CS 1332 Exam 1A

Bingyao Wang

TOTAL POINTS

94 / 100

QUESTION 1

1 Circular Linked List - Coding 11 / 15

✓ + **1 pts** **Throws some exception when data == null**

When size == 0 (or head is null):

+ **1 pts** Creates a new node with the data

+ **2 pts** Sets the new node's next pointer to itself (NOT the old head, which is null)

+ **1 pts** Sets the head to the new node

Option 1: used the O(1) trick

✓ + **1 pts** **Creates a new node**

✓ + **2 pts** **new node's next = head's old next**

✓ + **2 pts** **head.next = new node**

✓ + **2 pts** **Attempts to swap/change nodes' data**

✓ + **2 pts** **Correctly swaps/arranges the data so that the head now has the new data, and head's next has the old data**

- **1 pts** Explicitly changes head to the wrong node

Option 2: used an O(n) loop

+ **0.5 pts** Creates a new node with the data

+ **1 pts** new node's next = head

+ **1 pts** Correctly iterates to the node before the head

+ **1 pts** Sets the tail's next pointer to the new node

+ **0.5 pts** Sets head = new node

Option 3: addToFront like in singly-linked list

+ **1 pts** Creates a new node with the data whose next pointer points to the current head

+ **2 pts** Sets head = new node

✓ + **1 pts** **Increments size if and only if a node is added**

- **1 pts** Syntax error

- **1 pts** Explicitly handles an edge case, but does so incorrectly

QUESTION 2

Binary Search Trees - Diagramming 15 pts

2.1 add(13) 5 / 5

✓ + **1 pts** **Valid BST, or has at most 2 minor transposition errors (e.g. data unaffected by the operation was copied incorrectly)**

✓ + **1.5 pts** **13 was added to the tree**

✓ + **2.5 pts** **13 was added to the tree correctly (to the left of 15)**

2.2 remove(32) 5 / 5

✓ + **1 pts** **Valid BST, or has at most 2 minor transposition errors (e.g. data unaffected by the operation was copied incorrectly)**

✓ + **1.5 pts** **32 was removed from the tree**

✓ + **2.5 pts** **32 was correctly removed from the tree (18's right child is now 26 with its children unchanged)**

2.3 remove(47) 5 / 5

✓ + **1 pts** **Valid BST, or has at most 2 minor transposition errors (e.g. data unaffected by the operation was copied incorrectly)**

✓ + **1.5 pts** **47 was removed from the tree**

✓ + **2.5 pts** **47 was correctly removed from the tree (data was replaced with predecessor/sucessor)**

- **1 pts** Successor was used

- **0.5 pts** Predecessor/successor was not removed correctly

+ **0 pts** Incorrect

QUESTION 3

Stacks and Queues - Lines of Code 10 pts

3.1 | 2.5 / 2.5

✓ + **2.5 pts** **A**
+ **0 pts** B
+ **0 pts** C
+ **0 pts** D

3.2 II **2.5 / 2.5**
+ **0 pts** A
+ **0 pts** B
+ **2.5 pts** C
✓ + **2.5 pts** **D**

3.3 III **2.5 / 2.5**
+ **0 pts** A
✓ + **2.5 pts** **B**
+ **0 pts** C
+ **0 pts** D

3.4 IV **2.5 / 2.5**
+ **0 pts** A
+ **0 pts** B
✓ + **2.5 pts** **C**
+ **0 pts** D

QUESTION 4

Scenario Matching 15 pts

4.1 I **3 / 3**
+ **0 pts** A
+ **0 pts** B
+ **0 pts** C
+ **0 pts** D
+ **0 pts** E
+ **0 pts** F
✓ + **3 pts** **G**
+ **0 pts** H
+ **0 pts** I
+ **0 pts** J
+ **0 pts** K
+ **0 pts** Multiple Answers

