95-771 Data Structures and Algorithms Midterm Exam
Spring 2019

Print Full Name____________________________ Andrew ID Key________________

Tree Problem Code (15 points)

Consider the following class named TreeNode.java.

```
package midterm;

public class TreeNode {

   private int data;
   private TreeNode lc,rc;

   public TreeNode(TreeNode lc, int data, TreeNode rc){
      this.data = data;
      this.lc = lc;
      this.rc = rc;
   }

   @Override
   public String toString() {
      String LC = "";
      String RC = "";
      if (lc == null) LC = "NULL<-";
      else LC = "<--";
      if (rc == null) RC = "->NULL";
      else RC = "-->";
      String s = LC + data + RC;
      return s;
   }

   public int getData() {
      return data;
   }
   public TreeNode getLc() {
      return lc;
   }

   public TreeNode getRc() {
      return rc;
   }

   public void setData(int data) {
      this.data = data;
   }

   public void setLc(TreeNode lc) {
      this.lc = lc;
   }
   public void setRc(TreeNode rc) {
      this.rc = rc;
   }
```

```
    if(y == null) {
      tree = z;
    }
    else {
      if (z.getData() <= y.getData()) {
        y.setLc(z);
      }
      else {
        y.setRc(z);
      }
    }
    z.setLc(null);
    z.setRc(null);
  }
  public boolean contains(int v) {

    TreeNode x = tree;
    while(x != null) {
      if (v == x.getData()) return true;

      if (v <= x.getData()) {
        x = x.getLc();
      }
      else {
        x = x.getRc();
      }
    }
    return false;
  }

  public static void main(String[] args) {
    Tree t = new Tree();
    t.insert(34);
    t.insert(100);
    t.insert(300);
    t.insert(300);
    t.insert(5);
    t.insert(2);
    t.insert(1);
    t.postOrderTraversal();
    if(t.contains(100)) System.out.println("Found 100");
    else System.out.println("No 100 found");
  }
}
```

2) Show the eight lines of output here. (1.5 Points per line = 12 points)

```
NULL<--1-->NULL
<--2-->NULL
<--5-->NULL
NULL<--300-->NULL
<--300-->NULL
NULL<--100-->
<--34-->
Found 100
```

```
        public static void main(String args[]) {
            TreeNode t = new TreeNode(null, 23, null);

            System.out.println(t);

            TreeNode l = new TreeNode(null,100,null);
            TreeNode r = new TreeNode(null,200,null);
            t.setLc(l);
            t.setRc(r);

            System.out.println(t);
        }
    }
```

1. There are two lines of output from TreeNode.java. Each is printed with a println() statement. What is the exact output of these two lines?

   NULL<--23-->NULL (1.5 Points)

   <--23--> (1.5 points)

Consider the following class that uses TreeNode shown above. There are eight lines of output from this program. Note that the comparison used during insert() and contains() is "<=" and not "<" .

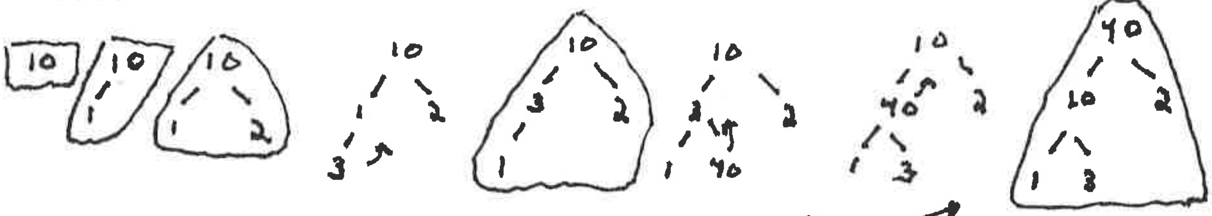
```
package midterm;
public class Tree {

  private TreeNode tree;

  public Tree() {
    tree = null;
  }
  public void postOrderTraversal(TreeNode t) {
    if(t != null) {
      postOrderTraversal(t.getLc());
      postOrderTraversal(t.getRc());
      System.out.println(t);
    }
  }
  public void postOrderTraversal(){
    postOrderTraversal(tree);
  }
  public void insert(int value) {
    TreeNode y = null;
    TreeNode x = tree;
    TreeNode z = new TreeNode(null, value, null);
    while(x != null) {
      y = x;
      if (z.getData() <= x.getData()) {
        x = x.getLc();
      }
      else {
        x = x.getRc();
      }
    }
```

