CS 240 Final Exam, Version A

Spring 2019

Name: _____

e-ID: _____

Page	Points	Score
3	18	
4	16	
5	5	
6	11	
7	5	
8	7	
9	5	
10	7	
11	11	
12	5	
Total:	90	

Please sign your name to indicate that you agree with the following honor code statement:

I have neither given nor received unauthorized assistance on this examination. I will not share information about this test with students in other sections of the course.

_____

You may find the following useful:

1 Series

Arithmetic series:

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

Geometric series:

$$\sum_{i=0}^{n} a^i = \frac{1-a^{n+1}}{1-a}$$

2 Big-Ω, O, Θ definitions

2.1 Algebraic definitions

In all of the following let $f(n)$ and $g(n)$ be complexity functions.

- $f(n) \in \Omega(g(n))$ if and only if there exist constants $N \geq 0$ and $c > 0$ such that $f(n) \geq cg(n)$ for all $n \geq N$.
- $f(n) \in O(g(n))$ if and only if there exist constants $N \geq 0$ and $c > 0$ such that $f(n) \leq cg(n)$ for all $n \geq N$.
- $f(n) \in \Theta(g(n))$ if and only if $f(n) \in O(g(n))$ and $f(n) \in \Omega(g(n))$.

2.2 Limit definitions

Let $L = \lim_{n\to\infty} \frac{f(n)}{g(n)}$. Then,

- If $L > 0$ then $f(n) \in \Omega(g(n))$.
- If $L < \infty$ then $f(n) \in O(g(n))$.
- If $0 < L < \infty$ then $f(n) \in \Theta(g(n))$.

1. **Multiple Choice**

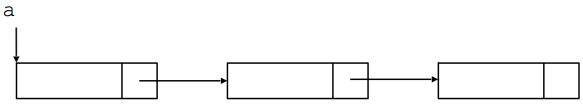
(a) (3 points) A data structure in which the elements are stored in adjacent memory locations is said to be:

A. Abstract
B. Contiguous
C. Linked
D. None of the above

(b) (3 points) Letting n denote the size of the array to be searched, the worst-case time bound of the binary search algorithm is:

A. $\Theta(n)$
B. $\Theta(n^2)$
C. $\Theta(\log n)$
D. None of the above

(c) (3 points) Consider the following instance of a linked data structure.



This kind of data structure would be *most* suitable for an implementation of:

A. A list
B. A queue
C. A stack
D. None of the above

(d) (3 points) Which of the following could plausibly be the height of an AVL tree? Select all that apply.

A. 1
B. 10
C. 100
D. 1,000
E. Any of the above.

(e) (3 points) Which of the following is most consistent with the statement "The running time of algorithm A is in $\Omega(n^2)$."

A. Algorithm A requires n^2 steps in the worst case.
B. Algorithm A requires n^2 steps in the best case.
C. Algorithm A requires more than n steps.
D. Algorithm A requires fewer than n^3 steps.

(f) (3 points) Which of the following data structures is most suitable for implementing the List ADT for an application that will frequently access elements from the middle of the list?

A. Linked list
B. Dynamic array
C. Hashtable
D. AVL tree

2. **Short Answer**

(a) (3 points) A queue is an abstract data type in which elements are added to the "rear" and removed from the "front". Describe a queue using only terms from the set {"last", "first", "in", "out"}.

(b) (3 points) List two data structures that would be appropriate for implementing each of the following ADTs. (Be as specific as possible. "Tree" would not be a good answer since there are many tree-based data structures.)

- Stack
- Map
- Priority Queue

(c) (3 points) In practice, which would we typically prefer, an algorithm that completes in $100n$ steps or an algorithm that completes in $n \log_2 n$ steps? Justify your answer.

(d) (3 points) Consider the following statements:

- Adding a single element to the end of a Java `ArrayList` may require $\Theta(1)$ or $\Theta(n)$ steps, depending on whether there is unused space in the underlying array.
- Adding n elements to the end an initially empty ArrayList requires $\Theta(n)$ steps.

Are both statements true? If the first statment *is* true, what sort of argument can be used to establish the truth of the second statement?

(e) (2 points) List two advantages of Quicksort over Mergesort.

(f) (2 points) List two advantages of Mergesort over Quicksort.

3. (5 points) Analyze the following method. Use the length of `values` as the input size and *calls to* `countMe()` as the basic operation. Provide both the exact growth rate and the appropriate big-Θ complexity class. You may assume that the length of `values` is a power of two.

```
public static void fun1(int[] values) {

    for (int i = values.length; i >= 2; i /= 2) {
            for (int j = 0; j < values.length; j++) {
                    countMe();
            }

            for (int j = 0; j < 100; j++) {
                    countMe();
            }
    }
}
```

4. (6 points) Each of the boxes below represents a set of growth functions. Place the functions $e - h$ into the appropriate sets. Functions *may appear in more than one set* and some sets *may not contain any of the provided functions*. I've started for you by correctly filling in the box for $\Omega(n)$.