4.2 II **3 / 3**
+ **0 pts** A
✓ + **3 pts** **B**
+ **0 pts** C
+ **0 pts** D
+ **0 pts** E
+ **0 pts** F
+ **0 pts** G
+ **0 pts** H
+ **0 pts** I
+ **0 pts** J
+ **0 pts** K
+ **0 pts** Multiple Answers

4.3 III **3 / 3**
+ **0 pts** A
+ **0 pts** B
+ **0 pts** C
+ **0 pts** D
+ **0 pts** E
+ **0 pts** F
+ **0 pts** G
+ **0 pts** H
+ **0 pts** I
+ **0 pts** J
✓ + **3 pts** **K**

4.4 IV **3 / 3**
+ **0 pts** A
+ **0 pts** B
+ **0 pts** C
+ **0 pts** D
+ **0 pts** E
✓ + **3 pts** **F**
+ **0 pts** G
+ **0 pts** H
+ **0 pts** I
+ **0 pts** J
+ **0 pts** K
+ **0 pts** Multiple Answers

4.5 V **3 / 3**
+ **0 pts** A
+ **0 pts** B

+ **0 pts** C
+ **0 pts** D
+ **0 pts** E
+ **0 pts** F
+ **0 pts** G
✓ + **3 pts** **H**
+ **0 pts** I
+ **0 pts** J
+ **0 pts** K

QUESTION 5

5 Binary Tree Traversal - Diagram 10 / 10

✓ + **2 pts** **Contains all 10 data**
✓ + **2 pts** **TA is last**
✓ + **3 pts** **AT is before AB (both must be present)**
✓ + **3 pts** **At least 9 nodes are present, and order is correct among those nodes: EM, RP, JA, AS, AT, ST, RG, DW, AB, TA**
+ **1 pts** Completely correct pre-order traversal (rubric items 1, 3, 5) -- TA, AT, RP, EM, AS, JA, AB, DW, ST, RG
+ **0 pts** Incorrect / No Answer

QUESTION 6

Efficiency - Multiple Choice 15 pts

6.1 A 3 / 3

+ **0 pts** O(1)
+ **0 pts** O(log n)
✓ + **3 pts** **O(n)**
+ **0 pts** O(n log n)
+ **0 pts** O(n^2)

6.2 B 3 / 3

+ **0 pts** O(1)
+ **0 pts** O(log n)
✓ + **3 pts** **O(n)**
+ **0 pts** O(n log n)
+ **0 pts** O(n^2)
+ **0 pts** No Answer

6.3 C 3 / 3

✓ + **3 pts** **O(1)**
+ **0 pts** O(log n)
+ **0 pts** O(n)
+ **0 pts** O(n log n)
+ **0 pts** O(n^2)
+ **0 pts** No Answer

6.4 D 0 / 3

+ **0 pts** O(1)
✓ + **0 pts** **O(log n)**
+ **3 pts** O(n)
+ **0 pts** O(n log n)
+ **0 pts** O(n^2)
+ **0 pts** No Answer

6.5 E 3 / 3

+ **0 pts** O(1)
+ **0 pts** O(log n)
+ **0 pts** O(n)
✓ + **3 pts** **O(n log n)**
+ **0 pts** O(n^2)
+ **0 pts** No Answer

QUESTION 7

7 Linked List - Tracing 10 / 10

✓ + **2 pts** **Ila is in list once**
✓ + **2 pts** **Sanjana is in list once**
✓ + **2 pts** **Walker is in list once**
✓ + **1.5 pts** **An Ila node points to a Sanjana node**
✓ + **1.5 pts** **A Sanjana node points to a Walker node**
✓ + **0.5 pts** **All Sanjana nodes have 1332 appended**
✓ + **0.5 pts** **All Walker nodes have 1332 appended**
- **1 pts** Tim exists
- **3 pts** Extra non-Ila-Sanjana-Tim-Walker data added
+ **0 pts** Incorrect/ No Answer
- **1 pts** Ila has a 1332 appended

