Key

95-771 Data Structures and Algorithms
Spring 2016

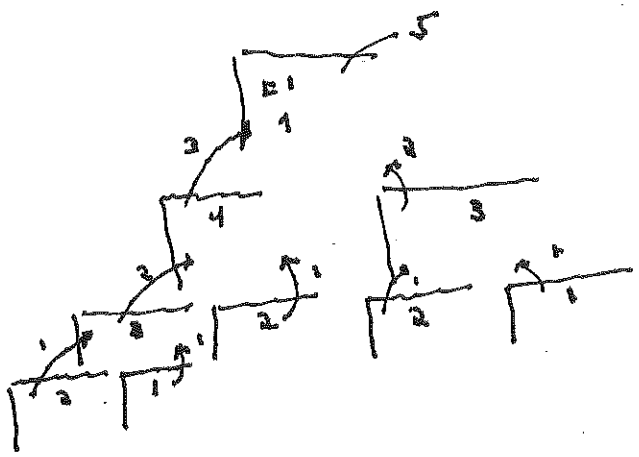
Midterm Exam

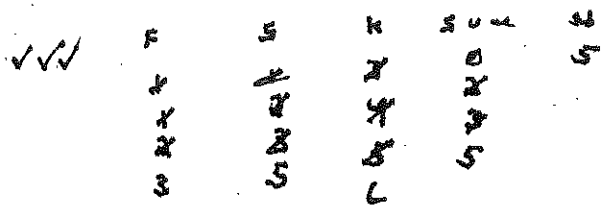
Print Full Name Key Print Andrew ID : Key

Tracing recursion and runtime analysis (18 points)

1. Show the exact output of the following program. (8 points):

```
public class Fibonacci {

   static int constantWorkInFib1 = 0;
   static int constantWorkInFib2 = 0;

   static int fib1(int n) {
      constantWorkInFib1++;
      if (n == 1 || n == 2) return 1;
      else return fib1(n-1) + fib1(n-2);
   }
   static int fib2(int n) {

      int first, second,k,sum = 0;
      if(n == 1 || n == 2) return 1;
      else {
            first = 1; second = 1;
            k = 3;
            while (k <= n) {
                constantWorkInFib2++;
                sum = first + second;
                first = second;
                second = sum;
                k = k + 1;
            }
      }
      return sum;
   }

   public static void main(String[] args) {
      System.out.println("Fib(5) == " + fib1(5));
      System.out.println("Work in fib1 == " + constantWorkInFib1);
      System.out.println("Fib(5) == " + fib2(5));
      System.out.println("Work in fib2 ==" + constantWorkInFib2);
   }
}
```





Show output here:

Fib(5) == 5 (2 Points)
Work in fib1 == 9 (2 Points)
Fib(5) == 5 (2 Points)
Work in fib2 == 3 (2 Points)

KEY

2. The following questions concern the program in Question 1. Let T(n) be the number of constant time operations executed by fib2. Write TRUE or FALSE in each blank. (10 points)

(a) $T(n) \in O(n^2)$ TRUE
(b) $T(n) \in \Omega(n^2)$ FALSE
(c) $T(n) \in \Theta(n^2)$ FALSE
(d) $T(n) \in O(\text{Log}(n))$ FALSE
(e) $T(n) \in \Omega(\text{Log}(n))$ TRUE
(f) $T(n) \in \Theta(\text{Log}(n))$ FALSE
(g) $T(n) \in \Theta(n)$ TRUE

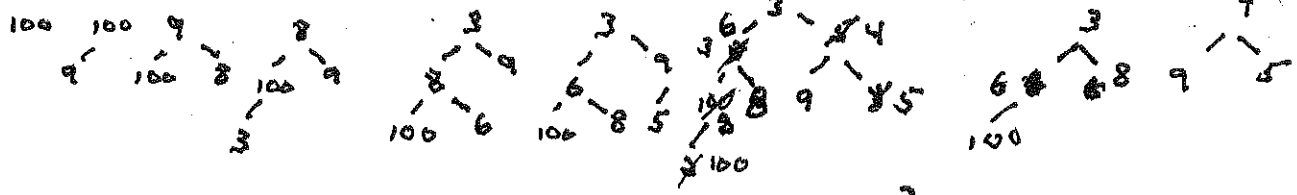
Suppose that $T_2(n)$ represents the number of constant time operations executed by fib1. Write TRUE or FALSE in each blank.