Key

Heaps (15 points)

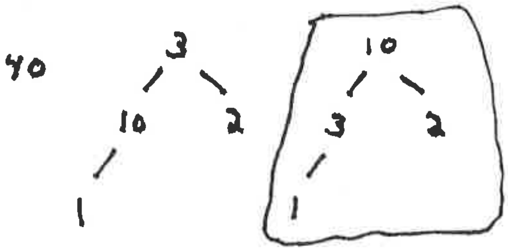
3) Insert the following numbers into a max heap. Draw a new tree for each heap insertion. (5 Points)

10,1,2,3,40



4) What is the height of the tree that you drew in question 3? (2 Points) 2

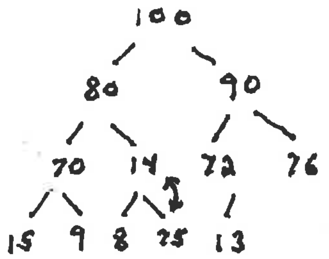
5) Perform a single delete operation on the heap that you drew in question 3. Draw the resulting tree. (3 Points)



KeY

6) Consider the following max heap implemented in an array. It is not quite correct. To make it a max heap exactly one swap of parent and child must occur. What two numbers need to be swapped in order to make this a max heap? (5 points) It is fine to just circle the two numbers.

100
80
90
70
14
72
76
15
9
8
75
13



Binary Trees (16 points)

7. Parts (a), (b), and (c) refer to the following binary tree:

```
              55
           /      \
        40          90
       /  \           \
      6   45           76
     / \              /  \
    5   7           65    7
```

(a) List the data that would be accessed by a pre-order traversal on the given tree by writing out the values in the nodes as they would be accessed, separated by commas. (3 points)

55, 40, 6, 5, 7, 45, 90, ~~27~~ 76, 65, 7

(b) List the data that would be accessed by an in-order traversal on the given tree by writing out the values in the nodes as they would be accessed, separated by commas. (2 points)

5, 6, 7, 40, 45, 55, 90, 65, 76, 7

(c) List the data that would be accessed by a post-order traversal on the given tree by writing out the values in the nodes as they would be accessed, separated by commas. (2 points)

5, 7, 6, 45, 40, 65, 7, 76, 90, 55

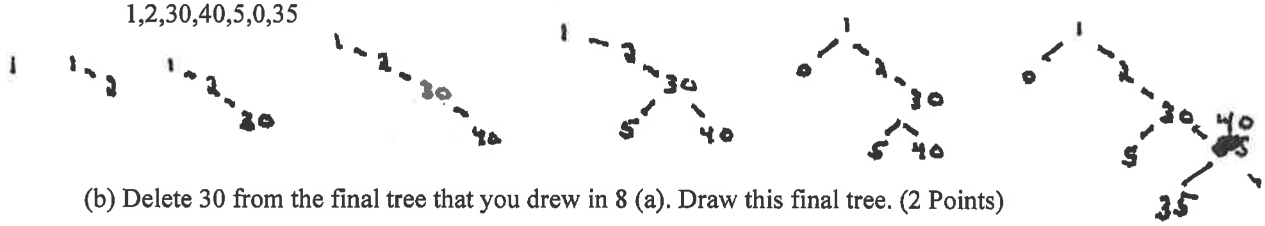
(d) In general, if a binary tree is perfectly balanced (unlike the tree pictured here) and complete with height h, how many leaves, in terms of h, will the tree have? (2 points) 2^h Note, this tree has a perfectly flat bottom.

