# Data Structures and Algorithms

# INFO 6205

# Homework 6

# Due: June 21, 2020

Put all your java, compiled class files and documentation files into a zip file named Homework6.zip and submit it via the dropbox on the blackboard before the END of due date. Put your name on all .java files. There will be a short quiz on this homework.

1. Consider str=“BooksToRead”. Write algorithm (psuedocode) to reverse the

string using: a) String operations, and b) StringBuilder operations. Provide the running time

complexity for each case.

2. Consider the following string: “abdcceddfcabbeddccefddaaf”

a) What is R value?

b) Use key-indexed counting sort algorithm to sort the string. Show each steps and the results.

c) What is the running time complexity of the algorithm as compared to selection sort?

d) Write the java code to sort the string using steps described in (a).

3. What is the Balanced Tree, Complete Tree and Non-Complete Tree?

4. Consider following data: {9,23,45,1,5,14,55,24,13,11,8,19,4,31,35,56}

a) Construct Binary Tree

b) Construct 2-3 Tree

c) Construct 2-3-4 Tree

d) Construct Binary Heap Tree

e) What is Time complexity of each case, Why would you use one versus the other?

f) Insert 17, 22, 32, 6, 33 in (b)

g) Delete 13 in (a) and (b)

h) What is the Height of (a), (b), (c)?

i) Write Java Search and Insert code for (a) and (b)

j) Write Java code for DeleteMin() and DeleteMax Algorithms for (a), provide example

5. Class Record is described below.

A) Write Java code to build BinaryTree, 2-3 Tree, 2-3-4 Tree where “key” is the value in

data presented in problem-4 (a), (b), (c) parts.

B) Build AVL Tree for data presented in problem-4 (a). What change do you have to make to class

Record to make AVL work?

public class Record {

private int key

private Node leftNode;

private Node rightNode;

public Record(int key, Node leftNode, Node rightNode) {

this.key = key;

this.leftNode = leftNode;

this.rightNode = rightNode;

}

public Record(int key){

this.key = key;

}

public int getKey() {

return key;

}

public Node getLeftNode() {

return leftNode;

}

public Node getRightNode() {

return rightNode;

}

public void setKey(int key) {

this.key = key;

}

public void setLeftNode(Node leftNode) {

this.leftNode = leftNode;

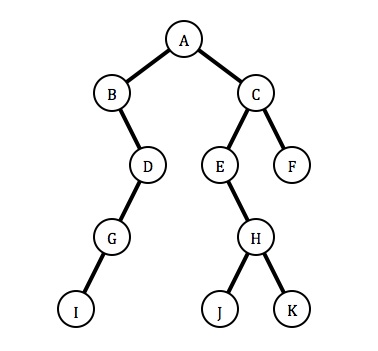
}

public void setRightNode(Node rightNode) {

this.rightNode = rightNode; }

}

6. Consider the following Binary tree, write Java code to find **maximum** element in binary search tree. You may write either a recursive or iterative implementation.



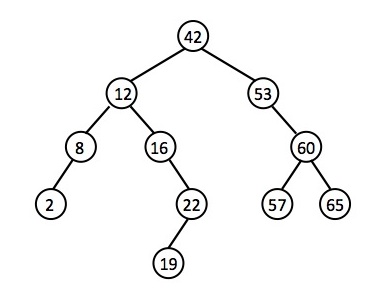
7. Consider this input array data: {30, 40, 23, 58, 49, 26, 11, 13}

a) Insert the following items into an empty binary search tree in order:

b) What is the maximum height of a binary search tree? why?

c) What is the time complexity of the Tree? why?

8. Consider the following binary tree:



Use BST functions deleteKey, deleteRec, and minValue discussed in class: Show

step-by-step code logic to delete the following nodes and redraw the binary tree

for each deletion: delete 8, delete 2, delete 42

9. Does Thread synchronization works correctly with the following code? Why or

Why not? If Not, how do you fix it?

class Table{

synchronized void printTable(int n){//synchronized method

for(int i=1;i<=5;i++){

System.out.println(n\*i);

try{

Thread.sleep(400);

}catch(Exception e){System.out.println(e);}

}

}

}

class MyThread1 extends Thread{

Table t;

MyThread1(Table t){

this.t=t;

}

public void run(){

t.printTable(5);

}

}

class MyThread2 extends Thread{

Table t;

MyThread2(Table t){

this.t=t;

}

public void run(){

t.printTable(100);

} }

public class TestSynchronization2{

public static void main(String args[]){

Table obj1 = new Table();//only one object

Table obj2 = new Table();//only one object

MyThread1 t1=new MyThread1(obj1);

MyThread2 t2=new MyThread2(obj2);

t1.start(); t2.start(); }}

10. Consider this diagram and add these code to:

String s3=s2;

String s4=s1;

String s5= new String(“Cat”);

String s6=new String(“Dog”);

Redraw Diagram

