95-771 Data Structures and Algorithms
Spring 2012

Midterm Exam

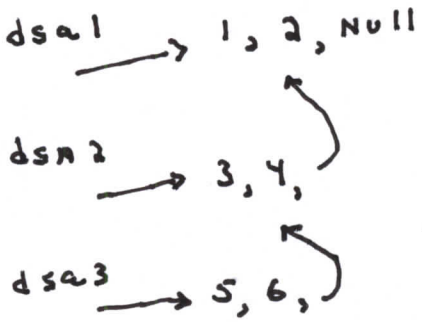
Print Full Name key

Coding Lists and Trees (15 points)

Score
100/100

1. Show the exact output of the following program (5 points):

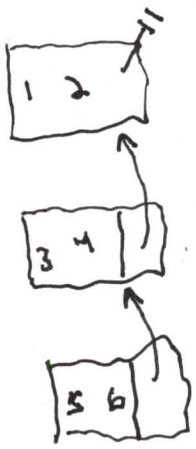
```
public class DSAExam {

  private int a;
  private int b;
  private DSAExam next;

  public DSAExam(int x, int y, DSAExam n) {
   a = x;
   b = y;
   next = n;
  }

  public void print() {
    System.out.print(a + " ");
    System.out.println(b);
    if (next != null) {
      next.print();
    }
  }
  public static void main(String args[]) {

    DSAExam dsa1 = new DSAExam(1,2,null);
    DSAExam dsa2 = new DSAExam(3,4,dsa1);
    DSAExam dsa3 = new DSAExam(5,6,dsa2);
    dsa1.print();
    dsa2.print();
    dsa3.print();
  }
}
```



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

2. Show the exact output of the following program. (5 Points)

```
public class DSAExam {

   private int a;
   private int b;
   private DSAExam next;

   public DSAExam(int x, int y, DSAExam n) {
     a = x;
     b = y;
     next = n;
   }

   public void print() {
     System.out.print(a + " ");
     System.out.println(b);
     if (next != null) {
        next.print();
     }

   }

   int xyz() {
      DSAExam p = this;
      int ctr = 0;
      while (p != null) {
         ctr++;
         p = p.next;
      }
      return ctr;
   }


   public static void main(String args[]) {

      DSAExam dsa1 = new DSAExam(1,2,null);
      DSAExam dsa2 = new DSAExam(3,4,dsa1);
      DSAExam dsa3 = new DSAExam(5,6,dsa2);
      System.out.println(dsa1.xyz());
      System.out.println(dsa2.xyz());
      System.out.println(dsa3.xyz());
   }
}
```



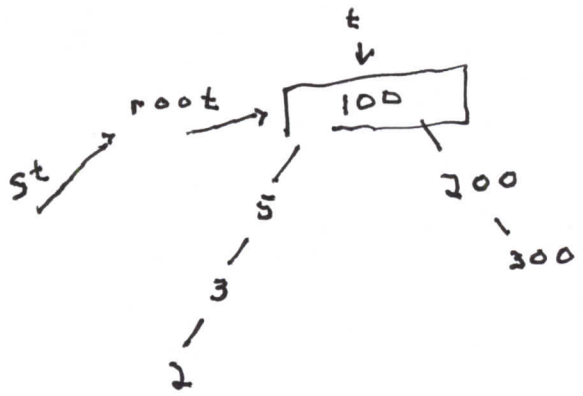
1
2
3

95-771 Data Structures and Algorithms
Spring 2012

3. Show the exact output of the following program. (5 Points)

```
package simpletree;

class Node {
   public int data;
   public Node lc;
   public Node rc;
   public Node(Node lc, int x, Node rc) {
      this.lc = lc;
      this.data = x;
      this.rc = rc;
   }
}

public class SimpleTree {

   public Node root = null;

   public void insert(int x) {

      if (root == null) {
         root = new Node(null,x,null);
      }
      else {
         Node p = root;
         Node q = p;
         while ( p != null) {
            q = p;
            if(x < p.data ) p = p.lc;
            else p = p.rc;
         }
         if(x > q.data)
            q.rc = new Node(null,x,null);
         else
            q.lc = new Node(null,x,null);
      }
   }

   public void traversal(Node t) {

      if(t != null) {
         traversal(t.lc);
         System.out.println(t.data);
         traversal(t.rc);
      }
   }
```

```
public int height(Node t) {

   if(t.lc == null && t.rc == null) return 0;
   else
   {
      if(t.lc == null && t.rc != null) return (1 + height(t.rc));
      if(t.rc == null && t.lc != null) return (1 + height(t.lc));
      if(t.rc != null && t.lc != null) {
            int leftHeight = 1 + height(t.lc);
            int rightHeight = 1 + height(t.rc);
            if (leftHeight >= rightHeight) return leftHeight;
            else return rightHeight;
       }
    }
    return -1;
}

public static void main(String[] args) {

    SimpleTree st = new SimpleTree();
    st.insert(100);
    st.insert(200);
    st.insert(300);
    st.insert(5);
    st.insert(3);
    st.insert(2);
    st.traversal(st.root);
    System.out.println(st.height(st.root));
  }
}
```



