIC: 3 points for mentioning balanced BST (or sorted ArrayBuffer) in some way (1 if mentioning plain BST or HashTable)
2 points for reasonable description

The data should be organized in a balanced BST with the data as the key.

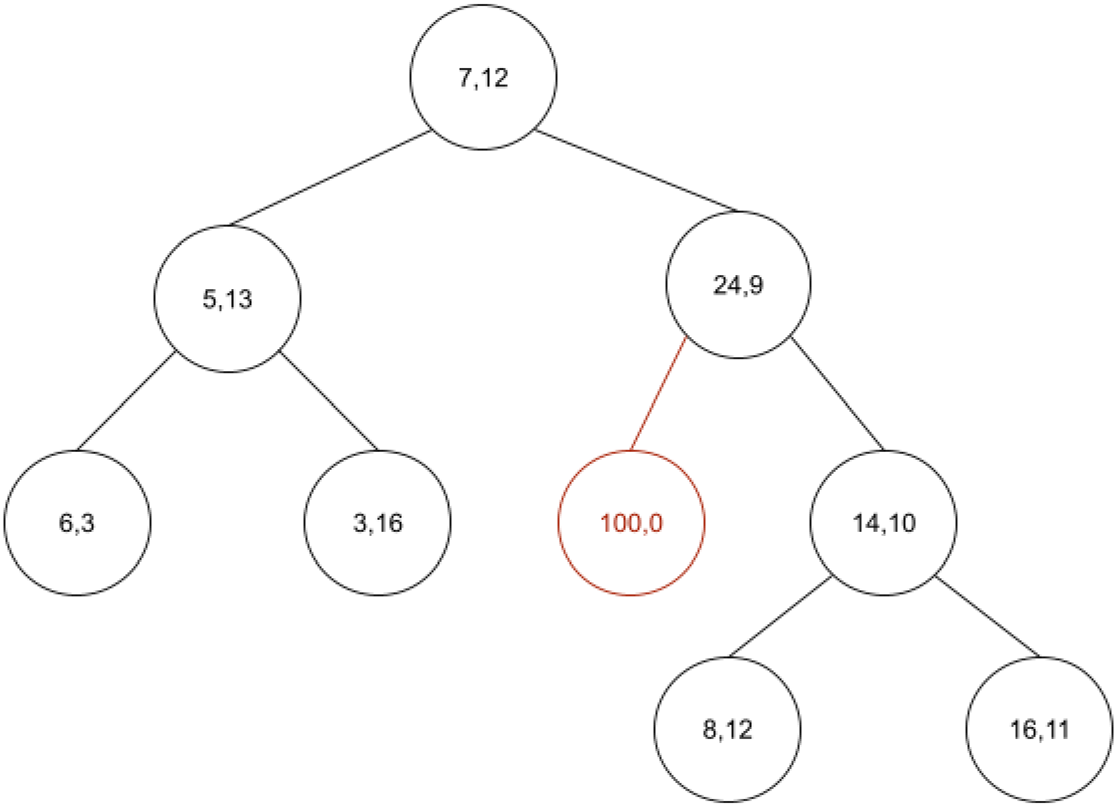
Tree based structures allow us to search in log(n) time, but also maintain ordering unlike HashTables. We can therefore find the first record in the range in log(n) time and continue an in-order traversal until we hit the last item in the range.

This could also be done with a sorted array, but that would not allow for insertions for new records efficiently. EDIT: If we assume entries are added in chronological order, then sorted ArrayBuffer also works.

HashTables would require an O(n) query time instead of O(log n) because the data is not sorted. So for an arbitrary range you have to search the whole hash table (you can only hash on a single date, not a range of dates).

Part 6 - Extra Credit [5 Points]

1. KD-Trees are a variation on BSTs we discussed in class that allow us to organize multi-dimensional data. For two dimensional data, the dimension that is partitioned on at each level of the tree alternates. Insert (100, 0) into the KD-Tree shown below. [5 points]
   RUBRIC: 5 points for correct insertion location



Scrap Paper