QUESTION 8

8 Recursion - Tracing 10 / 10

✓ + **2 pts** **Letters o, i, a, s, d printed**
✓ + **2 pts** **Letters e, p, c, t printed**
✓ + **2 pts** **At least one letter is printed from each of the first two rubric items. No letters from the second**

rubric item are printed before letters from the first rubric item

✓ + **2 pts** At least 4 of "oiasd" are printed, and they are in order

✓ + **2 pts** At least 3 of "epct" are printed, and they are in order

- **1 pts** Any letter printed more than once

+ **0 pts** Incorrect

QUESTION 9

9 Bonus - Arithmetic 1 / 0

✓ + **1 pts** 3

+ **0 pts** Blank

+ **0 pts** Not 3

gradescope

Copyright © 2019. All rights reserved. Unauthorized reproduction, distribution, or transmission of this document is prohibited.

CS 1332 Exam 1A
Spring Semester 2019 - February 6, 2019

Name (print clearly including your first and last name): Bingyao Wang

Signature: Bingyao Wang

GT account username (msmith3, etc): bwang404

GT account number (903000000, etc): 903406074

- ***You must have your BuzzCard or other form of identification on the table in front of you during the exam. When you turn in your exam, you will have to show your ID to the TAs before we will accept your exam. It is your responsibility to have your ID prior to beginning the exam.***
- ***You are not allowed to leave the exam room and return. If you leave the room for any reason, then you must turn in your exam as complete.***
- ***Signing and/or taking this exam signifies you are aware of and in accordance with the Academic Honor Code of Georgia Tech and the Georgia Tech Code of Conduct.***
- ***Notes, books, calculators, phones, laptops, smart watches, headphones, etc. are not allowed.***
- ***Extra paper is not allowed. If you have exhausted all space on this test, talk with your instructor. There are extra blank pages in the exam for extra space.***
- ***Pens/pencils and erasers are allowed. Do not share.***
- ***All code must be in Java.***
- ***Efficiency matters. For example, if you code something that uses O(n) time or worse when there is an obvious way to do it in O(1) time, your solution may lose credit. If your code traverses the data 5 times when once would be sufficient, then this also is considered poor efficiency even though both are O(n).***
- ***Style standards such as (but not limited to) use of good variable names and proper indentation is always required. (Don't fret too much if your paper gets messy, use arrows or whatever it takes to make your answer clear when necessary.)***
- ***Comments are not required unless a question explicitly asks for them.***

Do not write below this line. It may not be scanned, therefore it will not be graded.

Copyright © 2019. All rights reserved. Unauthorized reproduction, distribution, or transmission of this document is prohibited.

This page is purposely left blank. You may use it for extra space, just mention on the page of the question that you want work here to be graded.

Copyright © 2019. All rights reserved. Unauthorized reproduction, distribution, or transmission of this document is prohibited.

1) Circular Linked List - Coding [15 points]

Goal: ***Given a basic class for a* Circular Singly Linked List, *write an* addToFront *method. As the name suggests, this method should add data to the front of the list such that the head contains the new data, head.next contains the data from the old head, and the last node points to the new head.***

Requirements: ***If the data passed into the method is null, throw a java.util.IllegalArgumentException. Since*** `Node` ***is an inner class, you should access fields directly (eg. node.data) instead of using getters/setters. This method should run in O(1) time (keep in mind that there is no tail pointer).***