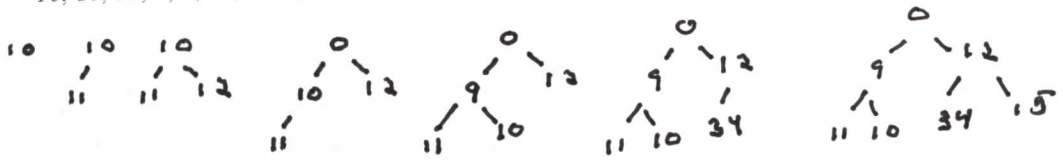
2 3 5 100 200 300

3

Heaps (15 points)

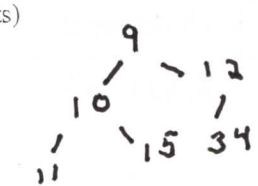
4) Insert the following numbers into a min heap. Draw a new tree for each heap insertion. (5 Points)

10, 11, 12, 0, 9, 34, 15



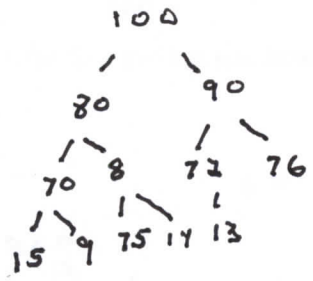
5) What is the height of the tree that you drew in question 4? (2 Points) 2

6) Perform a single delete operation on the heap that you drew in question 4. Draw the resulting tree. (3 Points)



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

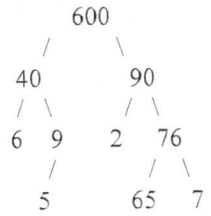
100
80
90
70
8
72
76
15
9
75
14
13



75 and 8

Binary Trees (16 points)

8. 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)

600, 40, 6, 9, 5, 90, 2, 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)

6, 40, 5, 9, 600, 2, 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)

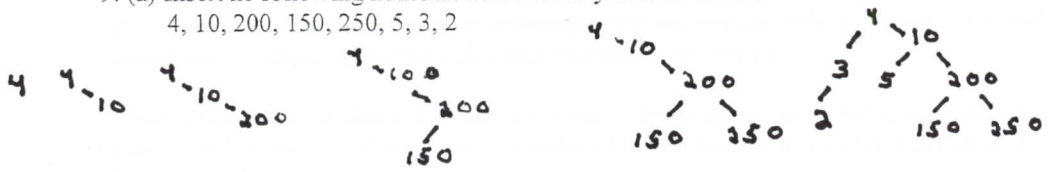
6, 5, 9, 40, 2, 65, 7, 76, 90, 600

L	N
1	1
2	3
4	7

(d) In general, if a binary tree is perfectly balanced (unlike the tree pictured here) and complete with n nodes, how many leaves, in terms of n, will the tree have? (2 points) (N+1)/2

(e) In general, if a binary tree is perfectly balanced (unlike the tree pictured here) and complete with a total of n nodes, then how many non-leaves, in terms of n, will the tree have? (2 points)
(N-1)/2

9. (a) Insert he following numbers into a Binary Search Tree. Draw the tree after each insertion. (3 Points)
4, 10, 200, 150, 250, 5, 3, 2



(b) Delete 4 from the final tree that you drew in 8 (a). Draw this final tree. (2 Points)

Project Questions (18 points)

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

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} and **k** = 36. Is there a subset of X which sums to k?
Yes (Yes/No (2 points)

(b) The type of problem you were asked to solve in question 10 (a) is (Circle one answer): (2 Points)

1. an optimization problem.
2. a problem that is impossible to solve.
3. a decision problem.
4. a problem that has been proven to take exponential time to solve.
5. a problem that has been proven to take factorial time to solve.

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

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

(d) 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 decreasing sequence, (4 points)

```
public static boolean subSetSum(int[] x, int n, int k) {
// pre: x is a super decreasing sequence with 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.
INT K = 0
while (K < N){
   IF (X[K] <= K)  K = K - X[K]
}  K = K+1
   return (K == 0)
```

(e) What is the worst case run time complexity of your code in part (d)? Use Big Theta. (2 Points) $\Theta(N)$

(f) In Project 3, we wrote a calculator that processed boolean expressions. Evaluate the following boolean expressions in the same manner as Project 3. (2 Points)

1 0 + 1 0 * *	0
0 1 + 0 1 + *	1
1 1 * 1 0 + *	1
0 - 0 +	1

TAKE A FULL POINT FOR EACH WRONG ANSWER. MAX OFF OF 2.