(e) In general, if a binary tree is perfectly balanced (unlike the tree pictured here) and complete with exactly k leaves. What is the height (in terms of k) of this tree? (2 points) $\text{Log}_2 K$ Note, this tree has a perfectly flat bottom.

wrong base or no base -1

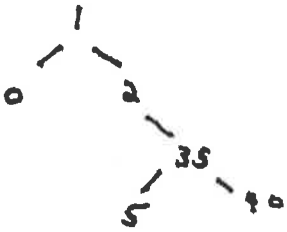
8. (a) Insert he following numbers into a Binary Search Tree. Draw the tree after each insertion. (3 Points)

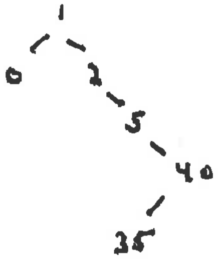
1,2,30,40,5,0,35



(b) Delete 30 from the final tree that you drew in 8 (a). Draw this final tree. (2 Points)

two possible answers





KEY

Project Questions (18 points)

9. Recall the Merkle-Hellman cryptosystem and the Merkle tree that we worked with in Project 1, the spell checker application and the dynamic programming exercise in Project 2, and the calculator problem from Project 3.

Not required to show.

The Merkle-Hellman cryptosystem in Project 1 was based on the subset sum problem which is known to be NP-Complete. The problem itself can be described as follows: given a set of numbers *X* and a number **k**, is there a subset of *X*, which sums to k?

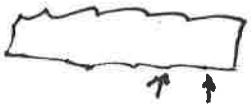
(a) Suppose X = {4, 16, 3, 9, 2, 8, 5} and **k** = 41. Is there a subset of X which sums to k?
Yes Yes/No (2 points)

(b) Suppose Alice sends messages to Bob encrypted with Bob's Merkle-Hellman public key. Circle the one statement that is true? (2 Points)

[✓ 1.] Bob decrypts with a super increasing sequence.
2. Alice encrypts with a super increasing sequence.
3. Alice decrypts with a super increasing sequence.
4. Bob decrypts with a set such as X in part a.
5. Alice decrypts with a set such as X in part a.

(c) Write a method in Java that returns true if there is a subset of X which sums to k and false otherwise. The method signature and pre-conditions are provided. Note, the array x is a super increasing sequence. (6 points)

```
public static boolean subSetSum(int[] x, int n, int k) {
// pre: x is a super increasing and x[0] < x[1] <x[2] <...<x[n-1].
// pre: n is the size of x.
// pre: all integers involved are small enough that overflow is not of concern
// post: returns true if some subset of x sums to k, false otherwise.
 while ( K > 0 && N > 0 ) {
        IF (x[N-1] <= K) K = K - x[N-1];
        N = N - 1;
 }
 IF (K==0) return true;
 else return FAlse;
```



(d) What is the worst case run time complexity of your code in part (c)? Use Big Theta. (1 Points) $\Theta(N)$

(e) In project 1 we wrote a recursive function that computed probabilities of a World Series win. Briefly describe the run time performance of this function. (1 Point)

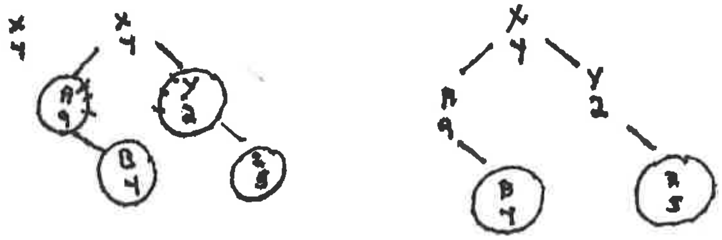
IT TOOK TOO LONG TO EXECUTE FOR LARGE VALUES OF N.
IT RAN IN EXPONENTIAL TIME.

No values -2

(f) In Project 3, we wrote a calculator that processed RPN expressions and used a Red Black Tree. Draw what the Red Black Tree would look like after the following user interaction. Circle RED nodes and leave BLACK nodes un-circled. (4 Points)

Key

```
X 4 =
A 9 =
Y 2 =
B Y 2 + =
Z 5 =
```



wrong coloring -3

wrong shape -2

max = -4 off

(g) In Project 2 we wrote a spell checker that loaded n words into a Red Black tree and allowed a user to make queries against the tree. We wrote a lookup method that checked if a word was present. Which of the following is true of the best-case lookup method? Circle all correct answers. (2 Points)

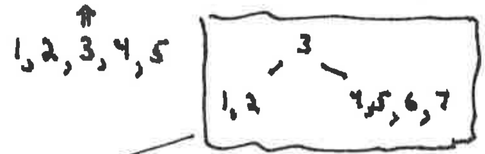
[✓ 1.] It ran in O(LogN)
[✓ 2.] It ran in O(1)
3. It ran in Ω(N^2)
4. It ran in Θ(N)
5. It ran in Θ(LogN)
[✓ 6.] It ran in Θ(1)