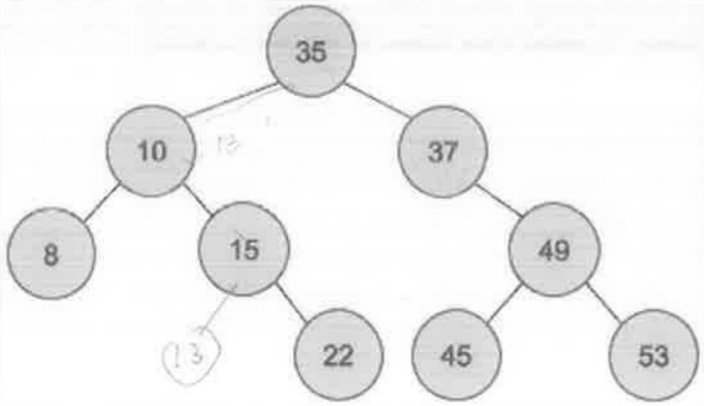
```
import java.util.IllegalArgumentException;
public class LinkedList<T> {

    private static class Node<T> {
        public T data;
        public Node<T> next;
        public Node(T data, Node<T> next) { ... }
    }

    private Node<T> head;
    private int size;

    /**
     * Adds data to the front of the list.
     * @param data the element to add to the head
     * @throws java.util.IllegalArgumentException if data is null
     */
    public void addToFront(T data) {
        if (data == null ){
        throw new java.util.IllegalArgumentException("Can't add null");
        }
        Node<T> temp = new Node (head. data, head. next);
        head. next = temp;
        head. data = data;
        size ++;

    }
}
```

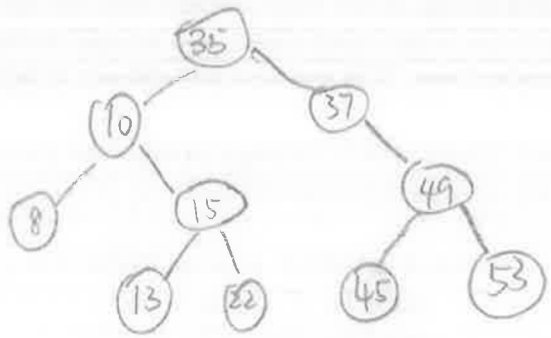
Copyright © 2019. All rights reserved. Unauthorized reproduction, distribution, or transmission of this document is prohibited.

2) Binary Search Trees - Diagramming [15 points]

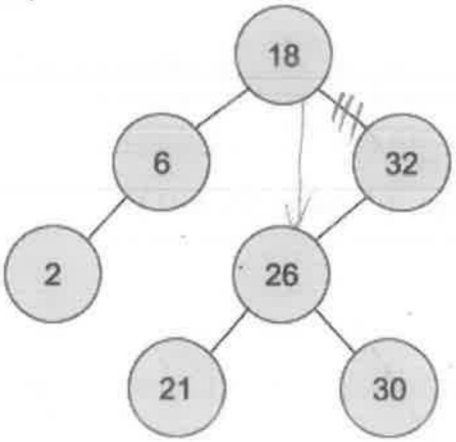
Given the following initial BSTs in the left column below, perform the stated operation, add or remove, for each tree. Draw the resulting BST in the right column. If you want, you can draw multiple steps (circle the final step if you do so). If necessary for any operation, use the predecessor node.