(h) $T_2(n) \in \Omega(\text{Log}(n))$ TRUE
(i) $T_2(n) \in \Omega(n)$ TRUE

Heaps (12 points)

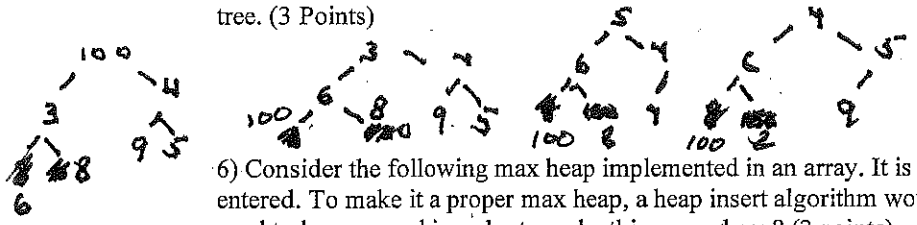
3) Insert the following 8 numbers into a min heap. Draw a new tree for each heap insertion. (4 Points)

100, 9, 8, 3, 6, 5, 4, 2

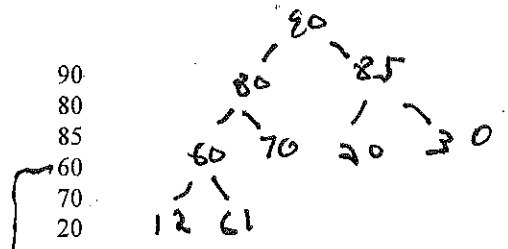


4) What is the height of the tree that you drew in question 3? (2 Points) 3

5) Perform exactly two deleteMin() operations on the heap that you drew in question 3. Draw the resulting tree. (3 Points)



6) Consider the following max heap implemented in an array. It is not quite correct. The 61 has just been entered. To make it a proper max heap, a heap insert algorithm would make one swap. What two numbers need to be swapped in order to make this a max heap? (3 points)



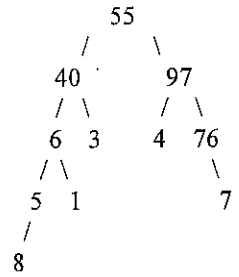
90
80
85
60
70
20
30
12
61

60 and 61

KEY

Binary Trees (16 points)

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



(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, 8, 1, 3, 97, 4, 76, 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)

8, 5, 6, 1, 40, 3, 55, 4, 97, 76, 7

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

55 40 97 6 3 4 76 5 1 7 8

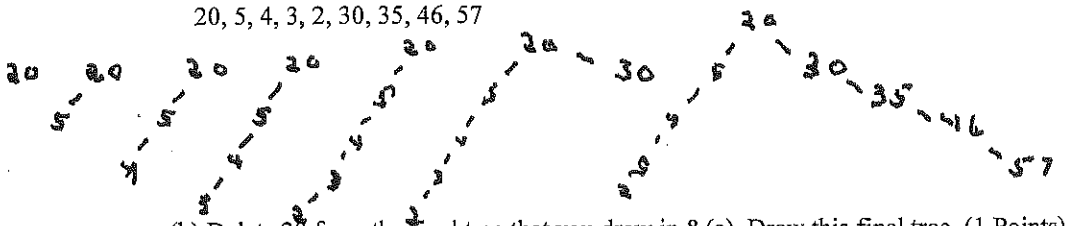
N	h	L
0	0	1
3	1	2
7	2	4

(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? (1 point) 2^h Note, this tree has a perfectly flat bottom. In addition, how many internal nodes would such a tree have (in terms of h)? (1 Point) $2^h - 1$

