CS 2501: DSA1 Weekly Quizzes

Name

You MUST write your e-mail ID on **EACH** quiz page that you choose to complete. Please put your name on the top of each page you complete as well.

You may complete up to two pages of this quiz booklet. If you complete more than that, we will only grade two of the pages (which two is up to our discretion).

There are 15 pages to this quiz booklet. Once the time starts, please make sure you have all the pages.

This quiz is CLOSED text book, closed-notes, closed-calculator, closed-cell phone, closed-computer, closed-neighbor, etc. Questions are worth different amounts, so be sure to look over all the questions and plan your time accordingly. Please sign the honor pledge below.

A crash reduces
Your expensive computer
To a simple stone.

Module 1: Basic Java 1

Your TA will select one of the following short answer questions for you to answer:

(a) What is the difference between a float and a double (looking for high level description here)?

(b) Java is strongly typed. What does this mean? Give an example to illustrate your point.

(c) Describe one difference between primitive and Object types in Java.

(d) Briefly, what is the Java API and why is it useful?

(e) What is casting? Provide an example to illustrate your point.

1. [1 points] Answer the question that was selected.

2. [1 points] For this question, write a simple java program that reads in two variables and does one of the following. Your TAs will randomly select one of these options.

(a) Reads in two Strings from the keyboard and prints them to the console.

(b) Reads in two integers from the keyboard and prints their sum to the console.

(c) Reads in two doubles from the keyboard, converts them to integers, and prints the sum of the integers to the console.

```
//IMPORTS AND SUCH HERE
public class QuizQuestion{
        public static void main(String[] args){
                //YOUR CODE STARTS HERE:









        }
}
```

Module 2: Basic Java 2

Your TA will select one of the following short answer questions for you to answer:

(a) If an array is passed to a method and its contents altered, will the actual parameter be changed as well? Why or why not?

(b) True or False: An if statement can contain an expression that evaluates to an integer because Java knows how to treat integers in this scenario. Explain your answer.

(c) In Java, what happens if you do not include curly braces with your if-statements?

(d) Briefly describe the difference between how primitives and references are stored in memory.

(e) Write a short code snippet that produces a shared reference.

3. [1 points] Answer the question that was selected.

4. [1 points] For this question, write a Java method that takes in an array of integers as a parameter and does one of the following. Your TA will select which method you write.

(a) Compute the third highest integer in the array and print it to the console.

(b) Print out all of multiples of 10 in the array in backwards order.

(c) Print out all of the positive, even numbers.

```
public static void quizMethod(int[] a){









}
```

Module 3: Basic Java 3

Your TA will select two of the following short answer questions for you to answer:

(a) When writing our **Card** class in lecture. What was the purpose of the *constructor*? Be as precise as you can here.

(b) What is an enum? Give an example of when an enum is useful and describe the advantage of using one.

(c) Why is it important to write the .equals() method when writing a class? Give a concrete example.

(d) Describe why you might choose to make a field in your class private (instead of public).

(e) Write a small Point class. The class should contain two integer fields (x and y), a constructor that sets x and y, and a method called *distance(Point other)* which returns the distance between this point and the other point.

(f) In lecture, we wrote/saw a **Deck** class. Write a short *equals* method for the Deck class. Two decks are equal if and only if the array of cards contains the same cards in exactly the same order and the top indicator is equal as well.

5. [1 points] Answer the first question that was selected.

6. [1 points] Answer the second question that was selected.

Module 4: Vectors

Your TA will select one of the following short answer questions for you to answer:

(a) Describe the difference in efficiency of inserting at the front of a Vector versus the back. Are there any special cases involved?

(b) Verbally describe how removing from the end of a Vector works? In other words, how do we model the removal?

(c) List one strength and one weakness of using a Vector. Please describe these in sufficient detail.

(d) What is polymorphism? Why is it useful? Describe using the example of List and Vector from class.

(e) How efficient is it to grab the item at a specific index of a Vector? How about to find a specific item in the Vector (e.g., is 10 in the Vector?). Briefly describe why these are different.

7. [1 points] Answer the question that was selected.

8. [1 points] For this question, write one of the methods for a Vector class. Your TA will select one of the following methods: insert(T data), remove(T data), insertAt(int index, T data), or resize().

```
public class Vector<T> implements List<T>
        private T[] data;
        private int size = 0;
        private static final int INITIAL_CAPACITY = 100;

        //TODO: WRITE THE CHOSEN METHOD HERE.
```

Module 5: Linked Lists

Your TA will select one of the following questions for you to answer:

(a) Describe the runtime of these Linked List operations: insert at head, at tail, remove at head, at tail.

(b) Describe why retrieving the item at index i is slower with a Linked List than with a Vector or Array.

(c) Describe the difference in how Linked Lists are laid out in memory versus Arrays. Be as precise as possible.