   - $e(n) = \log_2 n + 140$
   - $f(n) = n^2 + 6 \log n$
   - $g(n) = 4n \log_2 n + 2n$
   - $h(n) = 2^{50} n^4$

$O(1)$	$O(\log n)$	$O(n)$	$O(n \log n)$	$O(n^2)$

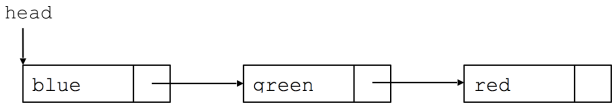
$\Omega(1)$	$\Omega(\log n)$	$\Omega(n)$	$\Omega(n \log n)$	$\Omega(n^2)$
		f, g, h		

$\Theta(1)$	$\Theta(\log n)$	$\Theta(n)$	$\Theta(n \log n)$	$\Theta(n^2)$

5. (5 points) Demonstrate that the following proposition is true. You may use any of the formal definitions that we have discussed.

   - $6n^2 + 2n \in \Theta(n^2)$

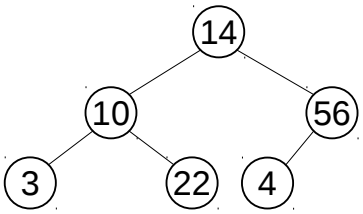
6. (5 points) Consider the following instance of a linked data structure in which each node contains a string named `value` and a reference named `next`. In this example, the node containing "red" was added first, the node containing "green" was added second, and the node containing "blue" was added third.



Implement a method named `remove()` that will remove and return the most recently added string (or `null` if there are no nodes) and maintain the order of the remaining nodes.

```
public class LinkedStructure {
   private Node head;
   public LinkedStructure() {
      head = null;
   }
   // Other methods not shown...
   public String remove() {









   }
}
```

7. Consider the following binary tree:



(a) (2 points) List the nodes in the order they would be visited in a pre-order traversal.

(b) (2 points) List the nodes in the order they would be visited in a post-order traversal.

(c) (1 point) Is this tree *full*? Justify your answer.

(d) (1 point) Is this tree *complete*? Justify your answer.

(e) (1 point) Is this a valid binary search tree? Justify your answer.

8. (5 points) Given the following message:

   "DEQUEUE"

   Produce a Huffman encoding of the message. When combining trees always place the lower-frequency child on the left. To break ties, use the recommended strategy from the PA: for two trees with equal frequency, the tree containing the earliest letter in the alphabet is "smaller".

9. (a) (1 point) Write the values of the following recurrence for $n = 0, 1, 2, 3$.
$T(0) = 7$
$T(n) = 1 + 2T(n-1)$

(b) (6 points) Solve the recurrence above using either the tree method or the method of backward substitution. Show your work.

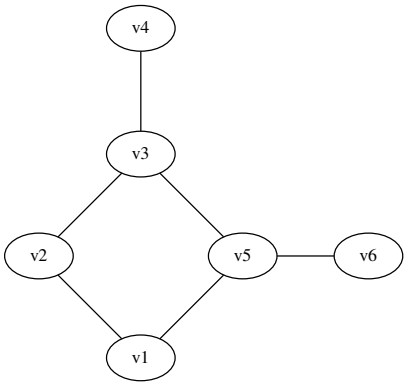
10. (a) (4 points) Assume there are two hash tables with five buckets and differing collision resolution strategies as labeled below. Show the contents of the hash tables after the following keys are inserted in order into each table: 2, 5, 10, 15. Use the following hash function: $h(x) = x^2 \ mod \ 5$.

Open Addressing with Linear Probing	Separate Chaining
0 1 2 3 4	0 1 2 3 4

(b) (1 point) What is the load factor for these hash tables after the four keys are inserted?

(c) (2 points) In practice, linear probing is rarely used as the collision resolution strategy for open addressing. Why not? Describe an alternative probing strategy and explain how it addresses the shortfalls of linear probing.

11. (4 points) Given the following graph:



list the order in which vertices are visited by a breadth-first-search from vertex `v2`. Assume that neighbors are accessed in increasing numerical order.

12. (5 points) The Department of Transportation has installed traffic cameras at each mile marker for all interstate highways in Virginia. These cameras record the license plate number, time and location information for each car that passes. All data is set to a central server in the form of `CameraData` objects as defined below:

```
public class CameraData {

  private int timeStamp;
  private String plateNumber;
  private MileID newLocation; // Identifier for the region of road the car is entering
  private MileID oldLocation; // Identifier for the region of road the car is departing

  // Methods not shown...
}
```

You have been tasked with developing an application to monitor this data. The application must support two operations: (1) Efficiently return the current location of any car given its license plate number. (2) Efficiently return the identity of all cars currently present on any highway segment. What ADTs would you use to solve this problem and how and why would you use them? (Be careful! This question is asking you to choose one or more ADTs. It is *not* asking about data structures.)