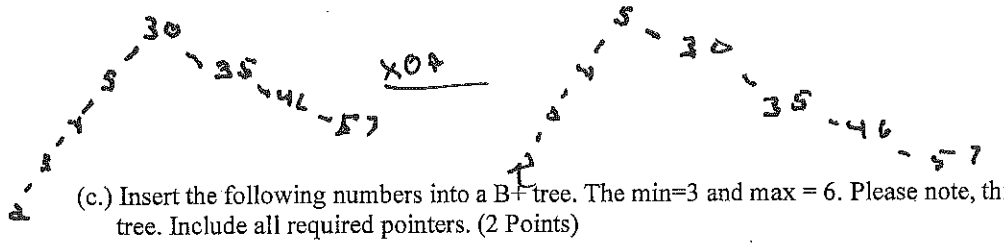
(e) In general, if a binary tree is perfectly balanced (unlike the tree pictured here) and complete with exactly k leaves. What is the height (in terms of k) of this tree? (2 points) $\log_2 K$ Note, this tree has a perfectly flat bottom.

KEY

8. (a) Insert he following numbers into a Binary Search Tree. Draw the tree after all insertions are complete. You need not show each step. (2 Points)
20, 5, 4, 3, 2, 30, 35, 46, 57

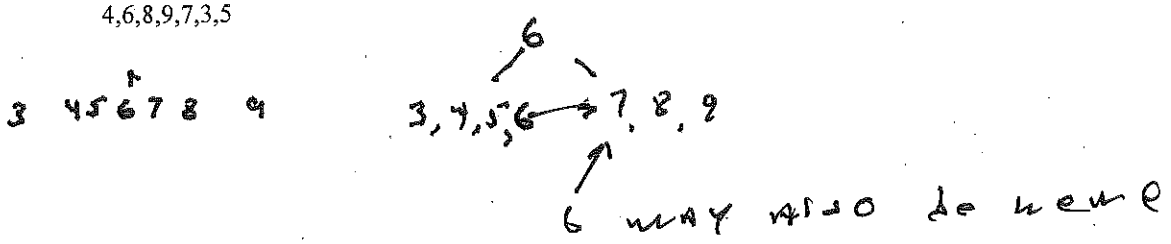


(b) Delete 20 from the final tree that you drew in 8 (a). Draw this final tree. (1 Points)



(c.) Insert the following numbers into a B+ tree. The min=3 and max = 6. Please note, this is a "B Plus" tree. Include all required pointers. (2 Points)

4,6,8,9,7,3,5



KEY

Project Related Question (18 points)

9. Show the exact output of the following program.

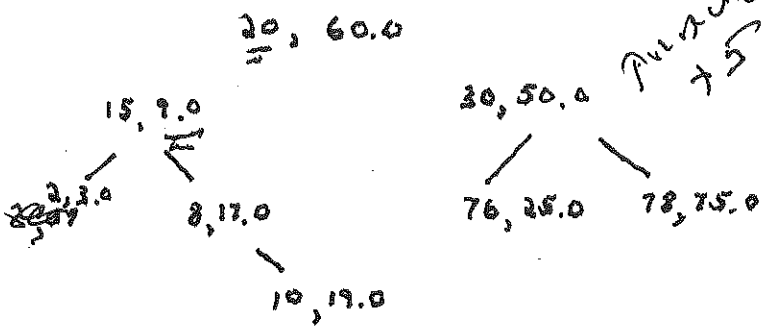
```
class Node {
   public int age;
   public double height;
   public Node lc;
   public Node rc;
   public Node(Node lc, int age, double h, Node rc) {
      this.lc = lc;
      this.age = age;
      this.height = h;
      this.rc = rc;
   }
   public String toString() {
      return "Age = " + age + " height = " + height ;
   }
}

public class SimpleTree {
   public Node root;

   public SimpleTree() {
      root = null;
   }

   public void traversal(Node t) {
      if(t != null) {
         traversal(t.lc);
         System.out.println(t);
         traversal(t.rc);
      }
   }

   public void traversal() {
      traversal(root);
   }
```