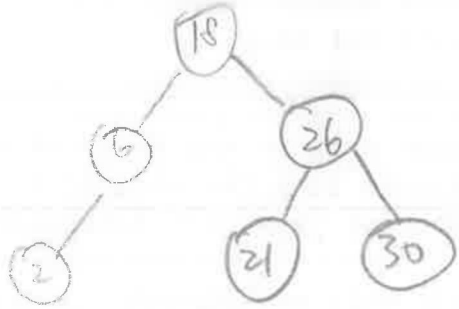
*add(**13**)*



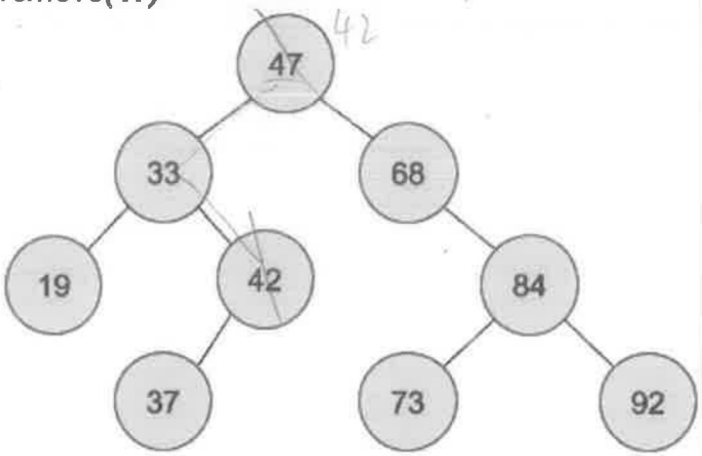


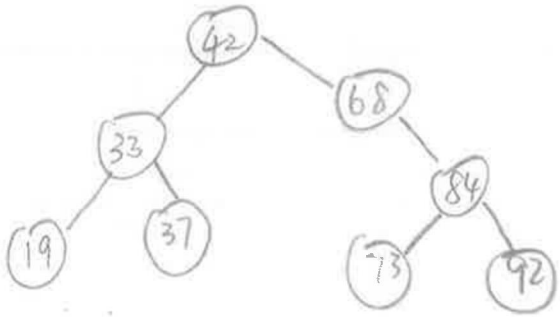
*remove(**32**)*





*remove(**47**)*





Copyright © 2019. All rights reserved. Unauthorized reproduction, distribution, or transmission of this document is prohibited.

3) Stacks and Queues - Lines of Code [10 points]

Given the following lines of code, determine which are part of the implementation of the `push()` ***and*** `dequeue()` ***methods for either the*** `ArrayStack`, `ArrayQueue`, `LinkedStack` ***or*** `LinkedQueue` ***classes in your homework. Pair each segment of code to the letter of their corresponding class below. For all methods, you can assume the class contains data of type*** `T`, ***that it contains any appropriate member variables such as*** `size`, `head` ***and*** `tail`, ***and that a*** `data` ***parameter of type*** `T` ***is given if needed.***

A. `ArrayStack`

B. `ArrayQueue`

C. `LinkedStack`

D. `LinkedQueue`

A I.
```
array[size] = data;
size++;
```
push

D II.
```
T element = head.getData();
head = head.getNext();
size--;
return element;
```
dequeue

B III.
```
T element = array[front % array.length];
array[front % array.length] = null;
front++;
size--;
return element;
```
dequeue

C IV.
```
head = new LinkedNode<T>(data, head);
size++;
```
push

Copyright © 2019. All rights reserved. Unauthorized reproduction, distribution, or transmission of this document is prohibited.

4) Scenario Matching [15 points]

For each of the scenarios listed below, determine the* most appropriate *data structure to use (not the backing structure). Write the letter corresponding to your choice in the space provided. Answer choices may be used once, multiple times, or not at all.

G I. *You work at a bookstore called Narnes and Boble, and around you are large towers of books. You are developing an application to track the movement of individual books in these towers, from which books can only be added or removed from the top.*

B II. *An operating system has an id for each process ranging from 0 to N. It needs to keep track of the amount of memory each process is using. New processes are added occasionally, and the capacity of the data structure will change accordingly. The OS needs to quickly lookup the amount of memory an arbitrary process i uses. What data structure should the OS use to keep track of the amount of memory each process is using?*

K III. *You're a fresh new software engineer at Headpage, and you want to find out how many levels of management are between you and Zark Muckerberg. There happens to be an organization chart that can help you here. It shows all the executives who work directly under Muckerberg, all the people who work for those executives, down to all the engineers who work for your manager. What data structure should represent this organization chart?*

F IV. *Spotifly is a hip new music player. Users can have playlists of songs that they enjoy. These playlists have a Next button to play the next song in the playlist, and a Previous button to play the previous song. Pressing the Next button on the last song in the playlist takes the user to the first song, and pressing the Previous button on the first song takes the user to the last song. What data structure should store the songs in a playlist?*