(d) Suppose you have a doubly-linked-list in which each node stores one char of a word. Write psuedo-code describing a method that returns true if this linked list is a palindrome (e.g., racecar).

(e) Suppose you have a singly-linked list (head pointer + next pointers). Write psuedo-code for a method that reverses the order of the list (change references here, not the contents of the nodes).

9. [1 points] Answer the question that was selected.

10. [1 points] For this question, write one of the methods for a LinkedList class. Your TA will select one of the following methods: insertAtHead(T data), removeAtTail(), insertAt(int index, T data), get(int index) or find(T data).

```
public class LinkedList<T> implements List<T> {
        private ListNode<T> head, tail;
        private int size;

        //TODO: WRITE THE CHOSEN METHOD HERE.
```

Module 6: Stacks and Queues

This quiz question is about the *space efficiency* of List and Array based stacks and queues. Your TA will randomly select the following things:

(a) Whether we are dealing with a stack or a queue

(b) What type is being stored in the queue (int, char)

(c) Whether the stack/queue is array-based (int fields where necessary, initial capacity of 100) or linked-list based (doubly-linked, head and tail pointer to dummy nodes, int count).

(d) How many items have been inserted into the stack/queue ($n \leq 30$)

When answering, remember that integers are 4 bytes, chars are 1 byte, and references are 8 bytes. The size of objects (e.g., ListNode) is just the sum of the size of its fields.

11. [1 points] Given the constraints above, how many bytes of memory does the data structure consume? Do not include the reference to the stack/queue in your total

12. [1 points] For this question, write one of the methods for an array-based queue class. Your TA will select one of the following methods: enqueue(T data), T dequeue(). You can (and should) invoke resize() if necessary.

```
public class Queue<T> implements IQueue<T> {
  private T[] data;
  int size, front, back;

  //TODO: WRITE THE CHOSEN METHOD HERE.
```

Module 7: Big-Oh

Your TA will select one of the following questions regarding the formal definition of Big-Oh.

(a) Explain the purpose of the constant n_0 in the formal definition of Big-Oh. In other words, in what specific situations is it most useful?

(b) Explain the purpose of the constant c in the formal definition of Big-Oh. In other words, in what specific situations is it most useful?

(c) Describe a specific situation in which we would use Big-Omega ($\Omega(g)$) to analyze an algorithm.

13. [1 points] Answer the selected question.

For this question, consider the following functions:

- $nlog_2(n)$
- n^2
- b^n
- $log_d(n)$
- n
- 1
- $n^a log_2(n)$
- n
- cn^3
- $log_2(n)$
- n^3

14. [1 points] Your TA will randomly select four constant integers a, b, c, and d. Substitute those constants in to the functionas and then order them from slowest to fastest growing. If the functions are in the same equivalence class, then make sure to note that in your ordering.

Module 8: Recursion

15. [1 points] Your TA will select one of the following methods for you to code **recursively** (your solution MUST be recursive, but you may write a helper method with additional parameters if you wish):

   (a) void print(ListNode c): Given the first node of a singly-linked list, print out the list in reverse order.

   (b) void reverse(int[] a): Given an array of integers, reverse the order of values in the array.

   (c) int sum(int[] a): Given an array of integers, return the sum of the even integers in the list.

16. [1 points] Your TA will select another method for you to code **recursively** (same rules as above):

   (a) void printBinary(int n): Prints all of the valid binary numbers of length n (any order is fine, but no duplicates).

   (b) targetSum(int n, int target): Prints all of the integers with exactly n digits whose digits sum to exactly the given target. For example, $n = 2$ and target$= 2$ would print (in any order) 11, 20, 02. You may assume you have access to a method called sum(int x) that sums the digits of x for you.

   (c) void splitList(LinkedList l, LinkedList a, LinkedList b): Given a linked-list l, and two empty lists b and c, split l by putting every other element in a, and the other alternating elements in b. List l should be empty after the method completes.

Module 9: BSTs and AVL

Your TA will select one of the following questions regarding trees.

(a) List the big-theta worst-case runtimes for find(), insert(), and remove() on a binary search tree. Explain your answers.

(b) List the big-theta worst-case runtimes for find(), insert(), and remove() on an avl tree. Explain your answers.

(c) When fixing an avl tree via rotations, what is the maximum number of rotations that can occur? Explain your answer.

(d) Write a short method that, given a TreeNode object, prints the tree rooted at that node using a post-order traversal.

(e) Explain a situation in which you would NOT want to use a tree data structure.

(f) When discussing Java Inheritance, we discussed how Java uses dynamic dispatch. Explain what dynamic dispatch is, BSTs and AVL Trees as an example in your explanation.

17. [1 points] Answer the selected question.

18. [1 points] Your TA will select a random method for you to code. You will either write insert(T data), find(T data), or remove(T data) for a BST class, or rotateRight(TreeNode node) for an AVL class.