KEY

Key

```
public void insert(int age, double height) {

   Node y = null;
   Node x = root;
   Node z = new Node(null,age,height,null);
   boolean twoD = true;
   while (x != null) {
      y = x;
      if(twoD) {
         if (age <  x.age)
            x = x.lc;
         else
            x = x.rc;
         twoD = !twoD;
      }
      else {
         if (height <  x.height)
            x = x.lc;
         else
            x = x.rc;
         twoD = !twoD;
      }
   }
   if (y == null)
      root = z;
   else
     if(!twoD) {
        if (age < y.age)
          y.lc = z;
        else
          y.rc = z;
     }
     else {
        if (height < y.height)
          y.lc = z;
        else
          y.rc = z;
     }
   }

   public static void main(String[] args) {

      SimpleTree st = new SimpleTree();
      st.insert(20, 60.0);
      st.insert(15, 9.0);
      st.insert(30, 50.0);
      st.insert(8, 17.0);
      st.insert(10, 19.0);
      st.insert(76, 25.0);
      st.insert(78, 75.0);
      st.insert(2, 3.0);
      st.traversal();
   }
}
```

Possible partial credit
Tree with fault
+8
Correct
+5
Traversal
+5



Look for partial credit

// SHOW OUTPUT HERE

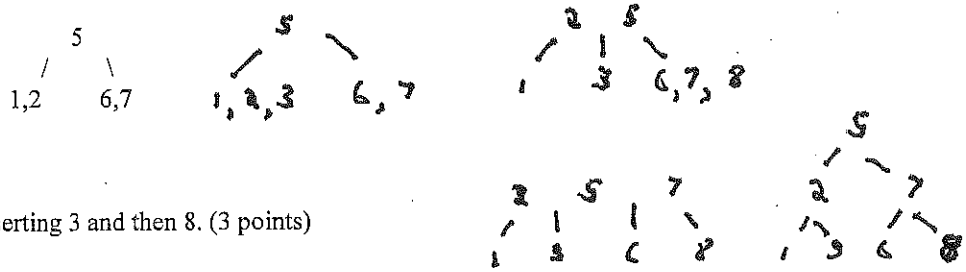
2 3
15 9
8 17
10 19
20 60
76 25
30 50
78 75.0

Allow this in FIN a.

KEY

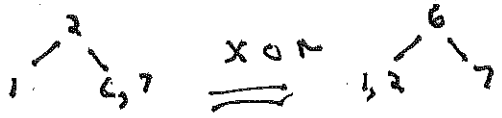
Balanced Trees (15 points)

10. Consider the following B-Tree with a minimum of 1 and a maximum of 2.

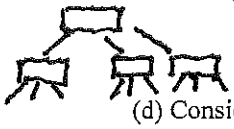


(a) Redraw the tree after inserting 3 and then 8. (3 points)

(b) Delete the value 5 from the B-Tree shown above. Begin work from the original tree, not the tree with the values added in Part a. (2 points)



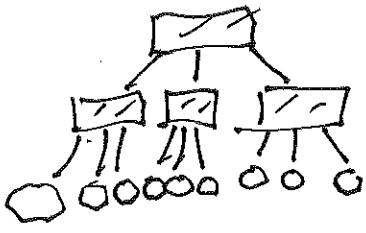
(c) Consider again the original tree of height of 1. What is the maximum number of keys that this type of tree (min = 1, max=2) could hold with a height of 1? (1 points). 8



(d) Consider again the original tree of height of 1. What is the maximum number of keys that this type of tree (min = 1, max=2) could hold with a height of 3? (2 points). 2 + 6 + 18 = 26

26 + 54 = 80

80



Keys	h
2	0
$2+3\cdot2=8$	1
$2+(3\cdot2)+(3^2\cdot2)=$	2
$8+18=26$	
$3^0\cdot2+3^1\cdot2+3^2\cdot2+3^3\cdot2$	3
$=26+54$	
$=80$	