1 pt off per answer max = -2

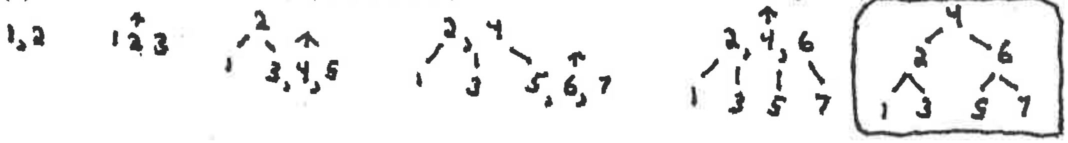
Balanced Trees (15 points)

10. B-Trees

(a) Insert these numbers into a B-Tree with min = 2.
1,2,3,4,5,6,7
Draw the final tree. (2 points)



(b) Insert the same numbers (1,2,3,4,5,6,7) into a B-Tree with min = 1. (2 points)

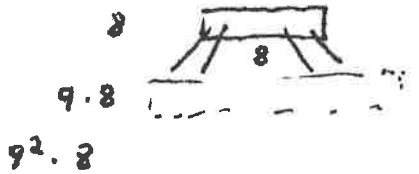


(c) What is the height of the B-Tree in question 1~~0~~(a)? 1 (1 Point)

(d) What is the height of the B-Tree in question 1~~0~~(b)? 2 (1 Point)

(e) What is the maximum number of keys a B-Tree can hold with min = 4 and height = 2? ____ (2 Points)

$= 8 + (9 \cdot 8) + (9^2 \cdot 8) = 728$



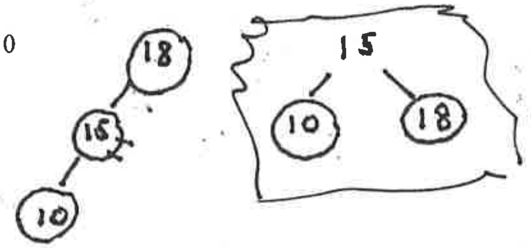
1 node 8 keys per node = 8 keys
9 nodes 8 keys per node = 9·8 keys

Key

11. Red-Black Trees

(a) Insert the following numbers, one by one, into a Red-Black Tree. Show the tree after each insertion. red vertices should be circled and black vertices should appear without circles. (2 points)

18, 15, 10



Poor notation -1

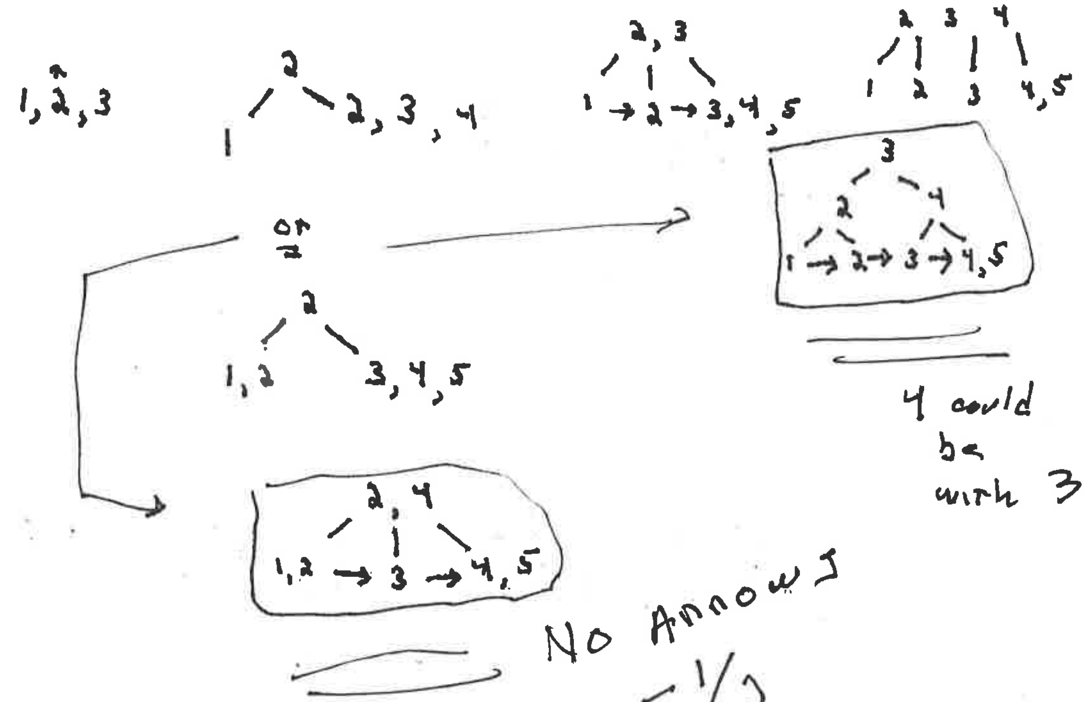
(b) What is the runtime complexity of an inorder traversal of a Red Black Tree ? Use Big Theta notation. (1 Point) $\Theta(N)$