Module 10: Basic Sorts

Your TA will select one of the following questions regarding sorting algorithms.

(a) In class, we proved that any sorting algorithm that only swaps adjacent elements in a list MUST be $\Omega(n^2)$. Describe the proof.

(b) Briefly describe why we used Java's Comparable Interface for sorting. What was the benfit of this interface?

(c) What is a stable sort? Give a concrete example in which we would want to use a stable sort.

(d) What is an in-place sort? What is the benefit of sorting in-place?

(e) In class, we made a distinction between comparison sorts and adjacent sorts. Describe this difference.

(f) Insertion sort is very fast in some situations. Describe and explain those situations.

19. [1 points] Answer the selected question.

20. [1 points] Your TA will randomly select either bubbleSort(T[] list) or insertionSort(T[] list) for you to implement.

```
public static<T extends Comparable T>> void aSortingAlgorithm(T[] list){
```

Module 11: Advanced Sorts

Your TA will select one of the following questions regarding sorting algorithms.

(a) What are the Big-Theta worst-case and best-case runtimes of mergesort? Explain your answer.

(b) What are the Big-Theta worst-case and best-case runtimes of quicksort? Explain your answer.

(c) Is mergesort in-place? How about stable? Explain your answers.

(d) Is quicksort in-place? How about stable? Explain your answers.

(e) In class, we saw an argument for why comparison sorts must be $\Omega(nlogn)$. Recreate this argument.

(f) Describe why one might implement a hybrid sorting algorithm? Give a concrete example to support your answer.

21. [1 points] Answer the selected question.

22. [1 points] Your TA will randomly select one of the following methods for you to write. You may assume you have a swap function that takes a list and two indices and swaps the items at those locations. You may also assume you have the methods you are not writing (e.g., if writing mergesort, you can invoke merge.)

```
private static<T extends Comparable<T>> void mergeSort(T[] list, int i, int j)
private static<T extends Comparable<T>> void merge(T[] list, int i, int mid, int j)
private static<T extends Comparable<T>> void quickSort(T[] list, int i, int j)
private static<T extends Comparable<T>> int lomutoPartition(T[] list, int i, int j)
```

Module 12: Hash Tables

Your TA will select one of the following questions regarding hash tables.

(a) What is the worst-case runtime for inserting into a hash table using each of linear probing, quadratic probing, and separate chaining? Assume that your hash table does not check for duplicates.

(b) What is the worst-case runtime for removing from a hast table using each of linear probing, quadratic probing, and separate chaining? Assume that your hash table does not check for duplicates.

(c) What the worst-case runtime of resizing and rehashing a hash table using each of linear probing, quadratic probing, and separate chaining? Assume that your hash table does not check for duplicates.

(d) Briefly explain the trade-off in using a high vs a low load factor threshold for resizing.

23. [1 points] Answer the selected question.

24. [1 points] Insert the following elements into the provided hash table. Your TA will select an arbitrary hash function(s), five arbitrary integer values to insert, and a collision resolution strategy (separate chaining, linear probing, quadratic probing)

0	
1	
2	
3	
4	
5	
6	
7	

$h(x) =$

$h_2(x) =$ //if necessary

Five values to insert:

Module 13: Priority Queues

Your TA will select one of the following questions regarding priority queues.

(a) Provide three advantages to storing a heap in an array, rather than using objects and references?

(b) What is the worst-case runtime of push() on a min-heap? What about the expected time? Explain your answer.

(c) What is the worst-case runtime of poll() on a min-heap? What about the expected time? Explain your answer.

(d) Given the following min-heap: [NULL, 5, 7, 9, 22, 15, 12, 35]. Push a random integer n (n will be chosen by your TAs)

25. [1 points] Answer the selected question.

26. [1 points] Your TA will randomly select one of the following min-heap methods for you to write. You may assume the others are implemented for you (e.g., you may use percolateDown() in poll())

```
private ArrayList<T> heap; //you may assume this field exists

private void percolateUp(int index)
private void percolateDown(int index)
public T poll()
public void heapify() //turns heap field
```

Module 14: Concurrency

Your TA will select one of the following questions regarding priority queues.

(a) When writing basic threads in Java, what is the difference between the methods run(), start(), and join(). Be as precise as you can.

(b) Briefly describe the overall principle behind creating Java Exceptions? Why wouldn't we just let the code crash, or print out an error message, or similar?

(c) What is a race condition? Give an example.

(d) What is a lock? What does it do and why is it useful?

(e) What is a deadlock? Give an example of how a deadlock might occur?

27. [1 points] Answer the selected question.

28. [1 points] Your TA will randomly select either enqueue(T data) or dequeue() for a concurrent queue. Write the selected method. Your queue should be linked-list based and be thread-safe. Please list any fields for the class you'd like to add.

```
public class ConcurrentQueue<T> implements IQueue<T>{
        //list any fields you'd like here:



        //Write the requested method here:
```