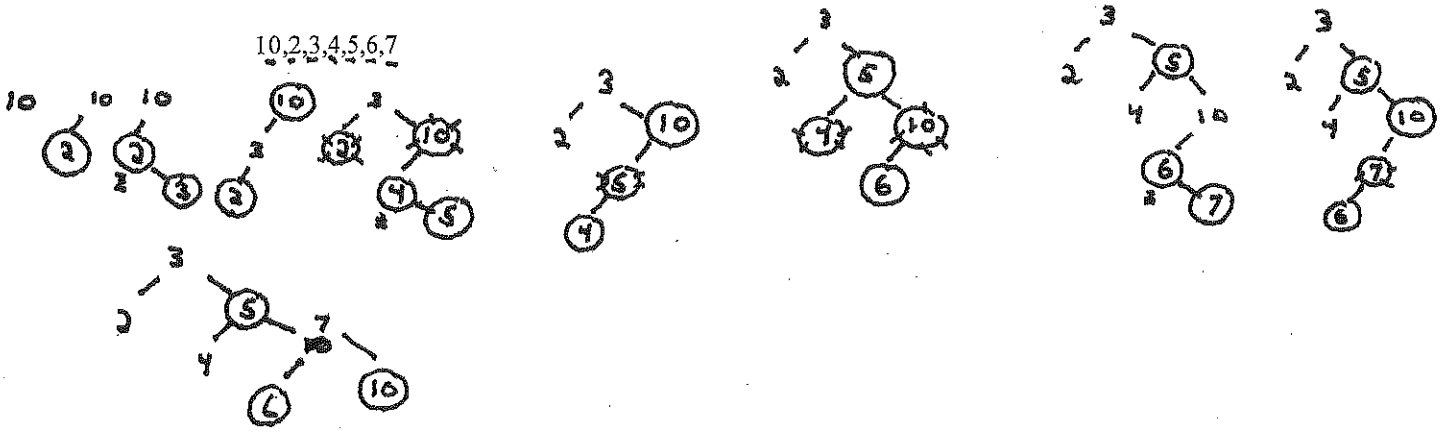
KEY

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. (5 points)

10,2,3,4,5,6,7



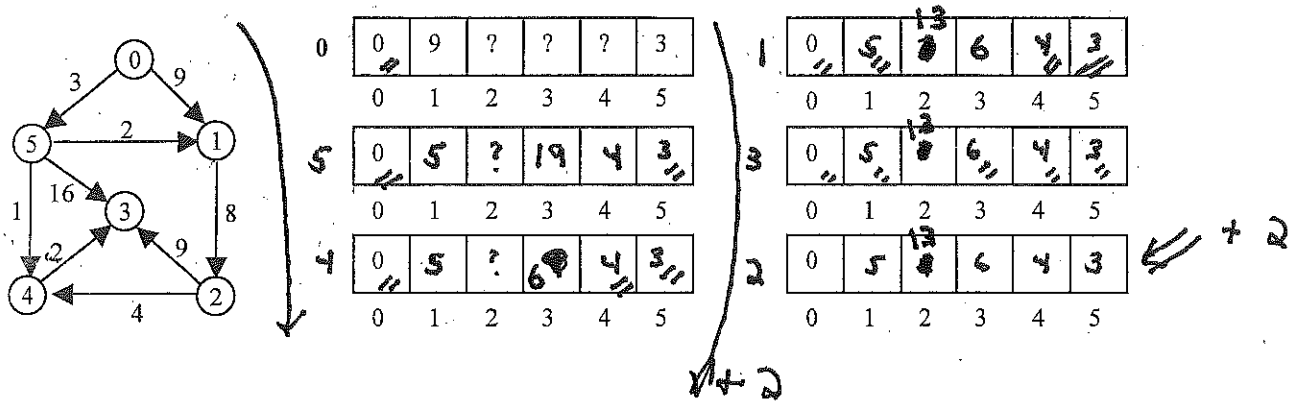
(b) What is the runtime complexity of an inorder traversal of a Red Black Tree ? Use Big Theta notation. (1 Point) $\Theta(N)$

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

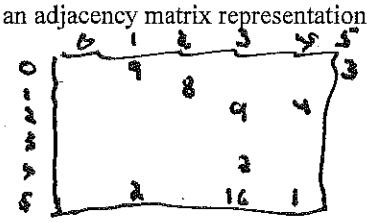
Graph Algorithms(25 points)