(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 level order traversal that displayed the values of this balanced tree. Which of the following is true of the level order traversal? (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)

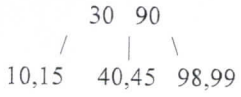
Project 3 was written to make good use of a Red Black Tree.

(h) The tree was used for (circle one answer). (2 Points)

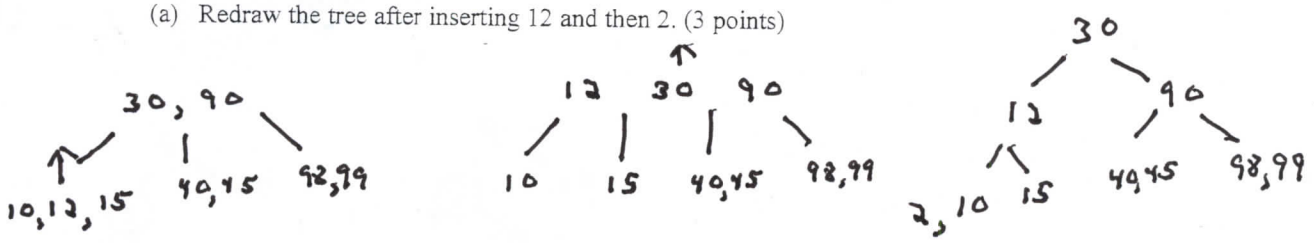
1. Maintain a stack of variable names.
2. Maintain a stack of values.
3. Maintain a set of variables quickly searched by value.
4. Maintain a set of name value pairs quickly searched by name.
5. Maintain a set of name value pairs quickly searched by value.

Balanced Search Trees (15 points)

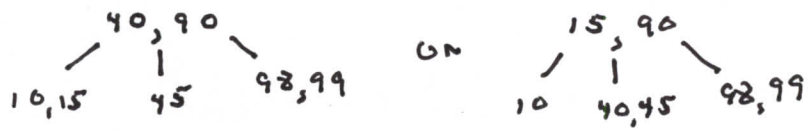
11. Consider the following B-Tree with a minimum of one and a maximum of two.



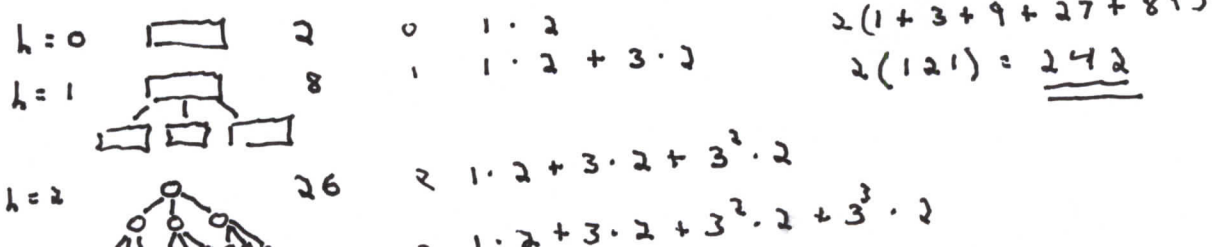
(a) Redraw the tree after inserting 12 and then 2. (3 points)



(b) Delete the value 30 from the B-Tree shown above. Begin work from the original tree, not the tree with the values 12 and 2. (2 points)



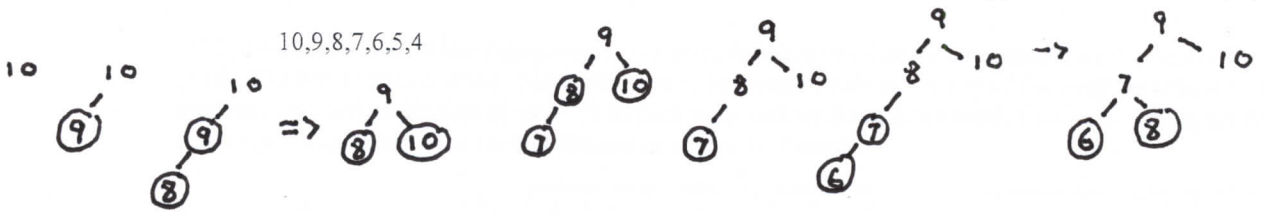
(c) The tree in question 11a has a height of 1. What is the maximum number of keys that this type of tree (min = 1, max=2) could hold with a height of 4? (3 points).