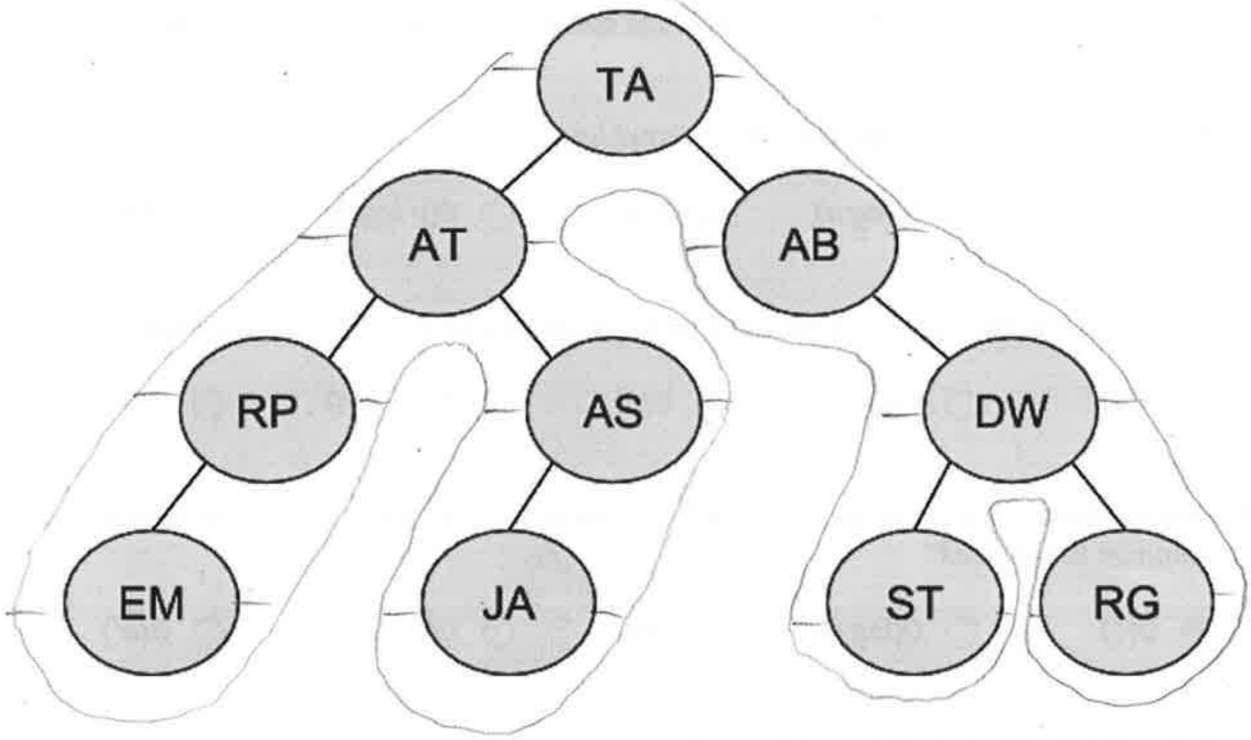
H V. *Gech University uses the CommonApp service to process applications. The university uses rolling admission, meaning that it reviews applications in the order they are submitted. What data structure should CommonApp use to store applications?*

- A. *Array*
- B. *ArrayList*
- C. *Singly-linked List*
- D. *Doubly-linked List*
- E. *Circular-Singly Linked List*
- F. *Circular-Doubly Linked List*
- G. *Stack*
- H. *Queue*
- I. *Binary Search Tree*
- J. *Binary Tree*
- K. *Tree*

Copyright © 2019. All rights reserved. Unauthorized reproduction, distribution, or transmission of this document is prohibited.

5) Binary Tree Traversal - Diagram [10 points]

Given the following binary tree containing hierarchical data, perform a* post order *traversal of the tree. Write the data as they appear in a post order traversal.



Post order traversal data:

EM RP JA AS AT ST RG DW AB TA

Copyright © 2019. All rights reserved. Unauthorized reproduction, distribution, or transmission of this document is prohibited.