12. (a) What is the shortest path from node 0 to node 3 in the graph immediately below? Your path must be a list of ordered pairs.(1 point) (0, 5), (5, 4), (4, 3)

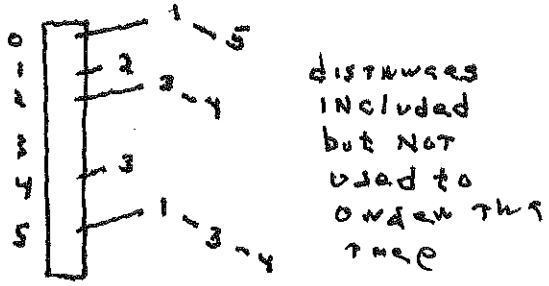
(b) Draw the contents of the distance array for each iteration of Dijkstra's Algorithm as it works on this graph. The initial state is given. **Mark the node to be selected next to the left of the array (note how 0 is marked to the left of the first array.)** Fill in each array cell working downward. That is, complete the first column of arrays before the second column of arrays. (4 Points)



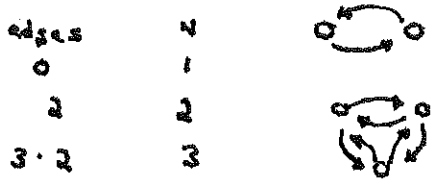
(c) Draw an adjacency matrix representation for the graph shown above. (2 points)



(d) Draw an adjacency set representation of the graph shown immediately above. In this adjacency set representation, each neighbor list is a binary search tree. Be very neat with your drawing. (2 Points)



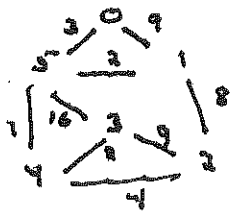
(e) The graph shown above is a simple, directed graph. It contains no loops or multiple edges. Suppose we have such a graph with V vertices. What is the maximum number of edges that such a graph can contain? Your answer should be an exact formula. Answers in Big Theta notation don't count. (1 Point) $N(N-1) = N^2 - N$



KEY

12 b

(f) Redraw the graph in question 14 (a) as an undirected graph. That is, replace each arrow by an undirected edge. The weight of each edge in the undirected graph will be the same as the respective edge weight in the directed graph of 14 (a). (1 Point)



(g) Draw the tree that would be computed by Prim using the graph that you drew in question 14 (f) as input. This is the tree computed by Prim of the undirected, weighted graph you created in 14 (f). (4 points)

(h) Name an appropriate and fast algorithm to determine whether or not there is a path from one vertex to another in a directed or undirected graph. BFS or DFS (2 points)

Dijkstra -1

(i) Using the Floyd Warshall algorithm shown below, draw a matrix each time the comment "draw cost matrix" is encountered in the code. Use the graph below for input. You may assume that the original graph is represented in an adjacency matrix c. Also, you must assume that the loops proceed through the graph in numerical order. That is, the first vertex is vertex 0 and the second is vertex 1 and so on. (4 Points)

```
for each vertex u in G do {
  for each vertex v in G do {
    cost[u,v] = c[u,v]
  }
}

// draw cost matrix in the first box

for each vertex w in G do {
  for each vertex u in G do {
    for each vertex v in G do {
      cost[u,v] = min(cost[u,v],cost[u,w] + cost[w,v]
    } // end for v
  } // end for u
  // draw the cost matrix in the next box
} // end for w
```

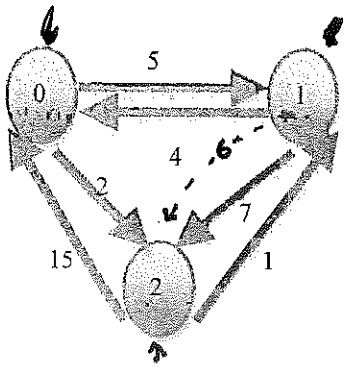
Cost

	0	1	2
0		5	2
1	4		7
2	15	1	

	0	1	2
0		5	2
1	4		6
2	5	1	

	0	1	2
0		5	2
1	4		6
2	5	1	

	0	1	2
0		3	2
1	4		6
2	5	1	



KEY