12. Red Black Trees

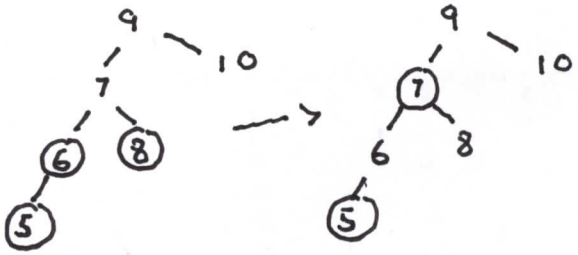
(a) Insert the following numbers, one by one, into a Red-Black Tree. Show the tree after each insertion. red vertices should be circled and black vertices should appear without circles. (5 points)

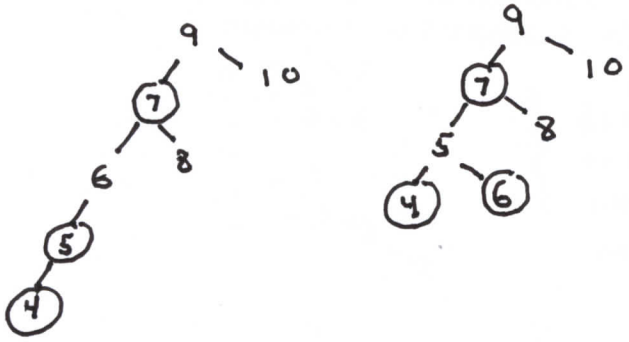
10,9,8,7,6,5,4



(b) What is the best-case runtime complexity of a Red-Black Tree insertion ? Use Big Theta notation. (1 Point) $\Theta(\log N)$ or $\Theta(\log N)$

(c) What is the best-case runtime complexity of a Binary Search Tree insertion? Use Big Theta notation. (1 point) $\Theta(1)$



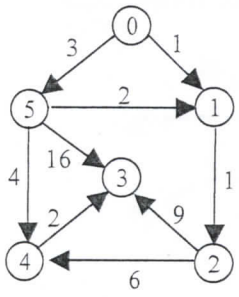


20 21

Graph Algorithms(~~25~~ points)

13. (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) ~~(0,1),(1,2),(2,3)~~ (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)

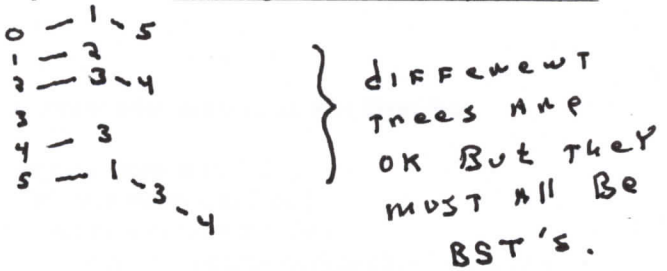


	0	1	2	3	4	5
0	0	?	?	?	?	?
1	0	1				3
2	0	1	2			3
5	0	1	2	18	8	3
4	0	1	2	11	7	3
3	0	1	2	9	7	3

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

	0	1	2	3	4	5
0		1				3
1			1			
2				9	6	
3						
4				2		
5		2		16	4	

(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)

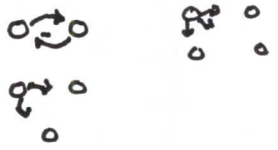


(e) The graph shown above is a simple, directed graph. It contains no loops or multiple edges. Suppose we have such a graph with V vertices. What is the maximum number of edges that such a graph can contain? Your answer should be an exact formula. Answers in Big Theta notation don't count. (1 Point)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

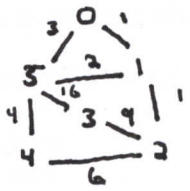
or

$V^2 - V$

e	V
0	1
2	2
6	3
12	4



(f) Redraw the graph in question 13 (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 13 (a). (1 Point)



(g) Draw the tree that would be computed by Prim using the graph that you drew in question 13 (f) as input. This is the tree computed by Prim of the undirected, weighted graph you created in 13 (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)

(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
```

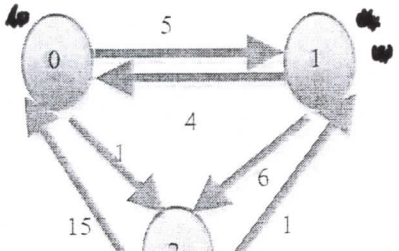
Cost

	0	1	2
0		5	1
1	4		6
2	15	1	

	0	1	2
0			
1			5
2	15		

	0	1	2
0			
1			
2	5		

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



3

15

15

16

18

15

21

100