6) Efficiency - Multiple Choice [15 points]

For each of the operations listed below, determine the time complexity of the operation. Select the bubble corresponding to your choice in the space provided. Unless otherwise stated, assume the worst-case time complexity. However, make sure you choose the tightest Big-O upper bound possible for the operation. Do not use an amortized analysis for these operations unless otherwise specified.

A.) Adding one element to the front of an ArrayList.

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

B.) Removing one element from the back of a Doubly-Linked List without a tail pointer.

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

C.) Dequeuing in a Queue backed by a Circular Singly-Linked List without a tail pointer where you enqueue to the back.

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

D.) Average case of searching in a binary tree.

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

E.) Average case of adding n elements into a BST from an empty tree by calling the add method repeatedly.

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

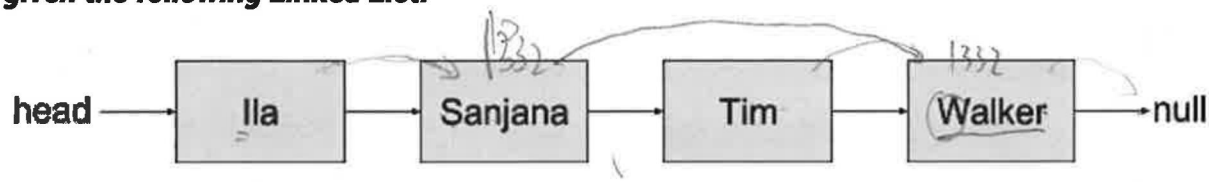
Copyright © 2019. All rights reserved. Unauthorized reproduction, distribution, or transmission of this document is prohibited.

7) Linked List - Tracing [10 points]

The following is a Singly Linked List's Node class and a "mystery" recursive method.

```
public class Node {
      public String data;
      public Node next;
}

public Node grayson(Node node) {
    if (node == null) {
        return null;
    }

    node.next = grayson(node.next);

    if (node.data.length() >= 6) {
        node.data = node.data + "1332";
    }

    if (node.data.charAt(0) != 'T') {
        return node;
    } else {
        return node.next;
    }
}
```

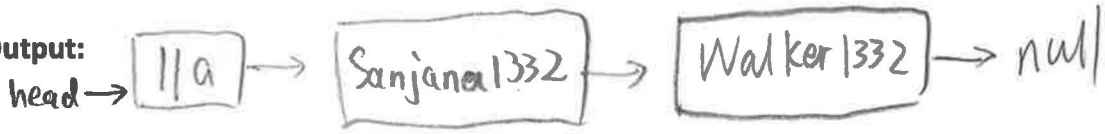
You are given the following Linked List:



Draw the Linked List after the following code is executed:

```
grayson(head);
```

Output:



Copyright © 2019. All rights reserved. Unauthorized reproduction, distribution, or transmission of this document is prohibited.

8) Recursion - Tracing [10 points]

The following is a "mystery" recursive method.

```
public void joshy(char[] arr, int i) {
      if (i < 0) {
            return;
      }

      System.out.println(arr[i]);

      joshy(arr, i - 2);

      if (i - 1 >= 0) {
            System.out.println(arr[i - 1]);
      }
}
```

8 6 4 i=2 i=0
o i a s d e p c t

8 6

What is output when this code is executed?

```
char[] arr = { 'd', 'e', 's', 'p', 'a', 'c', 'i', 't', 'o' };
joshy(arr, arr.length - 1);
```

Output:

'o'
'i'
'a'
's'
'd'
'e'
'p'
'c'
't'

Copyright © 2019. All rights reserved. Unauthorized reproduction, distribution, or transmission of this document is prohibited.

9) Bonus - Arithmetic [1 point]

Evaluate the following expression (you may, but are not required to, use the full page for your work):

19

(8 + 5 + 6) % 8

3

Copyright © 2019. All rights reserved. Unauthorized reproduction, distribution, or transmission of this document is prohibited.

This page is purposely left blank. You may use it for extra space, just mention on the page of the question that you want work here to be graded.