(c) What is the worst-case runtime complexity of a Red Black Tree lookup operation? Use Big Theta notation. (1 point) $\Theta(\log N)$

12. B+ Trees

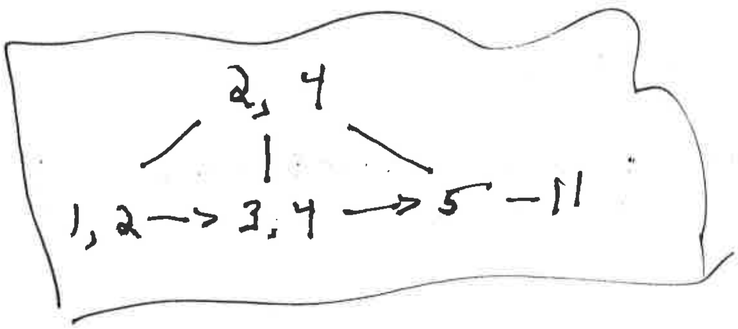
Insert the following numbers into a B+ Tree with minimum = 1. (3 Points)
1,2,3,4,5

Poor ✓



4 could be with 3

No Arrows
-1/2

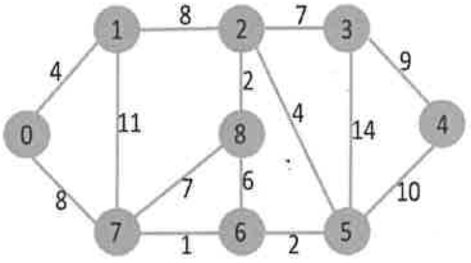


-1

Key

Graph Representations (14 points)

13. (a) Represent the following graph in an adjacency matrix representation. Show your matrix. (7 Points)



$$\begin{array}{c|ccccccccc}
 & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\
\hline
0 & & 4 & & & & & & 8 & \\
1 & 4 & & & & & & & 11 & \\
2 & & & & & & & & & \\
3 & & & & & & & & & \\
4 & & & & & & & & & \\
5 & & & & & & & & & \\
6 & & & & & & & & & \\
7 & 8 & 11 & & & & & & & \\
8 & & & & & & & & &
\end{array}$$

etc.

No distances
-3

13. (b) Represent the graph above as an adjacency list. Draw a sketch of this representation. (7 Points)

0 → 1 (4) → 7 (8)
1
2
3
4
5
6
7 → 0 (8) → 8 (7) → 6 (1) → 1 (11)
8

etc.

No distances
-3

Stacks (7 points)

14. Write a class Stack. The stack data will live in an array of size 5000. You will provide a constructor, a method named addToStack, a method named removeFromStack, a method named isEmpty, and a method named isFull. This must be written in Java and the syntax must be good but not perfect. For full credit, be exceptionally neat.

Sloppy -1

```
public class Stack {

    int S[];
    int i;
    public Stack() {
        S = new int[5000];
        i = -1;
    }
    // pre: stack not full
    void addToStack (int x){
        i++
        S[i] = x
    }
    // pre: stack not empty
    int removeFromStack() {
        int t = S[i];
        i--;
        return t;
    }

    boolean isEmpty() {
        return(i == -1);
    }
    boolean isFull() {
        return(i == 4999)

    }
}
```